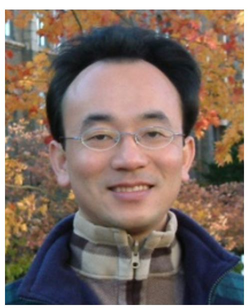
國立臺北護理健康大學
National Taipei University of Nursing and Health Sciences

Tamkang University
淡江大學

人工智慧文本分析基礎與應用
(Artificial Intelligence for Text Analytics: Foundations and Applications)

Time: 2020/05/22 (Fri) (9:10 -12:00)
Place: 國立臺北護理健康大學 (台北市明德路365號) G210
Host: 祝國忠 院長 (健康科技學院院長)



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/

2020-05-22

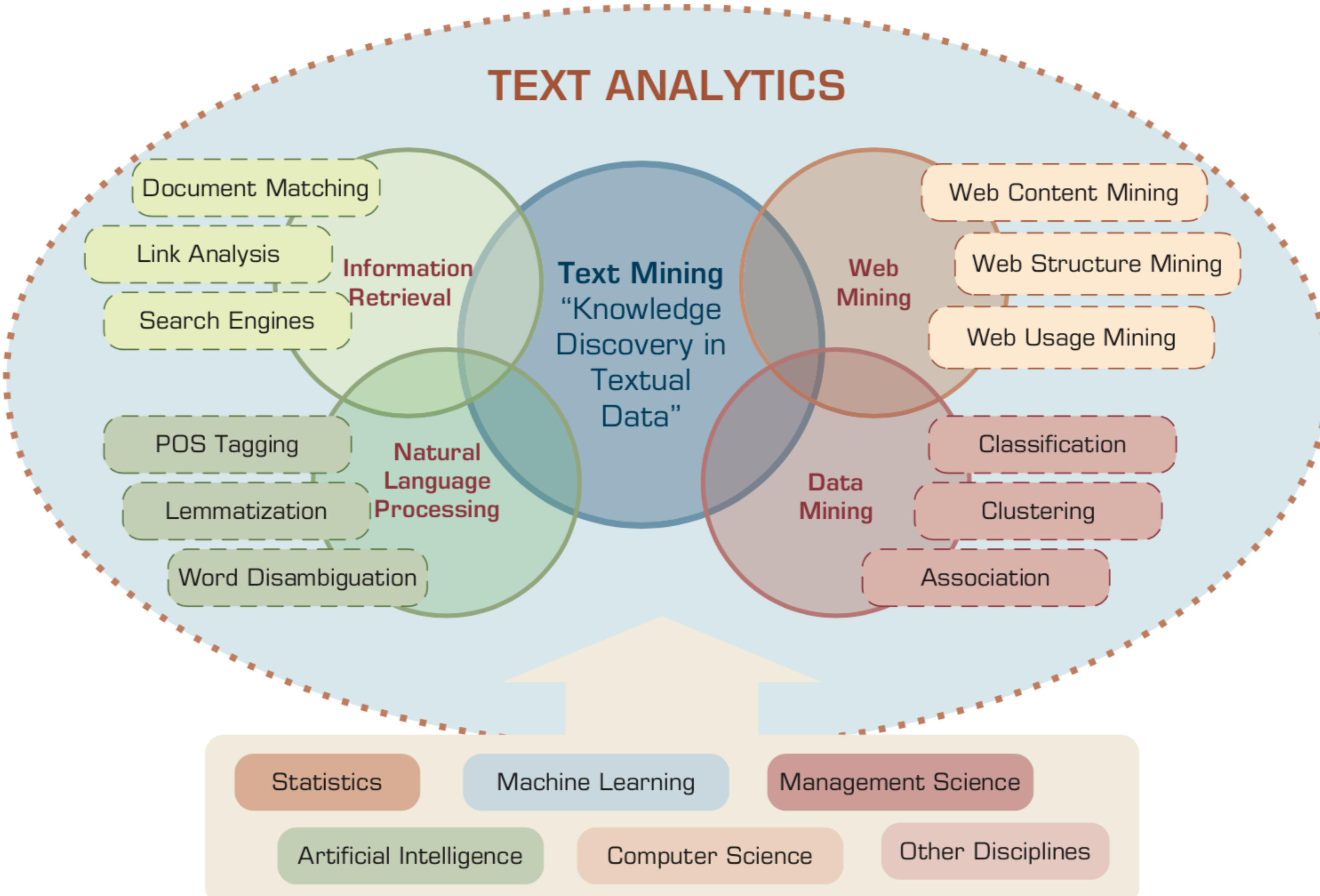
Topics

1. **自然語言處理核心技術與文字探勘**
   (Core Technologies of Natural Language Processing and Text Mining)
2. **人工智慧文本分析基礎與應用**
   (Artificial Intelligence for Text Analytics: Foundations and Applications)
3. 文本表達特徵工程
   (Feature Engineering for Text Representation)
4. 語意分析和命名實體識別
   (Semantic Analysis and Named Entity Recognition; NER)
5. 深度學習和通用句子嵌入模型
   (Deep Learning and Universal Sentence-Embedding Models)
6. 問答系統與對話系統
   (Question Answering and Dialogue Systems)

Outline

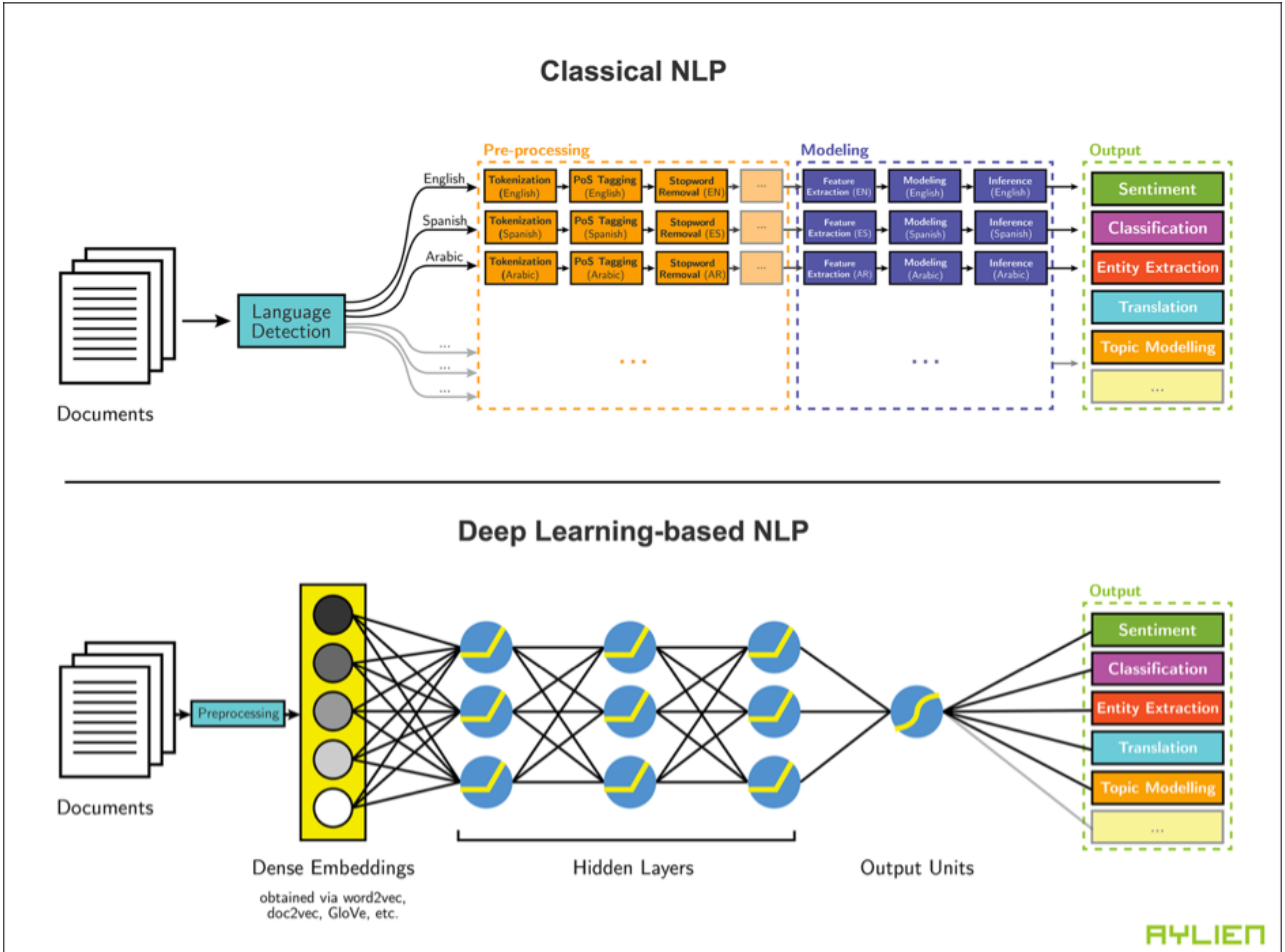
- Python for
  Natural Language Processing
- Processing Text and
  Understanding Text

Text Analytics and Text Mining



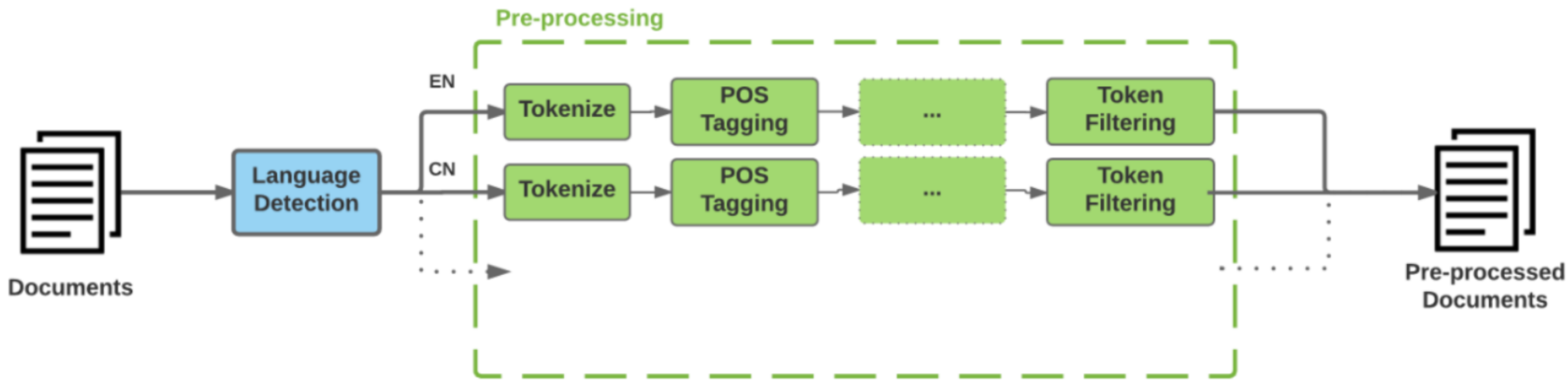
Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

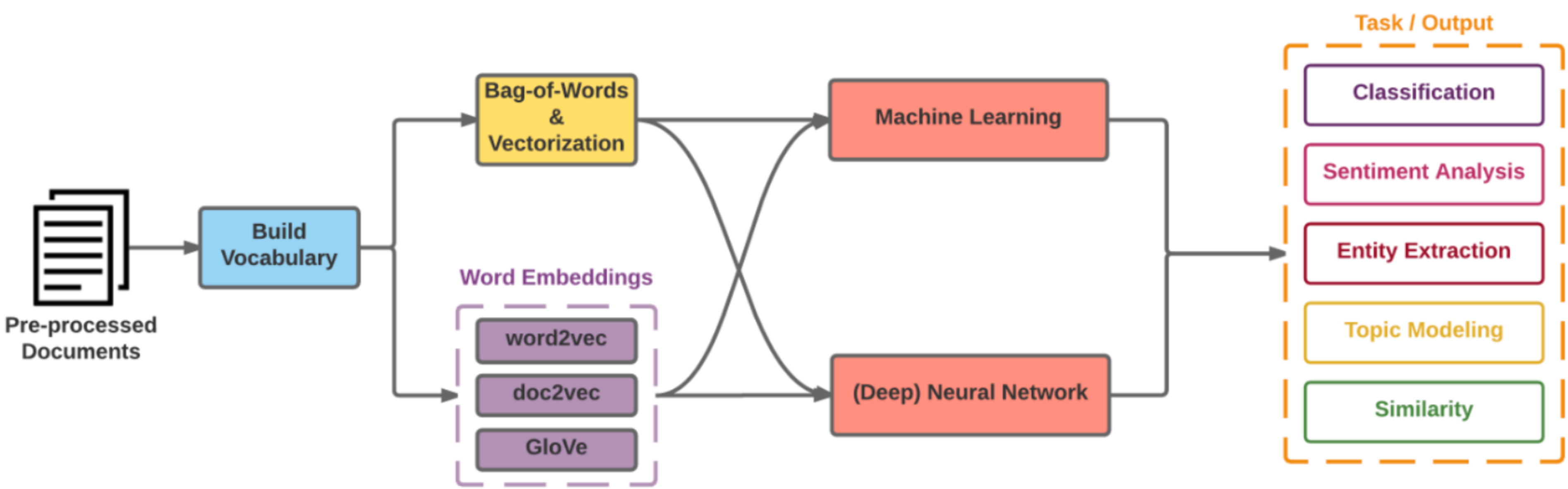
NLP



Source: http://blog.aylien.com/leveraging-deep-learning-for-multilingual/

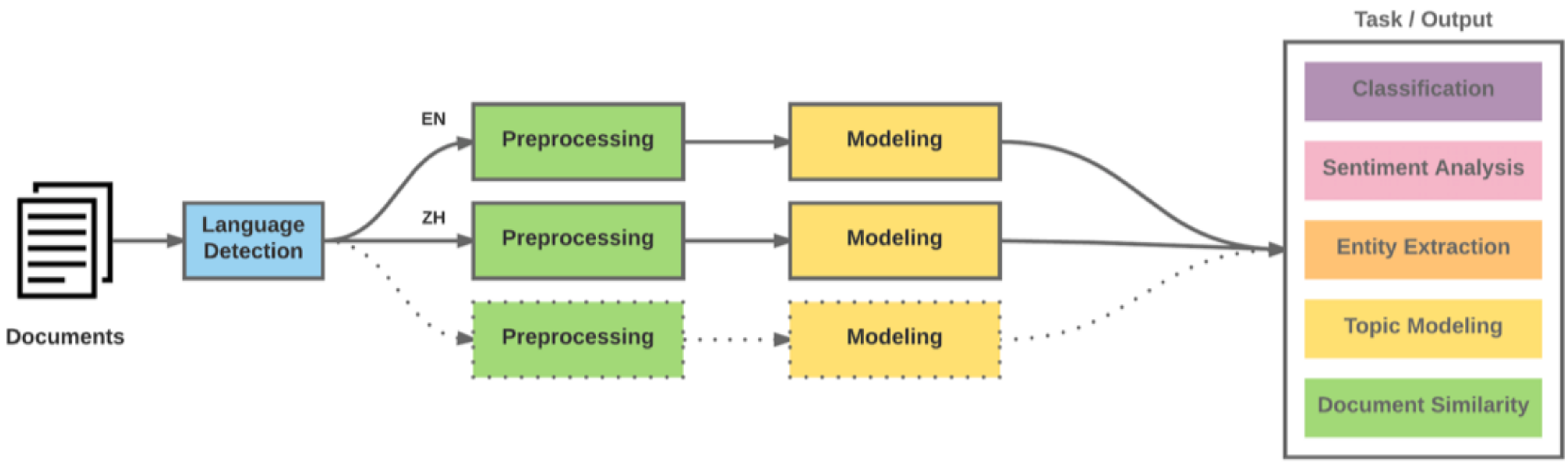
Modern NLP Pipeline





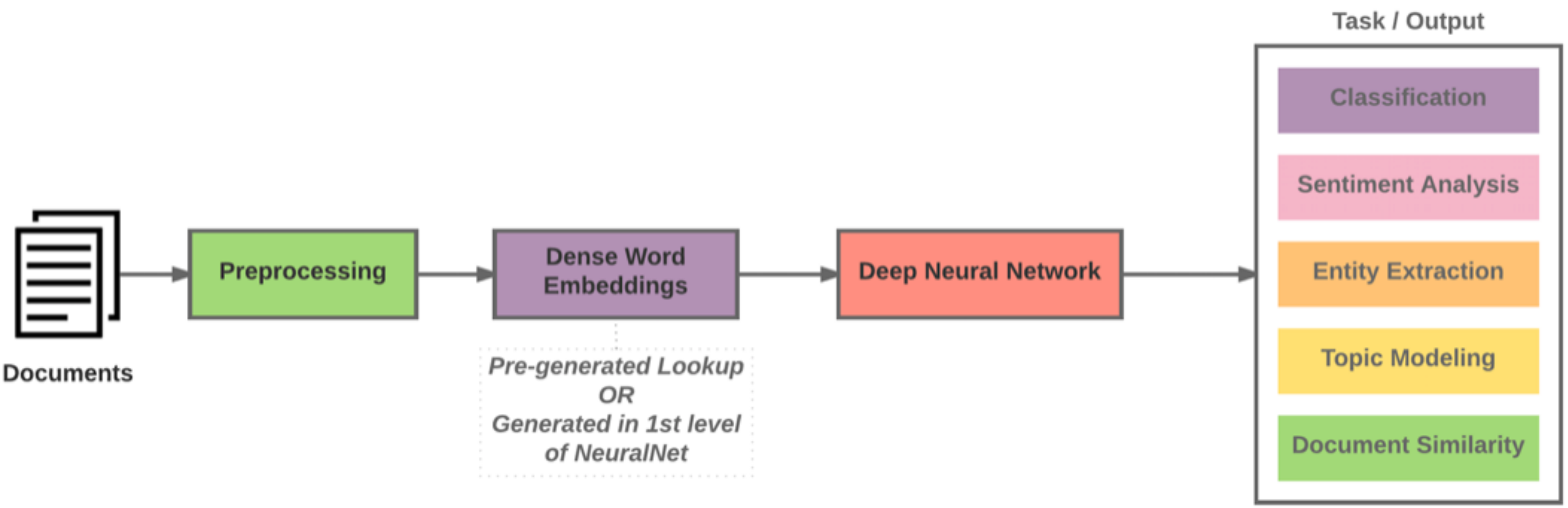
Source: https://github.com/fortiema/talks/blob/master/opendata2016sh/pragmatic-nlp-opendata2016sh.pdf

Modern NLP Pipeline



Source: http://mattfortier.me/2017/01/31/nlp-intro-pt-1-overview/

Deep Learning NLP



Source: http://mattfortier.me/2017/01/31/nlp-intro-pt-1-overview/

Papers with Code: NLP

Browse > Natural Language Processing

Natural Language Processing

500 leaderboards • 249 tasks • 100 datasets • 5219 papers with code

Representation Learning

Task	Leaderboards	Papers
Representation Learning	7 leaderboards	548 papers with code
Word Embeddings		454 papers with code
Graph Embedding		116 papers with code
Network Embedding		62 papers with code
Sentence Embeddings	3 leaderboards	52 papers with code

▸ See all 17 tasks

Machine Translation

Task	Leaderboards	Papers
Machine Translation	45 leaderboards	612 papers with code
Transliteration		17 papers with code
Unsupervised Machine Translation	9 leaderboards	12 papers with code
Low-Resource Neural Machine Translation		8 papers with code
Multimodal Machine Translation		7 papers with code

▸ See all 6 tasks

Question Answering

https://paperswithcode.com/area/natural-language-processing

NLP Benchmark Datasets

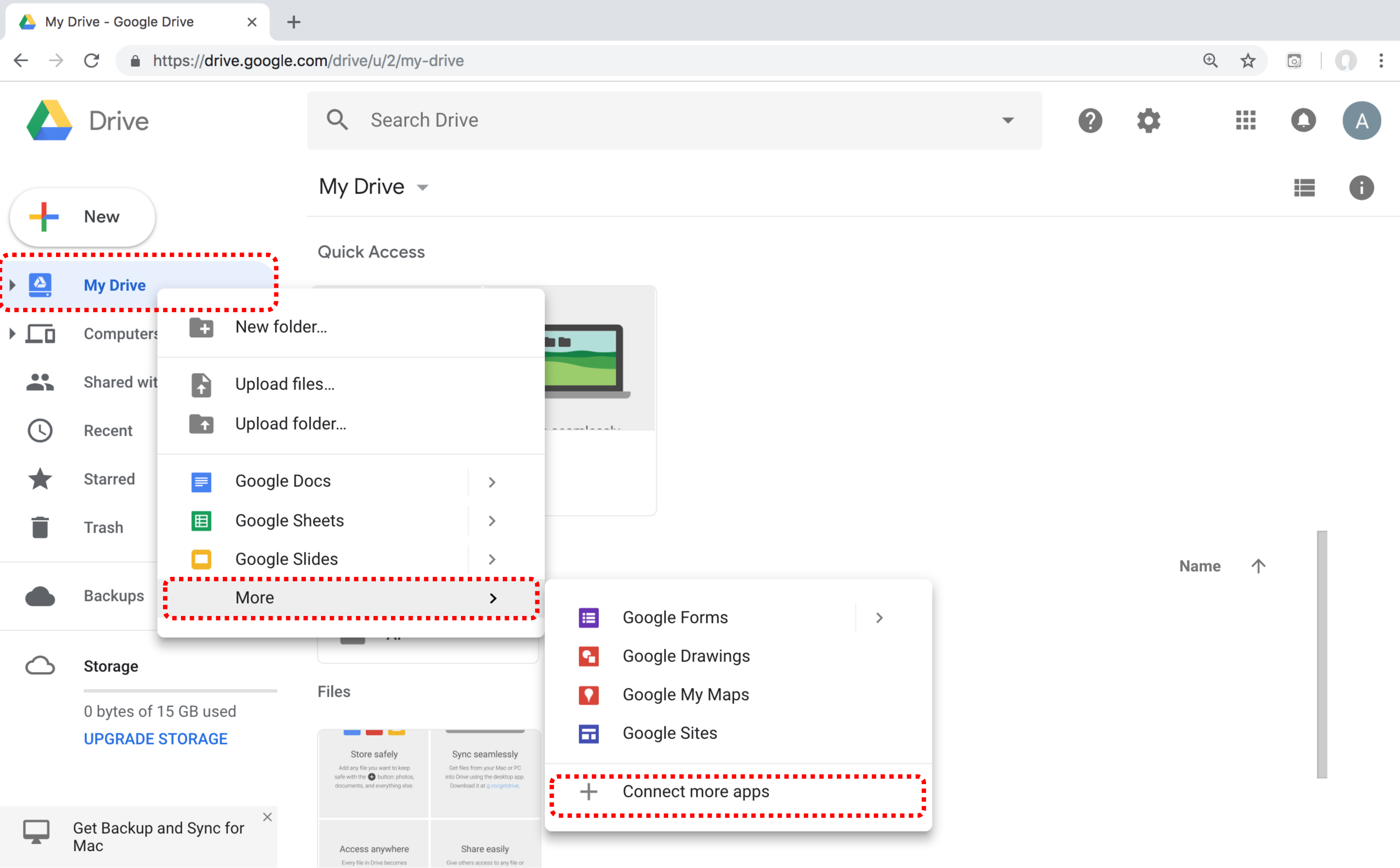
Task	Dataset	Link
Machine Translation	WMT 2014 EN-DE WMT 2014 EN-FR	http://www-lium.univ-lemans.fr/~schwenk/cslm_joint_paper/
Text Summarization	CNN/DM Newsroom DUC Gigaword	https://cs.nyu.edu/~kcho/DMQA/ https://summari.es/ https://www-nlpir.nist.gov/projects/duc/data.html https://catalog.ldc.upenn.edu/LDC2012T21
Reading Comprehension Question Answering Question Generation	ARC CliCR CNN/DM NewsQA RACE SQuAD Story Cloze Test NarativeQA Quasar SearchQA	http://data.allenai.org/arc/ http://aclweb.org/anthology/N18-1140 https://cs.nyu.edu/~kcho/DMQA/ https://datasets.maluuba.com/NewsQA http://www.qizhexie.com/data/RACE_leaderboard https://rajpurkar.github.io/SQuAD-explorer/ http://aclweb.org/anthology/W17-0906.pdf https://github.com/deepmind/narrativeqa https://github.com/bdhingra/quasar https://github.com/nyu-dl/SearchQA
Semantic Parsing	AMR parsing ATIS (SQL Parsing) WikiSQL (SQL Parsing)	https://amr.isi.edu/index.html https://github.com/jkkummerfeld/text2sql-data/tree/master/data https://github.com/salesforce/WikiSQL
Sentiment Analysis	IMDB Reviews SST Yelp Reviews Subjectivity Dataset	http://ai.stanford.edu/~amaas/data/sentiment/ https://nlp.stanford.edu/sentiment/index.html https://www.yelp.com/dataset/challenge http://www.cs.cornell.edu/people/pabo/movie-review-data/
Text Classification	AG News DBpedia TREC 20 NewsGroup	http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html https://wiki.dbpedia.org/Datasets https://trec.nist.gov/data.html http://qwone.com/~jason/20Newsgroups/
Natural Language Inference	SNLI Corpus MultiNLI SciTail	https://nlp.stanford.edu/projects/snli/ https://www.nyu.edu/projects/bowman/multinli/ http://data.allenai.org/scitail/
Semantic Role Labeling	Proposition Bank OneNotes	http://propbank.github.io/ https://catalog.ldc.upenn.edu/LDC2013T19

Source: Amirsina Torfi, Rouzbeh A. Shirvani, Yaser Keneshloo, Nader Tavvaf, and Edward A. Fox (2020). "Natural Language Processing Advancements By Deep Learning: A Survey." arXiv preprint arXiv:2003.01200.

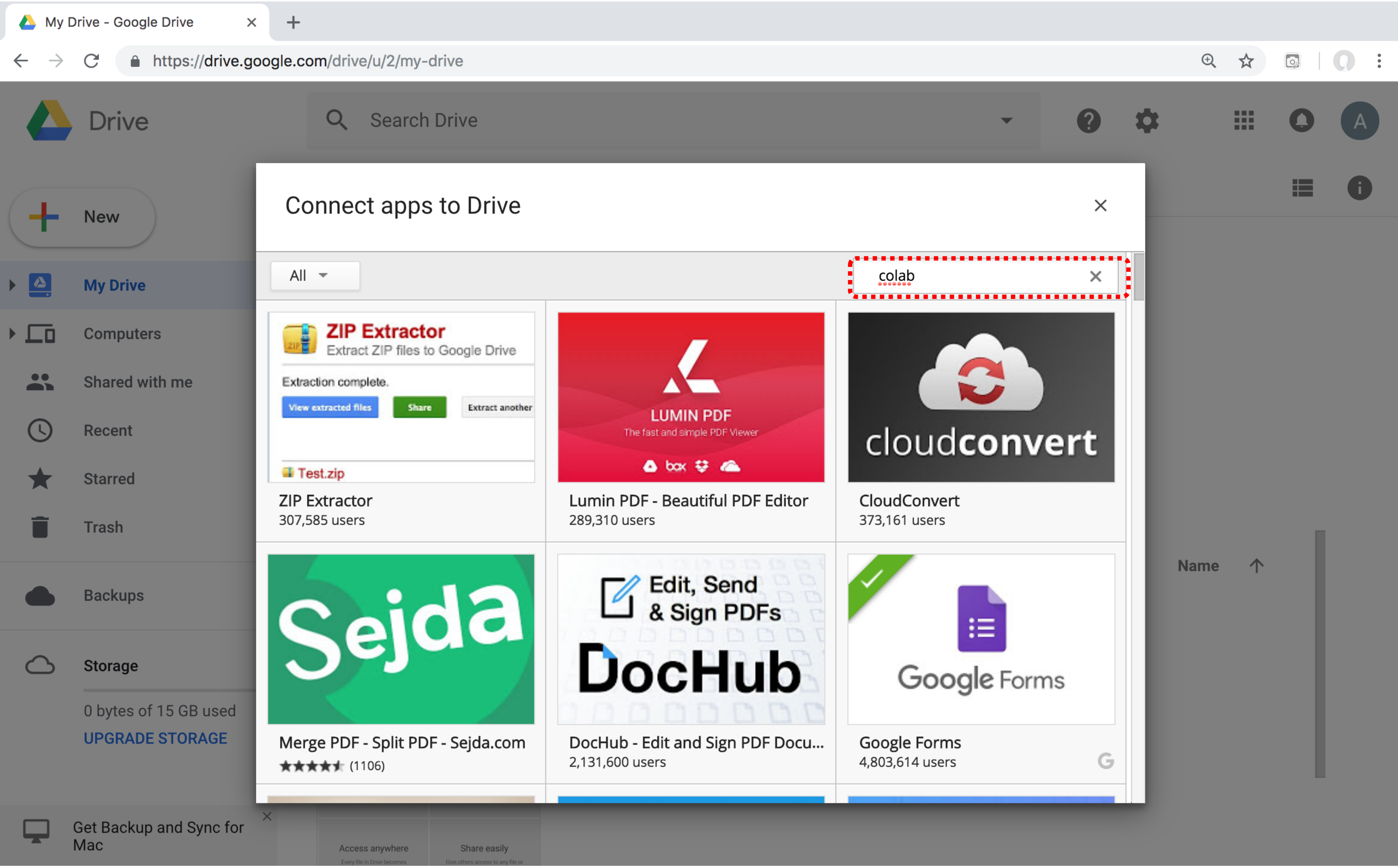
Python for Natural Language Processing

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

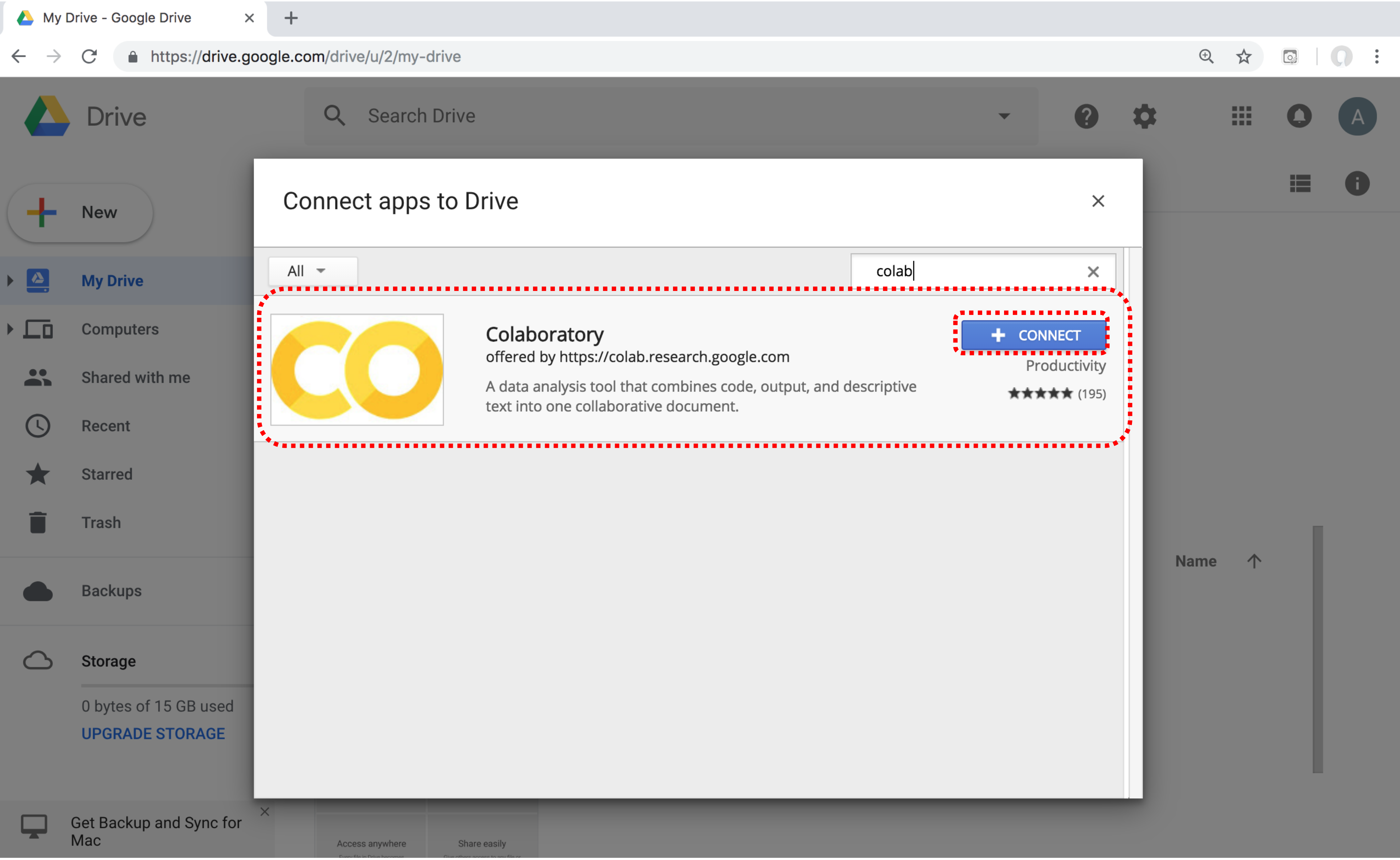
Connect Google Colab in Google Drive



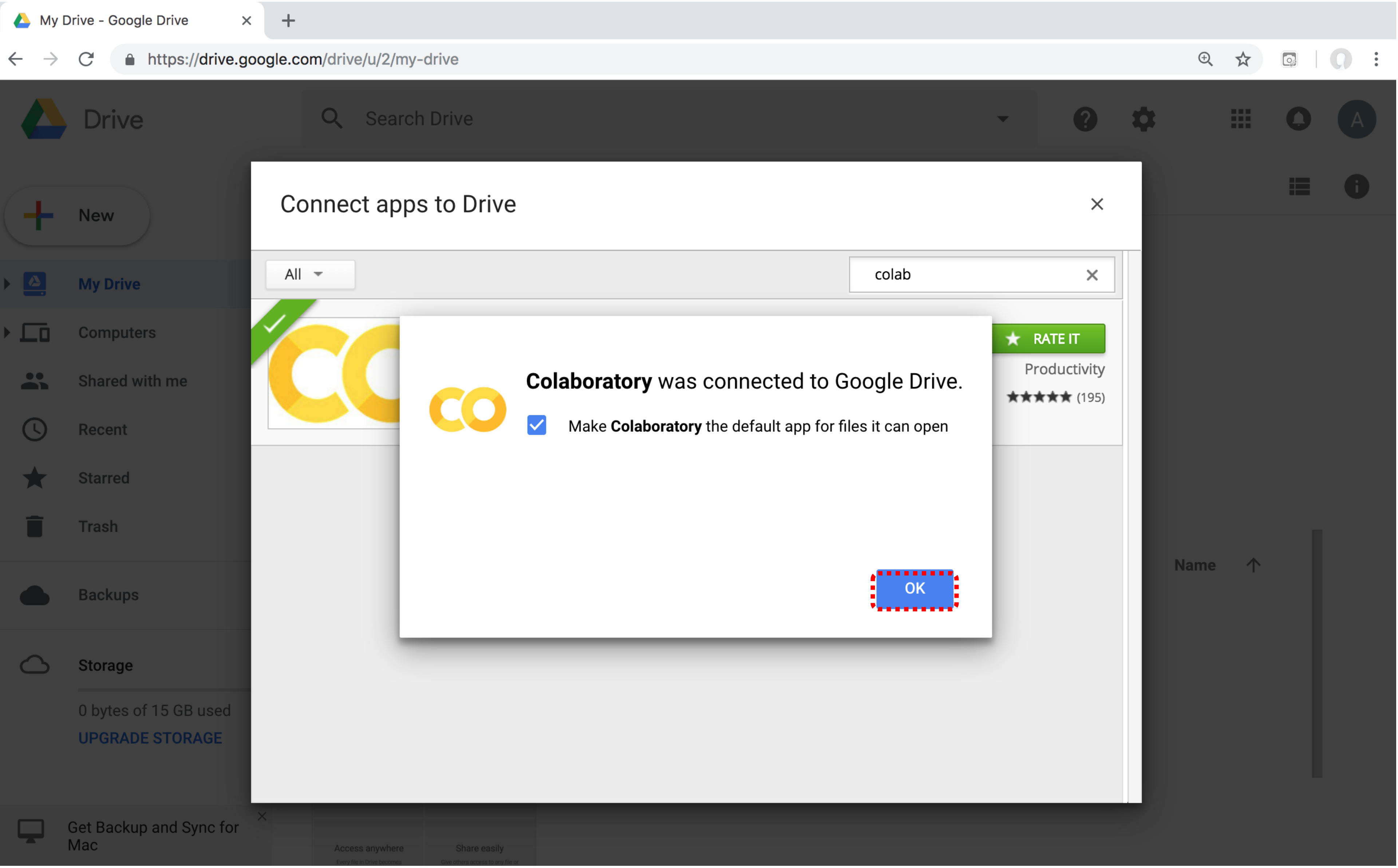
Google Colab



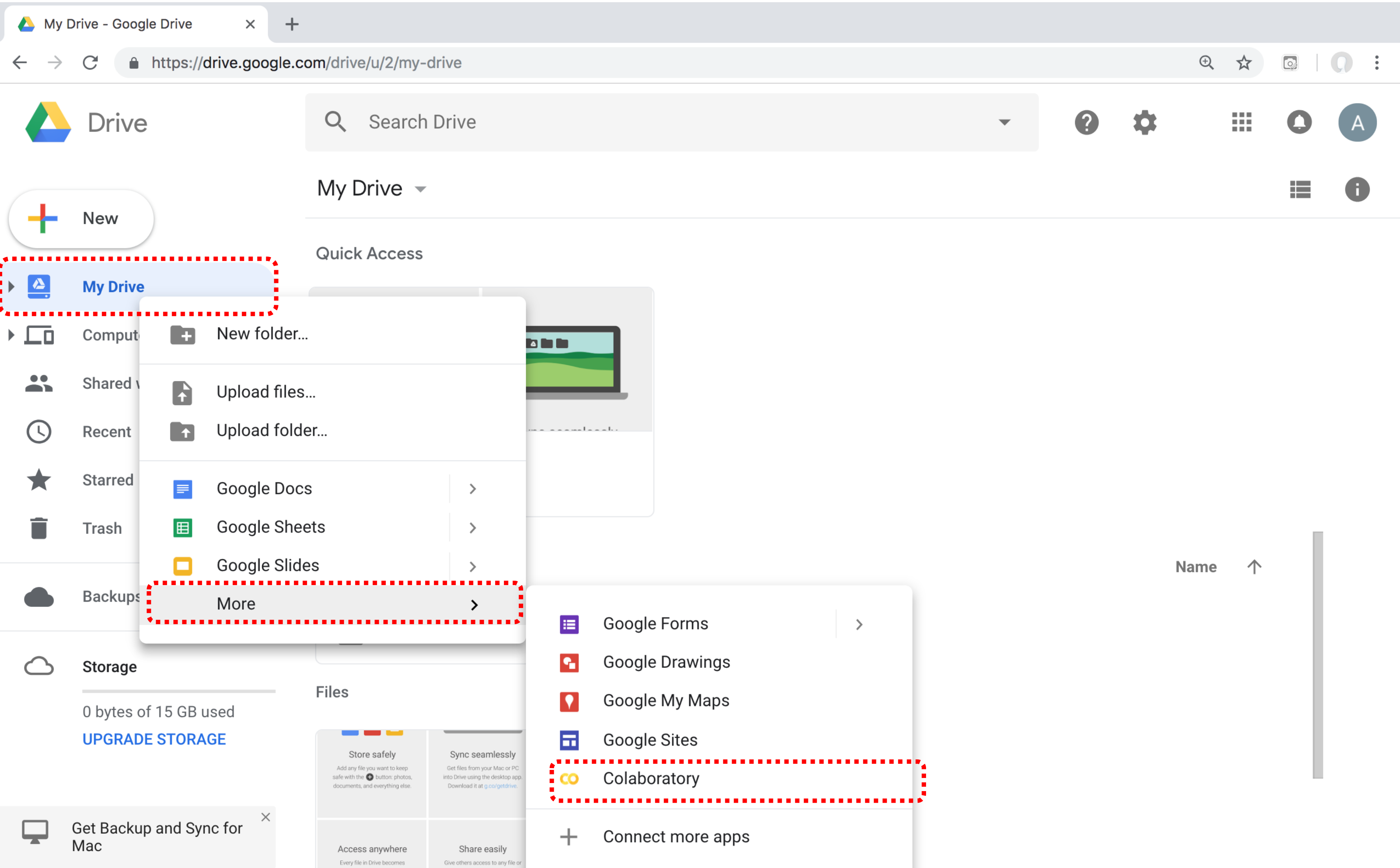
Google Colab



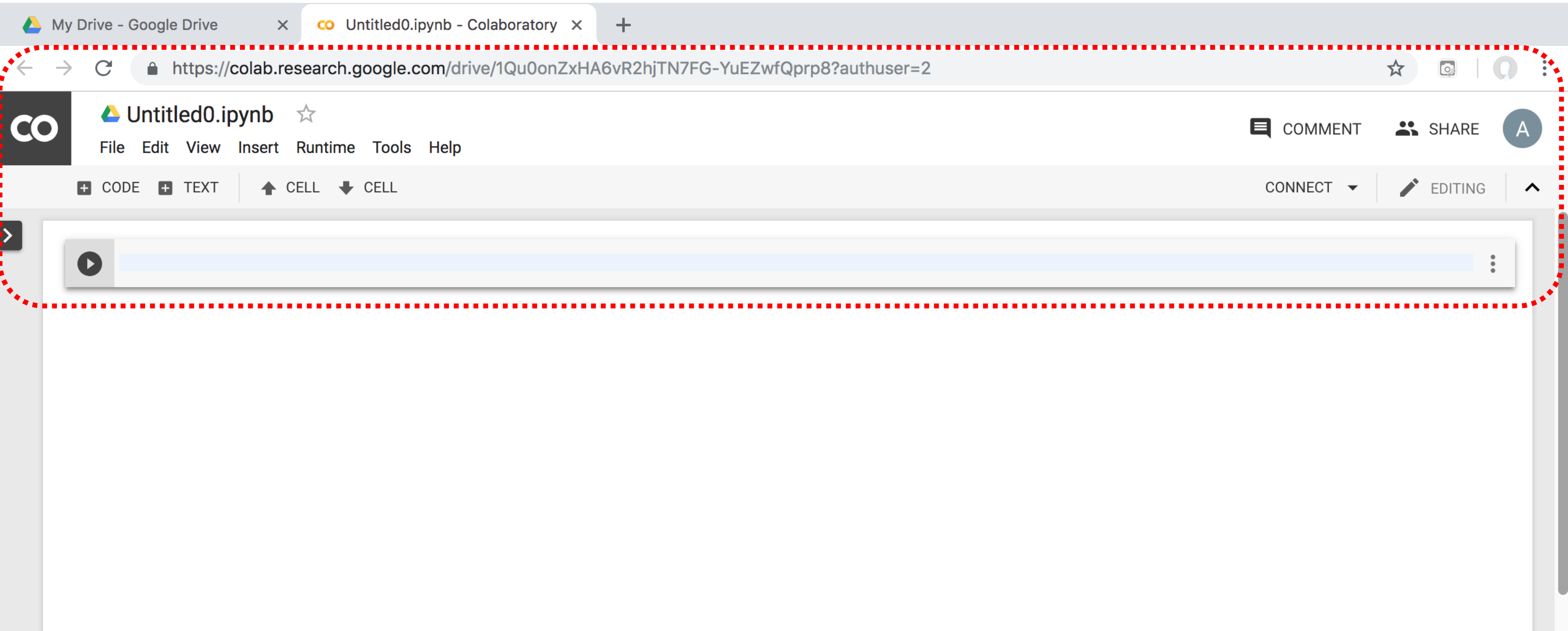
Connect Colaboratory to Google Drive



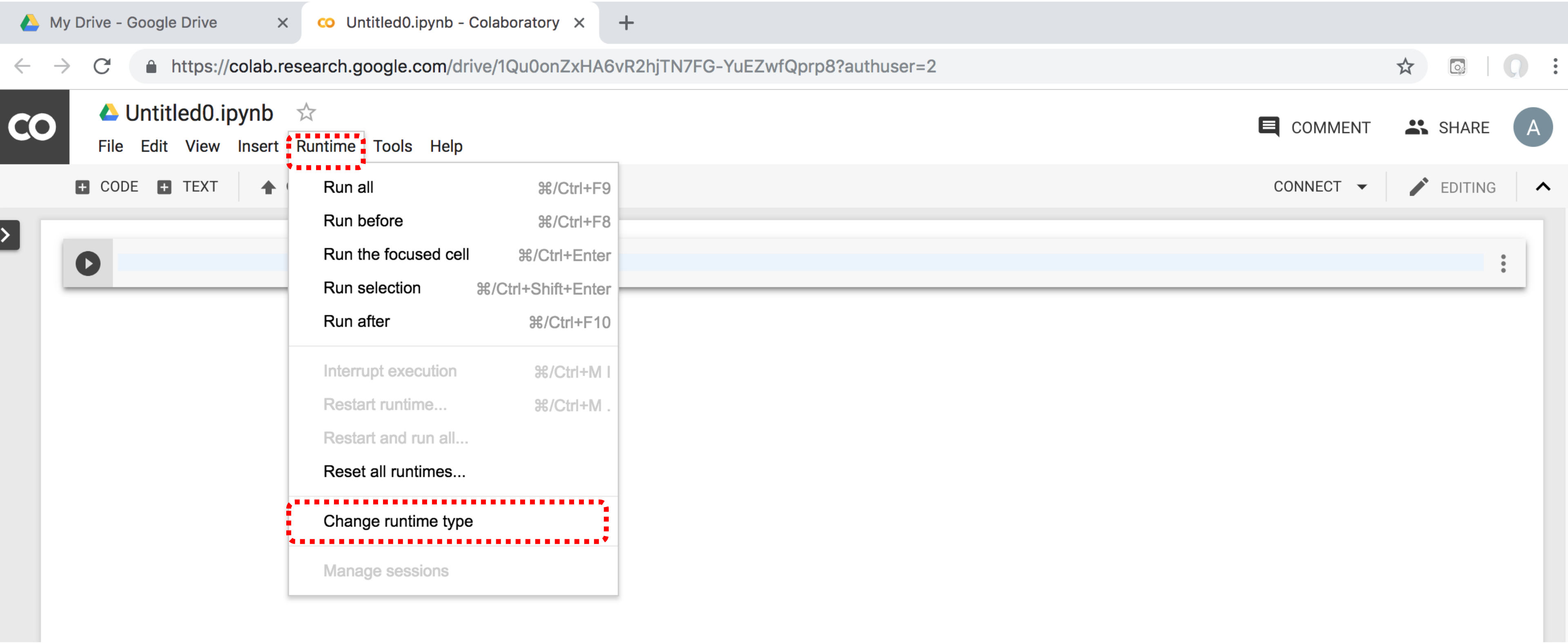
Google Colab



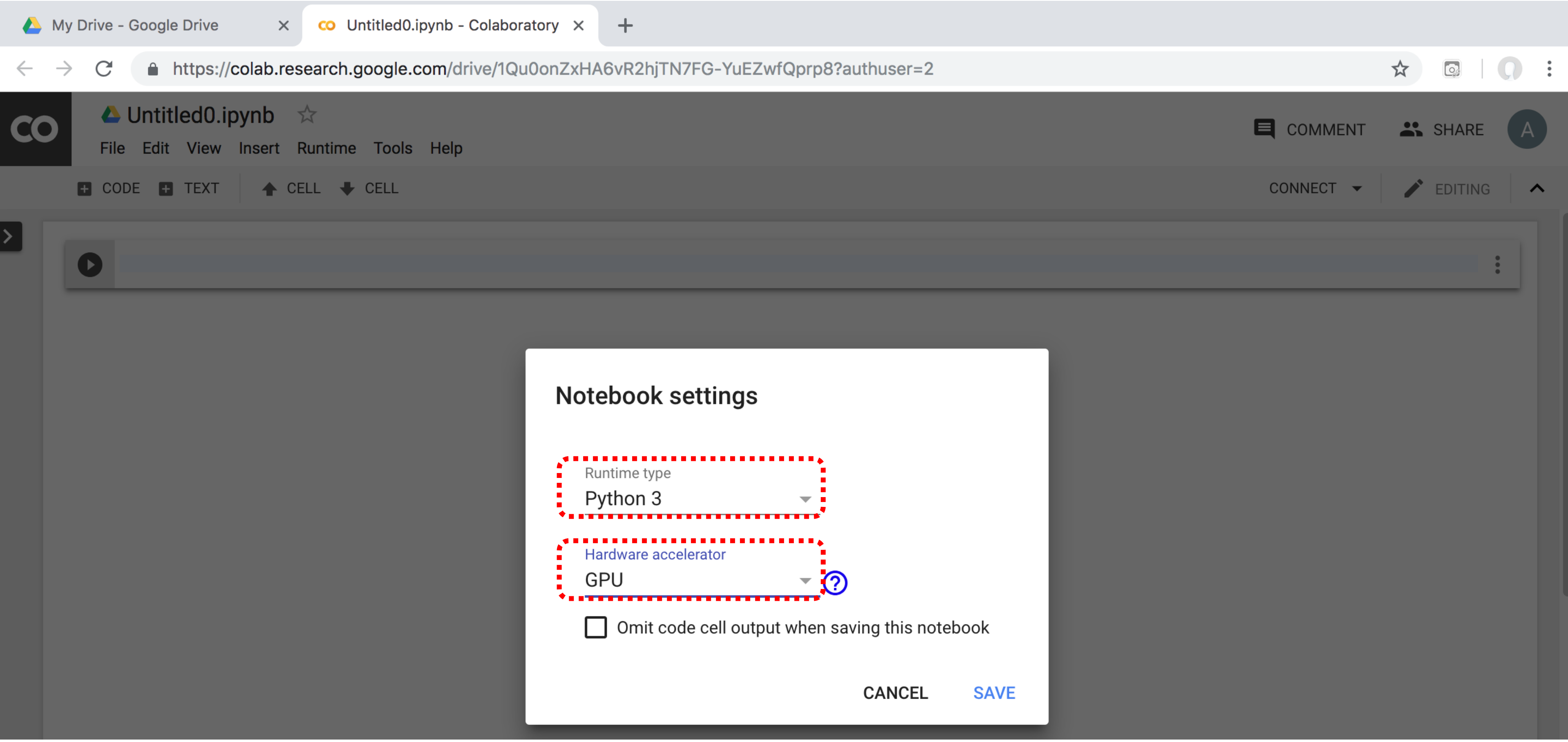
Google Colab



Google Colab



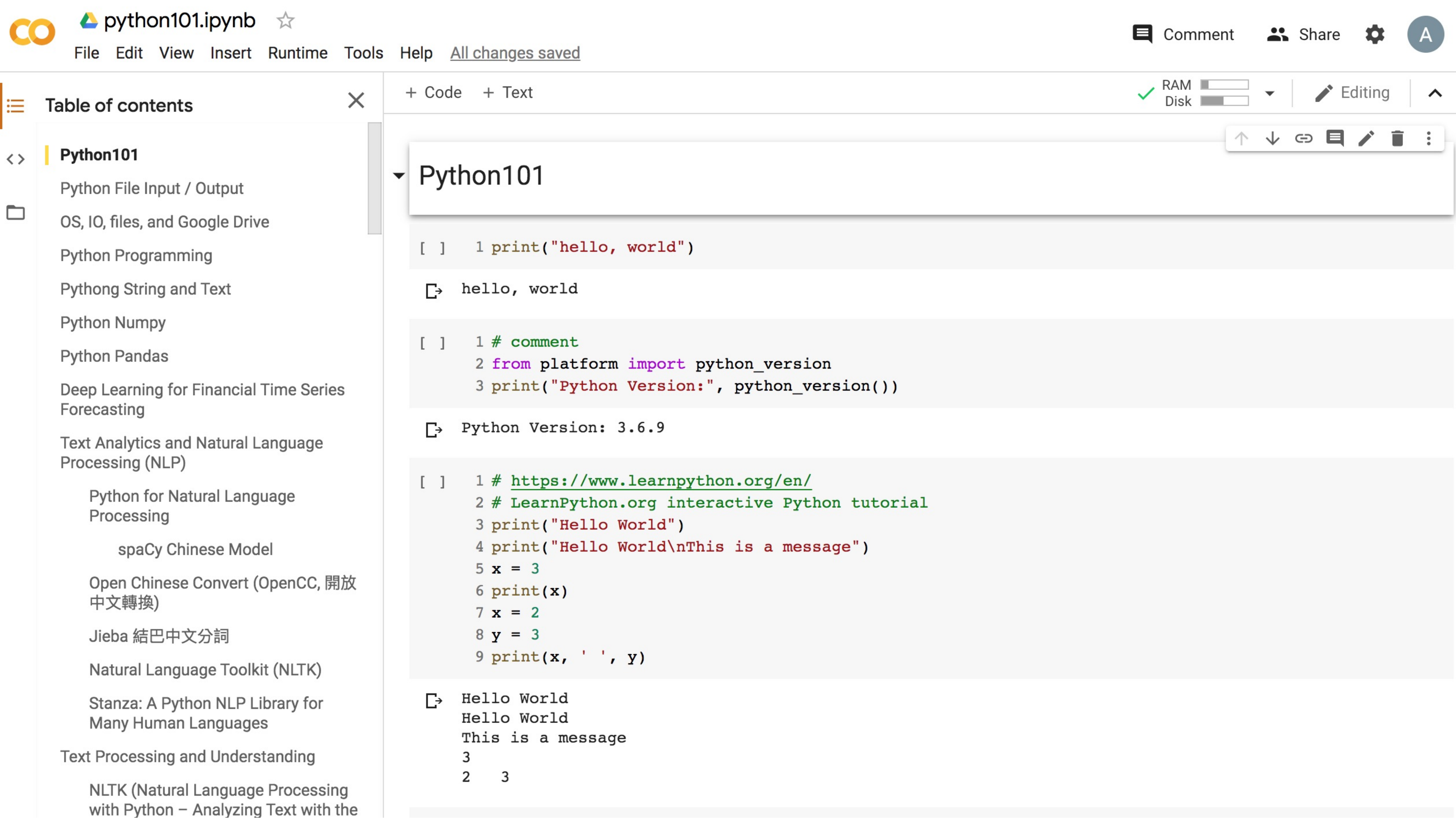
Run Jupyter Notebook Python3 GPU Google Colab



Google Colab Python Hello World
print('Hello World')

Python in Google Colab (Python101)

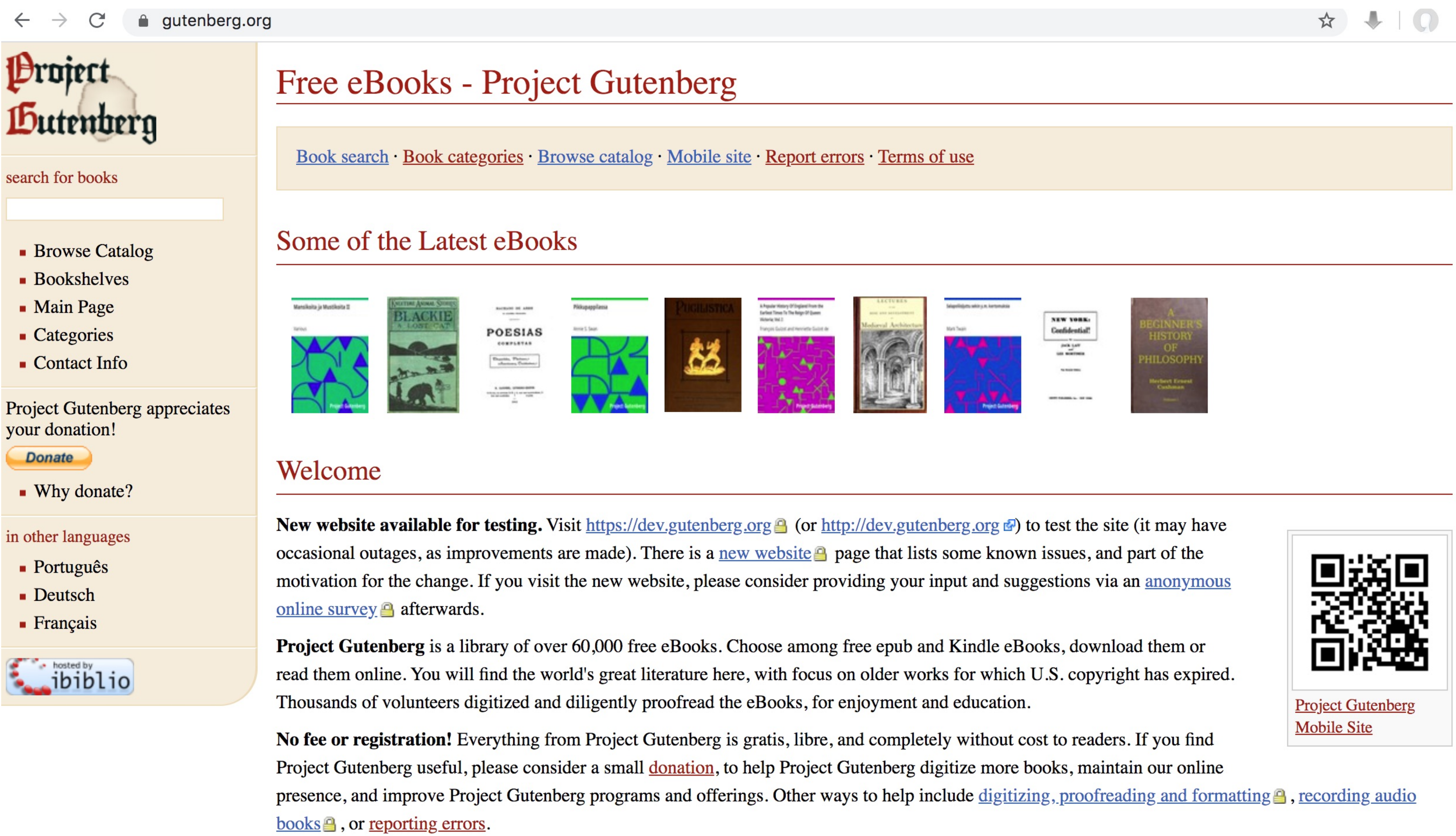
https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT



https://tinyurl.com/imtkupython101

Processing and Understanding Text

Free eBooks - Project Gutenberg



gutenberg.org

Project Gutenberg

search for books

- Browse Catalog
- Bookshelves
- Main Page
- Categories
- Contact Info

Project Gutenberg appreciates your donation!

Donate

- Why donate?

in other languages

- Português
- Deutsch
- Français

hosted by ibiblio

Free eBooks - Project Gutenberg

Book search · Book categories · Browse catalog · Mobile site · Report errors · Terms of use

Some of the Latest eBooks

Welcome

New website available for testing. Visit https://dev.gutenberg.org (or http://dev.gutenberg.org) to test the site (it may have occasional outages, as improvements are made). There is a new website page that lists some known issues, and part of the motivation for the change. If you visit the new website, please consider providing your input and suggestions via an anonymous online survey afterwards.

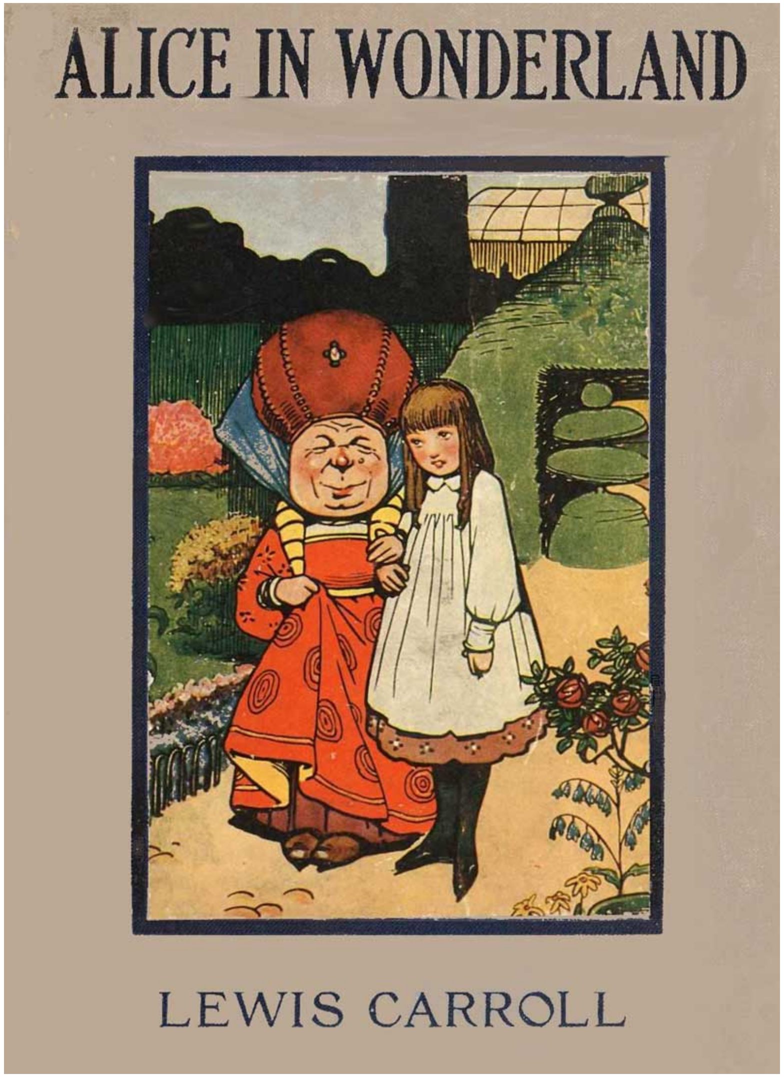
Project Gutenberg is a library of over 60,000 free eBooks. Choose among free epub and Kindle eBooks, download them or read them online. You will find the world's great literature here, with focus on older works for which U.S. copyright has expired. Thousands of volunteers digitized and diligently proofread the eBooks, for enjoyment and education.

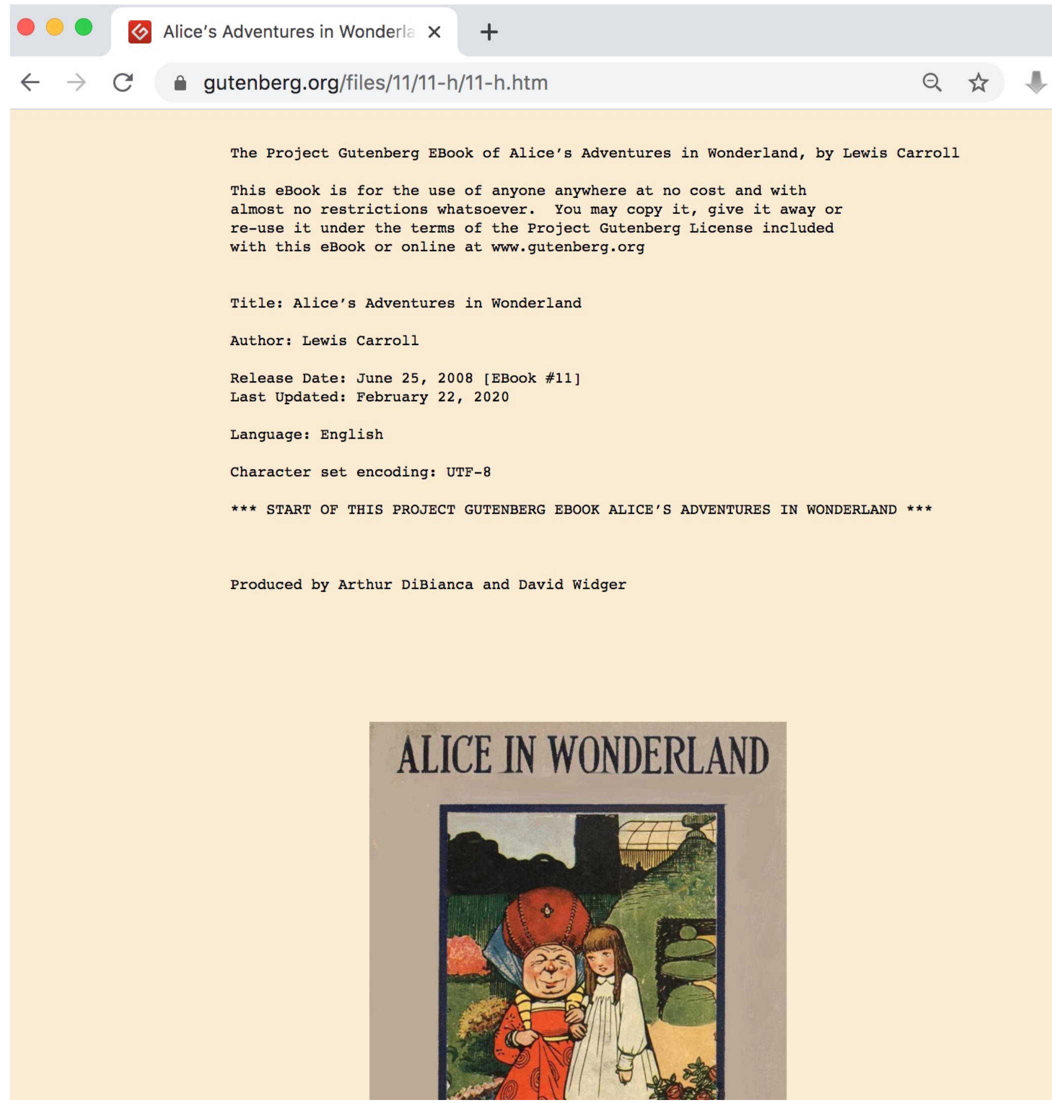
No fee or registration! Everything from Project Gutenberg is gratis, libre, and completely without cost to readers. If you find Project Gutenberg useful, please consider a small donation, to help Project Gutenberg digitize more books, maintain our online presence, and improve Project Gutenberg programs and offerings. Other ways to help include digitizing, proofreading and formatting, recording audio books, or reporting errors.

Project Gutenberg Mobile Site

https://www.gutenberg.org/

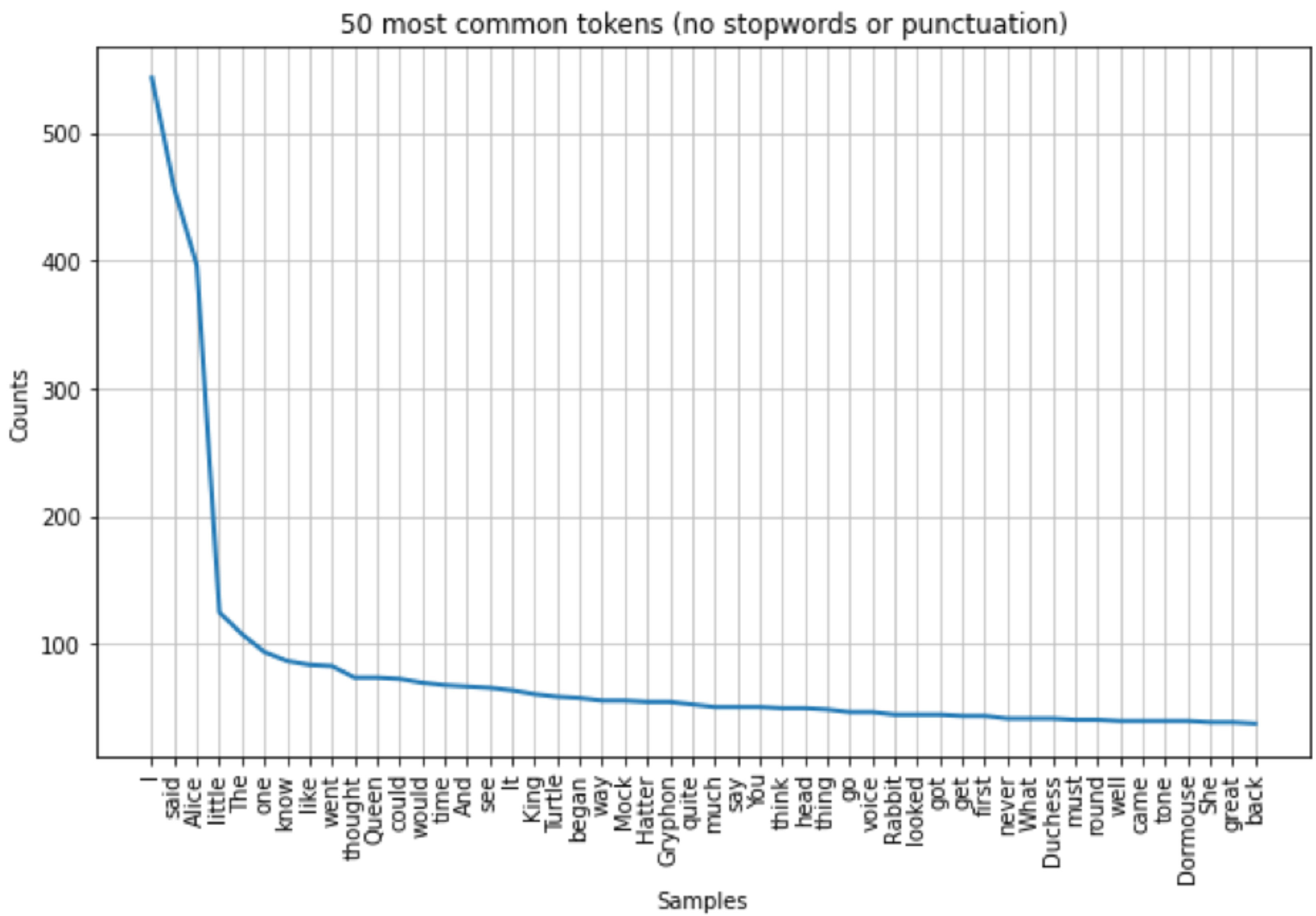
Free eBooks - Project Gutenberg
Alice in Wonderland





https://www.gutenberg.org/files/11/11-h/11-h.htm

Alice Top 50 Tokens



https://tinyurl.com/imtkupython101

Python in Google Colab (Python101)

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

```
nltk.download('gutenberg')
alice = Text(nltk.corpus.gutenberg.words('carroll-alice.txt'))
```

python101.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text RAM Disk Editing

Text Processing and Understanding

- Vincent Russo (2018), Natural Language Processing in Python: Part 1 -- Introduction, https://vprusso.github.io/blog/2018/natural-language-processing-python-1/
- NLTK (Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit) Book: https://www.nltk.org/book/
- Beautiful Soup 4 (BS4): https://www.crummy.com/software/BeautifulSoup/bs4/doc/

```
[ ]
!pip install nltk
import nltk
nltk.download('gutenberg')
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.6/dist-packages (3.2.5)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from nltk) (1.12.0)
[nltk_data] Downloading package gutenberg to /root/nltk_data...
[nltk_data]   Unzipping corpora/gutenberg.zip.
True
```

```
[ ]
from nltk.text import Text
alice = Text(nltk.corpus.gutenberg.words('carroll-alice.txt'))
alice
```

```
<Text: Alice ' s Adventures in Wonderland by Lewis Carroll 1865>
```

```
[ ]
print(nltk.corpus.gutenberg.fileids())
```

```
['austen-emma.txt', 'austen-persuasion.txt', 'austen-sense.txt', 'bible-kjv.txt', 'blake-poems.txt', 'bryant-stories.txt', 'burgess-buster
```

https://tinyurl.com/imtkupython101

alice.concordance("Alice")

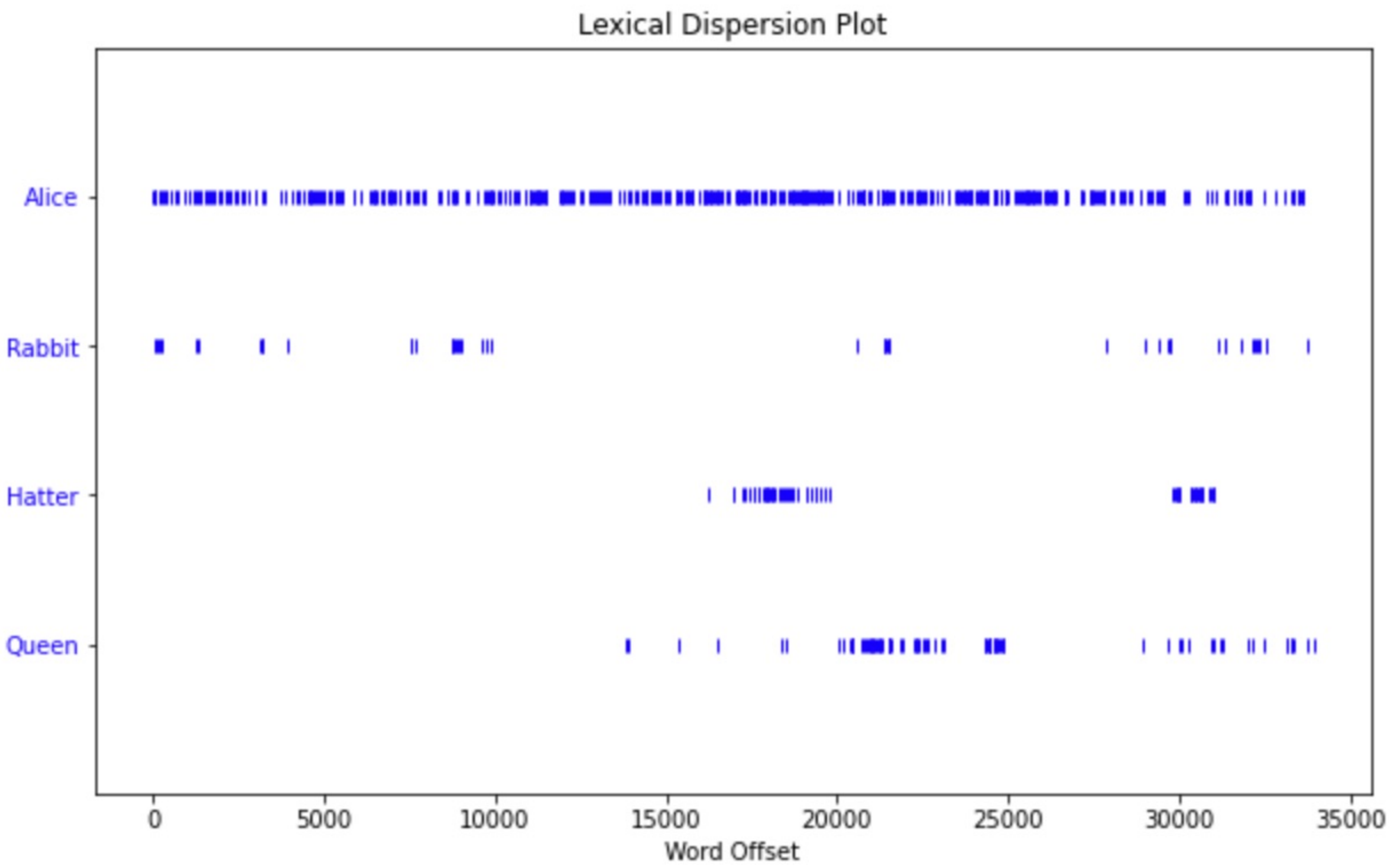
```
1 alice.concordance("Alice")
```

```
Displaying 25 of 398 matches:
                                      Alice ' s Adventures in Wonderland by Lewi
] CHAPTER I . Down the Rabbit - Hole Alice was beginning to get very tired of s
what is the use of a book ,' thought Alice ' without pictures or conversation ?
so VERY remarkable in that ; nor did Alice think it so VERY much out of the way
looked at it , and then hurried on , Alice started to her feet , for it flashed
 hedge . In another moment down went Alice after it , never once considering ho
ped suddenly down , so suddenly that Alice had not a moment to think about stop
she fell past it . ' Well !' thought Alice to herself , ' after such a fall as
down , I think --' ( for , you see , Alice had learnt several things of this so
tude or Longitude I ' ve got to ?' ( Alice had no idea what Latitude was , or L
 . There was nothing else to do , so Alice soon began talking again . ' Dinah '
cats eat bats , I wonder ?' And here Alice began to get rather sleepy , and wen
dry leaves , and the fall was over . Alice was not a bit hurt , and she jumped
 not a moment to be lost : away went Alice like the wind , and was just in time
 but they were all locked ; and when Alice had been all the way down one side a
on it except a tiny golden key , and Alice ' s first thought was that it might
and to her great delight it fitted ! Alice opened the door and found that it le
ead would go through ,' thought poor Alice , ' it would be of very little use w
ay things had happened lately , that Alice had begun to think that very few thi
ertainly was not here before ,' said Alice ,) and round the neck of the bottle
ay ' Drink me ,' but the wise little Alice was not going to do THAT in a hurry
bottle was NOT marked ' poison ,' so Alice ventured to taste it , and finding i
* * ' What a curious feeling !' said Alice ; ' I must be shutting up like a tel
 for it might end , you know ,' said Alice to herself , ' in my going out altog
garden at once ; but , alas for poor Alice ! when she got to the door , she fou
```

https://tinyurl.com/imtkupython101

```
alice.dispersion_plot(["Alice", "Rabbit",
"Hatter", "Queen"])
```

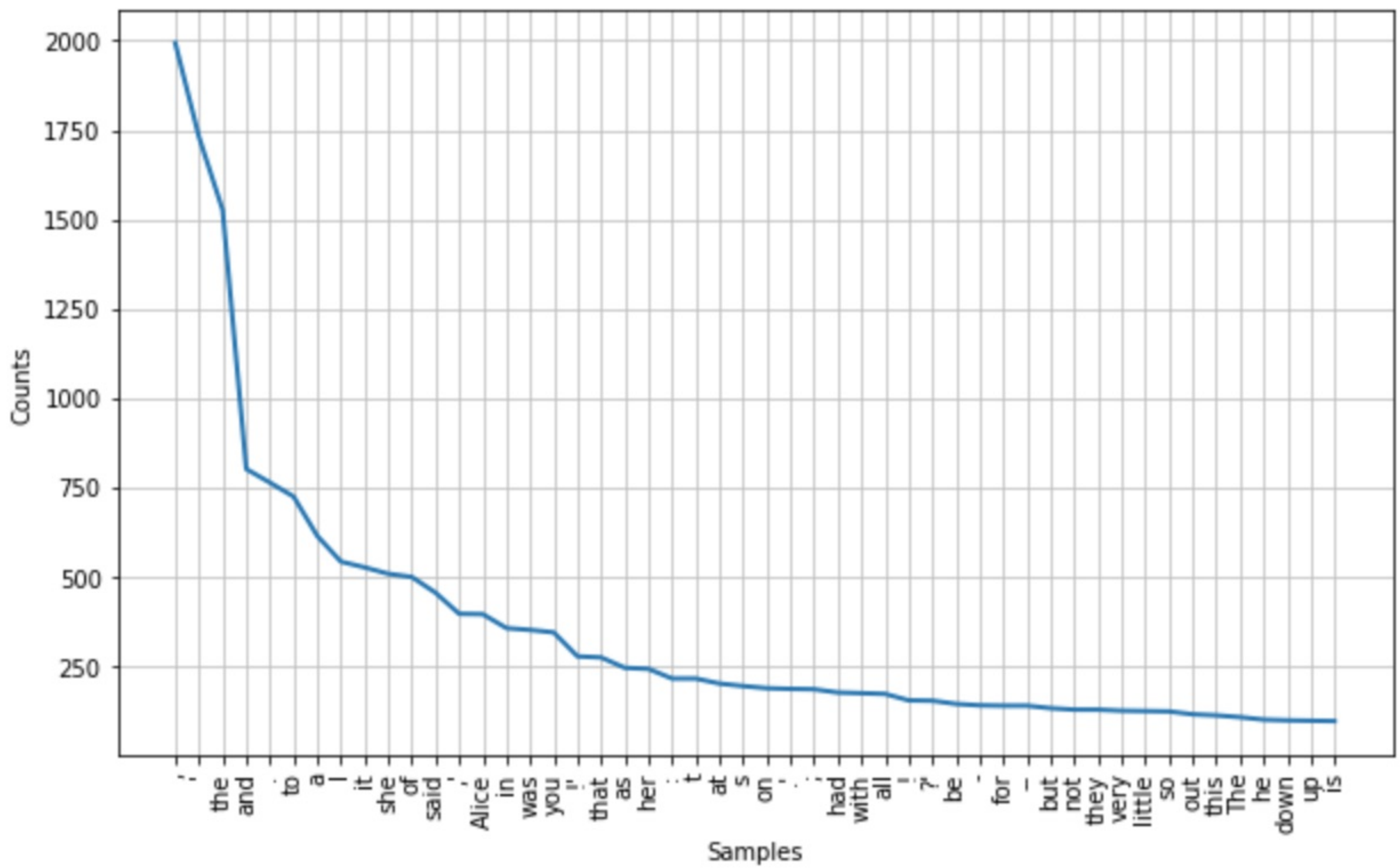
```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
alice.dispersion_plot(["Alice", "Rabbit", "Hatter", "Queen"])
```



https://tinyurl.com/imtkupython101

```
fdist = nltk.FreqDist(alice)
fdist.plot(50)
```

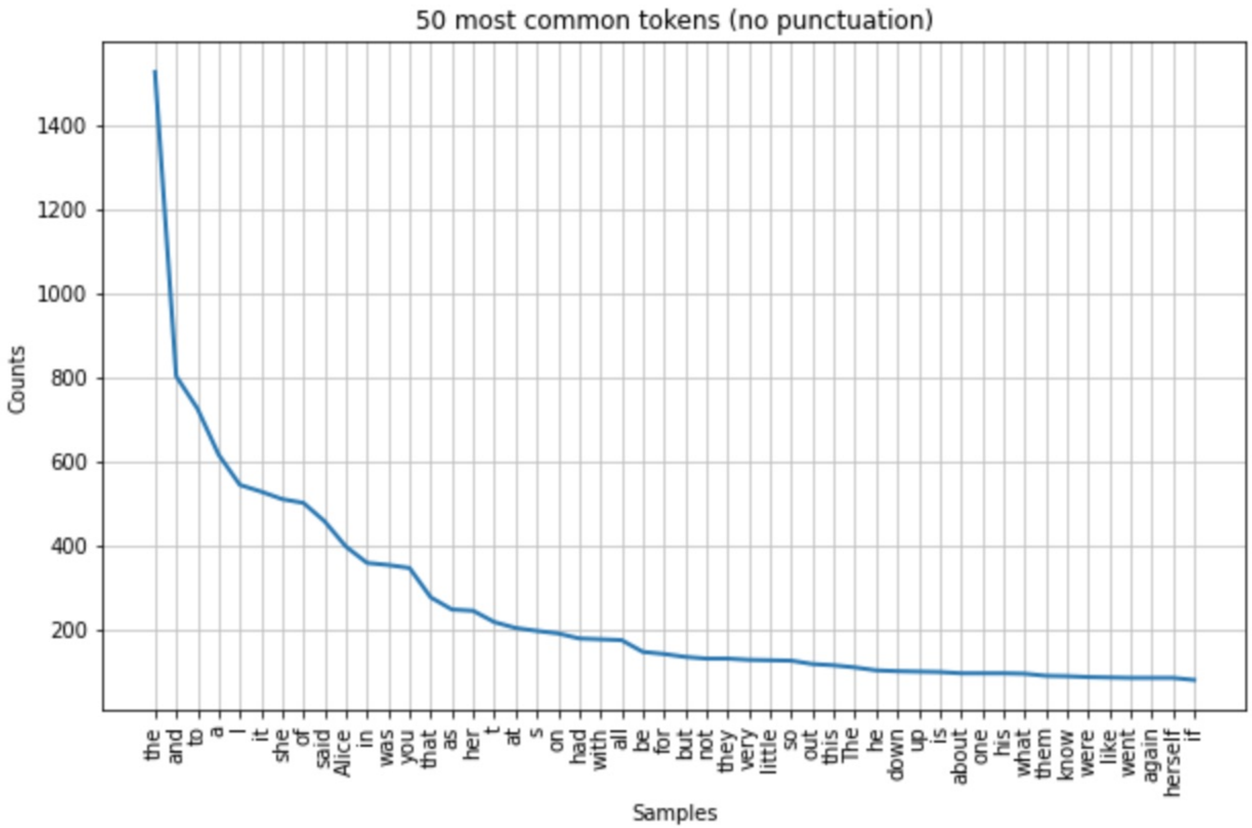
```
#import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
fdist = nltk.FreqDist(alice)
fdist.plot(50)
```



https://tinyurl.com/imtkupython101

```
for word, freq in fdist.items()
if word.isalpha()
```

```
#import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
fdist_no_punc = nltk.FreqDist(dict((word, freq) for word, freq in fdist.items() if word.isalpha()))
fdist_no_punc.plot(50, cumulative=False, title="50 most common tokens (no punctuation)")
```



https://tinyurl.com/imtkupython101

```python
nltk.download('stopwords')
stopwords = nltk.corpus.stopwords.words('english')
```

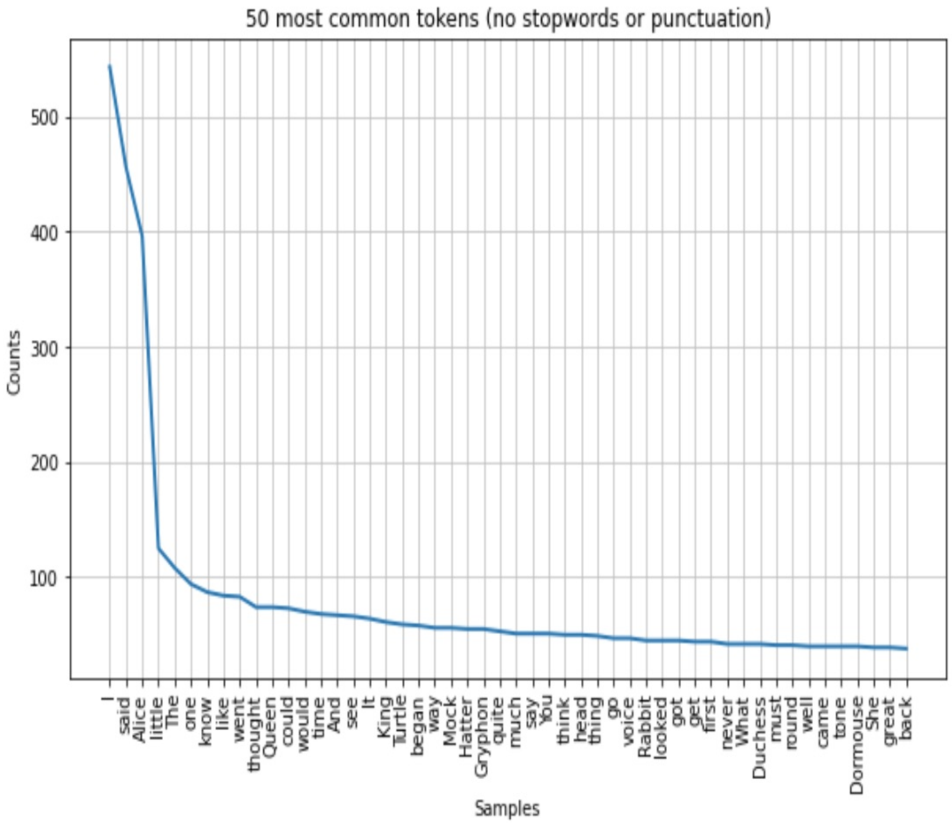
```python
import nltk
nltk.download('stopwords')
stopwords = nltk.corpus.stopwords.words('english')
stopwords
```

```
'same',
'so',
'than',
'too',
'very',
's',
't',
'can',
'will',
'just',
'don',
"don't",
'should',
"should've",
'now',
```

https://tinyurl.com/imtkupython101

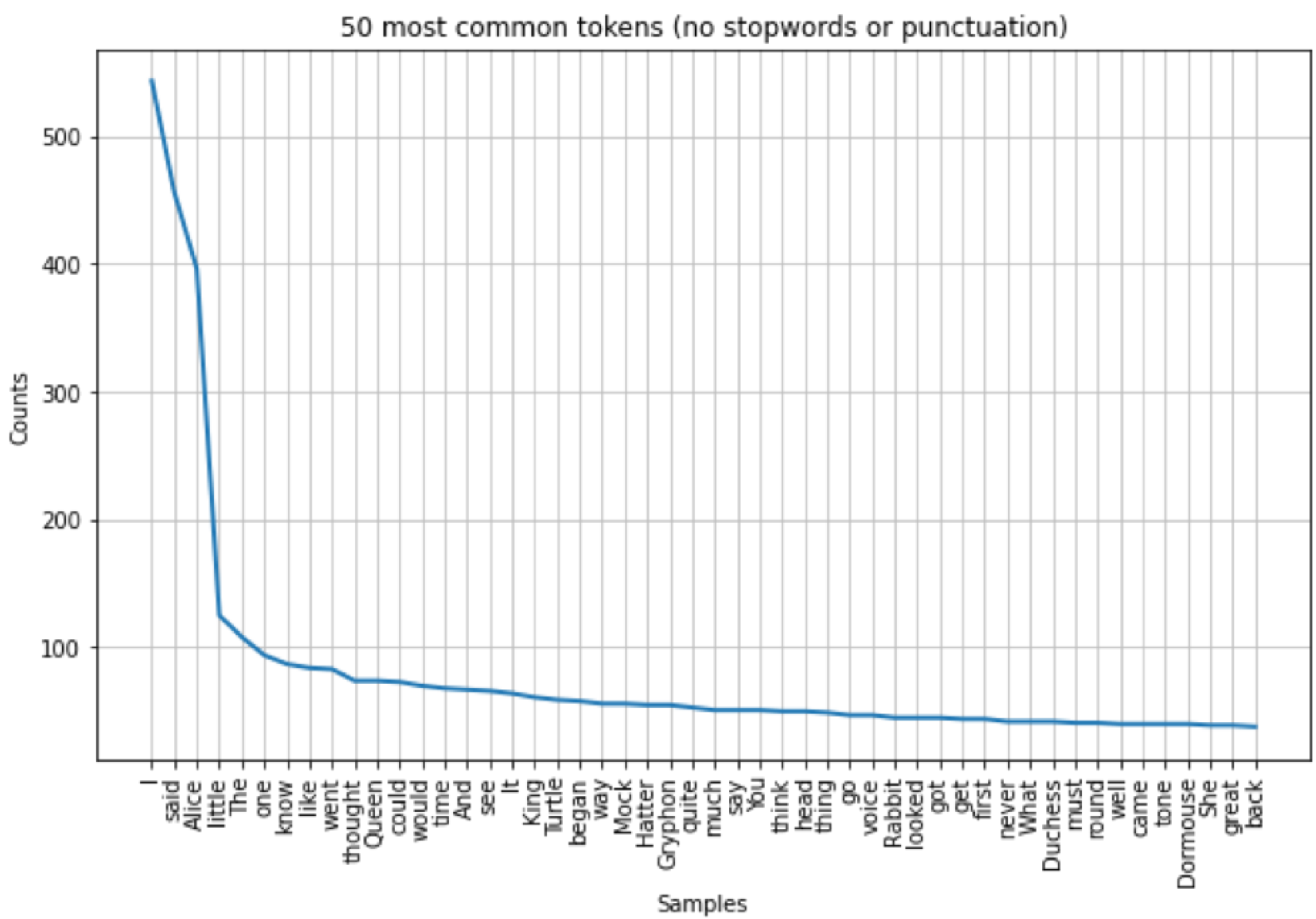
```
for word, freq in fdist.items()
if word not in stopwords and word.isalpha()
```

```
#import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
fdist_no_punc_no_stopwords = nltk.FreqDist(dict((word, freq) for word, freq in fdist.items() if word not in stopwords and word.isalpha()))
fdist_no_punc_no_stopwords.plot(50, cumulative=False, title="50 most common tokens (no stopwords or punctuation)")
```



https://tinyurl.com/imtkupython101

Alice Top 50 Tokens



https://tinyurl.com/imtkupython101

BeautifulSoup

```
import requests
from bs4 import BeautifulSoup

url = 'https://www.gutenberg.org/files/11/11-h/11-h.htm'
reqs = requests.get(url)
html_doc = reqs.text

soup = BeautifulSoup(html_doc, 'html.parser')
text = soup.get_text()
```

tensorflow.keras.preprocessing.text

```
from tensorflow.keras.preprocessing.text import Tokenizer

sentences = [
'i love my dog',
'I, love my cat',
'You love my dog!'
]

tokenizer = Tokenizer(num_words = 100)
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index
print('sentences:', sentences)
print('word index:', word_index)
```

```
sentences: ['i love my dog', 'I, love my cat', 'You love my dog!’]
word index: {'love': 1, 'my': 2, 'i': 3, 'dog': 4, 'cat': 5, 'you': 6}
```

https://tinyurl.com/imtkupython101

tensorflow.keras.preprocessing.sequence import pad_sequences

```
import tensorflow as tf
from tensorflow import keras

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

sentences = [
'I love my dog',
'I love my cat',
'You love my dog!',
'Do you think my dog is amazing?’
]

tokenizer = Tokenizer(num_words = 100, oov_token="<OOV>")
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index
sequences = tokenizer.texts_to_sequences(sentences)
padded = pad_sequences(sequences, maxlen=5)
print("sentences = ", sentences)
print("Word Index = " , word_index)
print("Sequences = " , sequences)
print("Padded Sequences:")
print(padded)
```

https://tinyurl.com/imtkupython101

```
tensorflow.keras.preprocessing.sequence
import pad_sequences
```

```
sentences = ['I love my dog', 'I love my
cat', 'You love my dog!', 'Do you think my
dog is amazing?’]

Word Index = {'<OOV>': 1, 'my': 2, 'love': 3,
'dog': 4, 'i': 5, 'you': 6, 'cat': 7, 'do':
8, 'think': 9, 'is': 10, 'amazing': 11}

Sequences = [[5, 3, 2, 4], [5, 3, 2, 7], [6,
3, 2, 4], [8, 6, 9, 2, 4, 10, 11]]

Padded Sequences: [[ 0 5 3 2 4] [ 0 5 3 2 7]
[ 0 6 3 2 4] [ 9 2 4 10 11]]
```

https://tinyurl.com/imtkupython101

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.]]
```

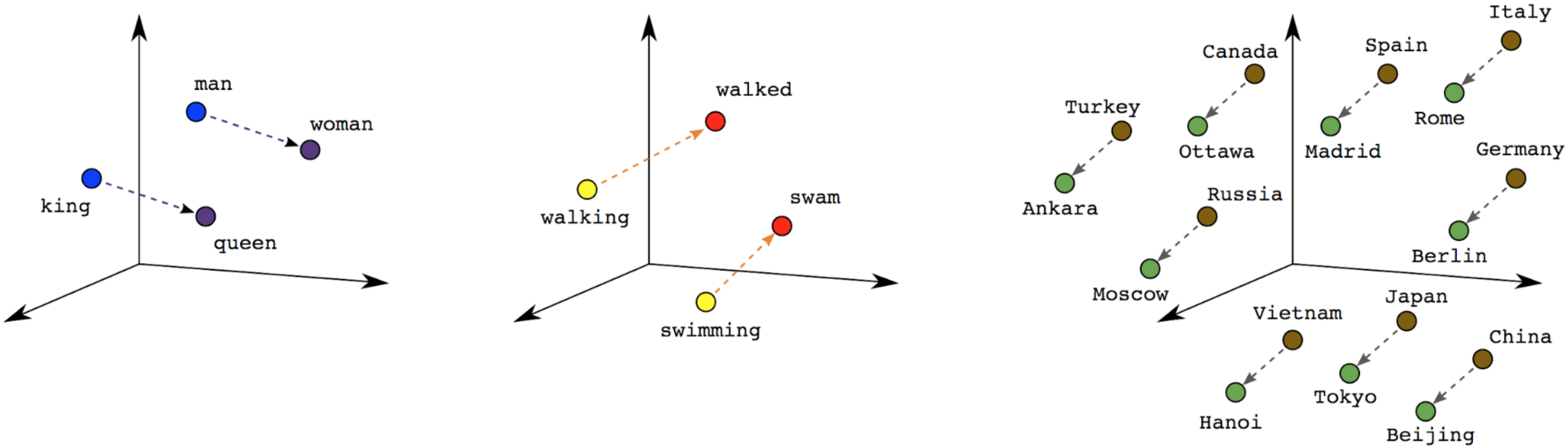
https://tinyurl.com/imtkupython101

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

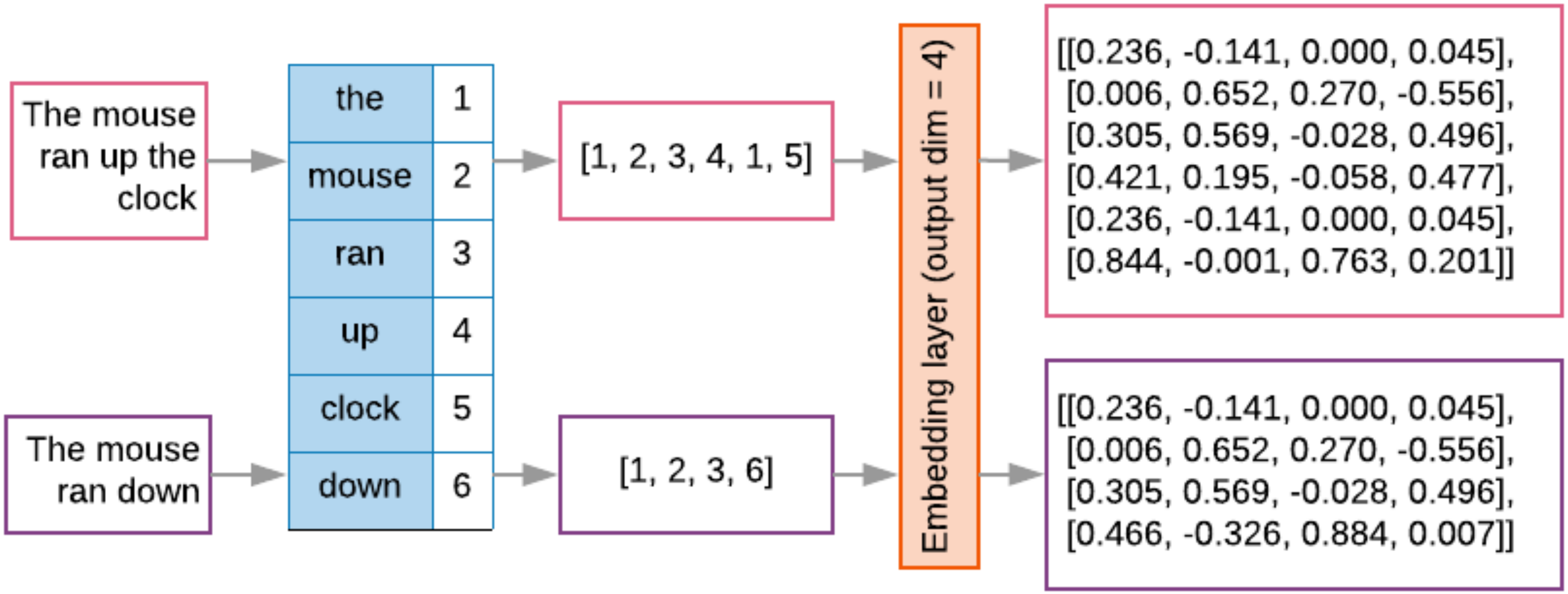
Word embeddings



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

```
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/

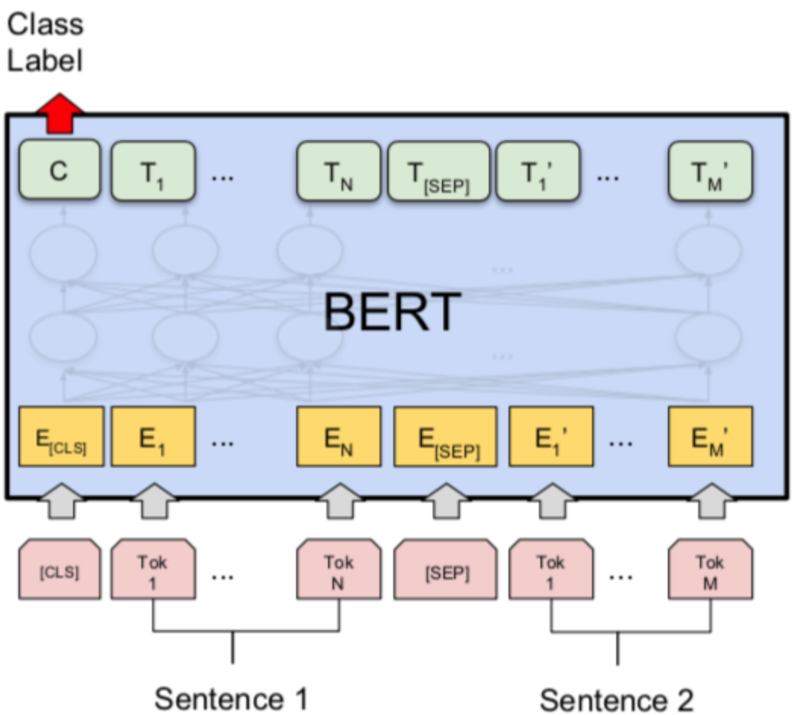
t.texts_to_matrix(docs, mode='tfidf')

```
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)
```

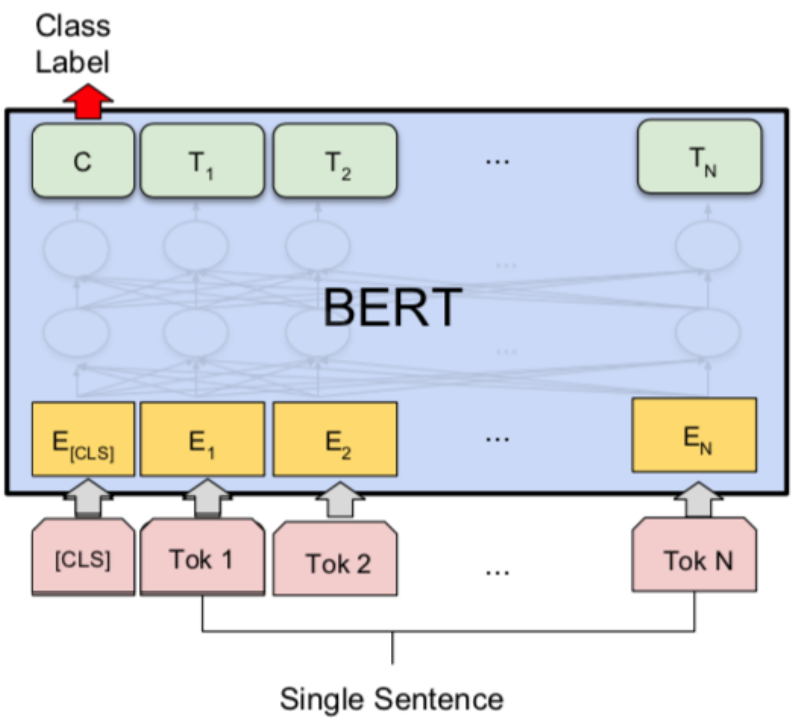
```
texts_to_matrix:
[[0. 0. 1.25276297 1.25276297 0. 0. 0. 0. 0. ]
[0. 0.98082925 0. 0. 1.25276297 0. 0. 0. 0. ]
[0. 0. 0. 0. 0. 1.25276297 1.25276297 0. 0. ]
[0. 0.98082925 0. 0. 0. 0. 0. 1.25276297 0. ]
[0. 0. 0. 0. 0. 0. 0. 0. 1.25276297]]
```

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

BERT Sequence-level tasks



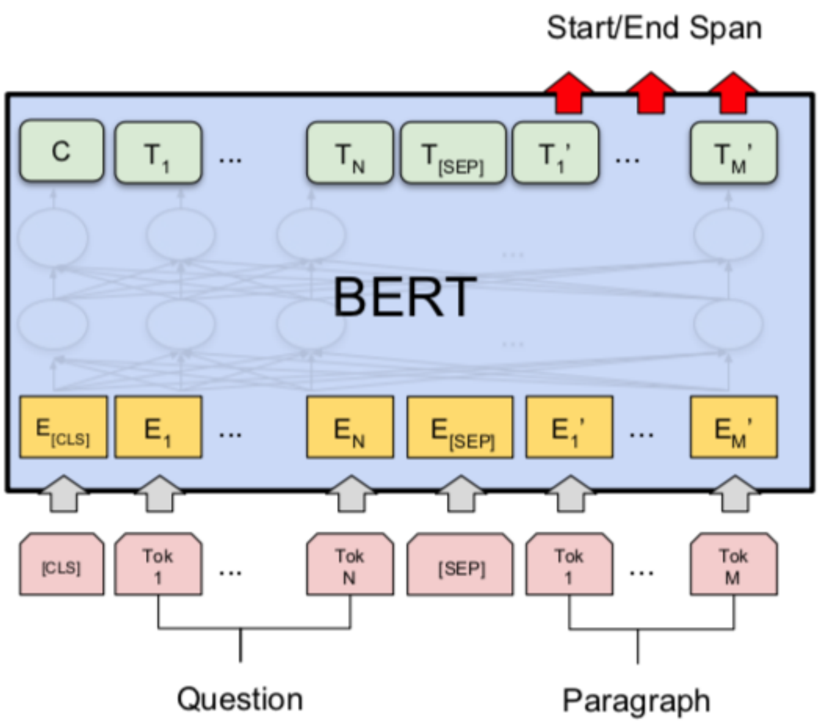
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



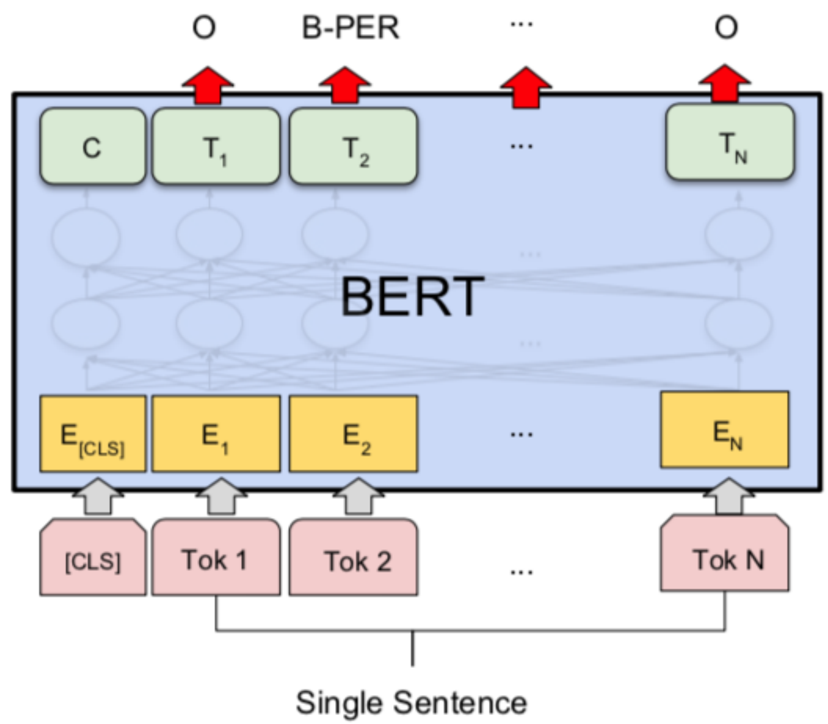
(b) Single Sentence Classification Tasks: SST-2, CoLA

Source: 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

BERT Token-level tasks



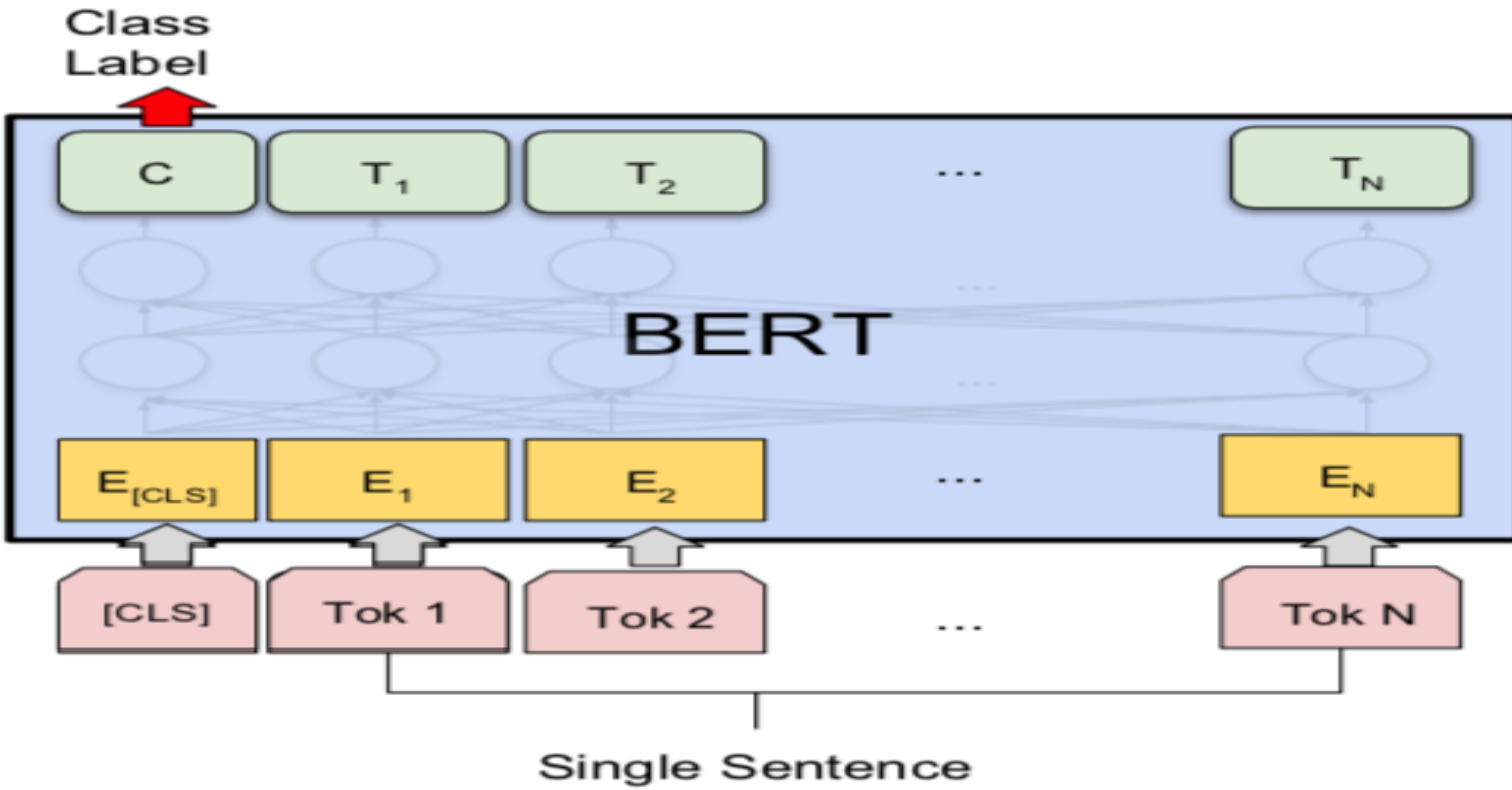
(c) Question Answering Tasks:
SQuAD v1.1



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

Source: 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

Sentiment Analysis: Single Sentence Classification

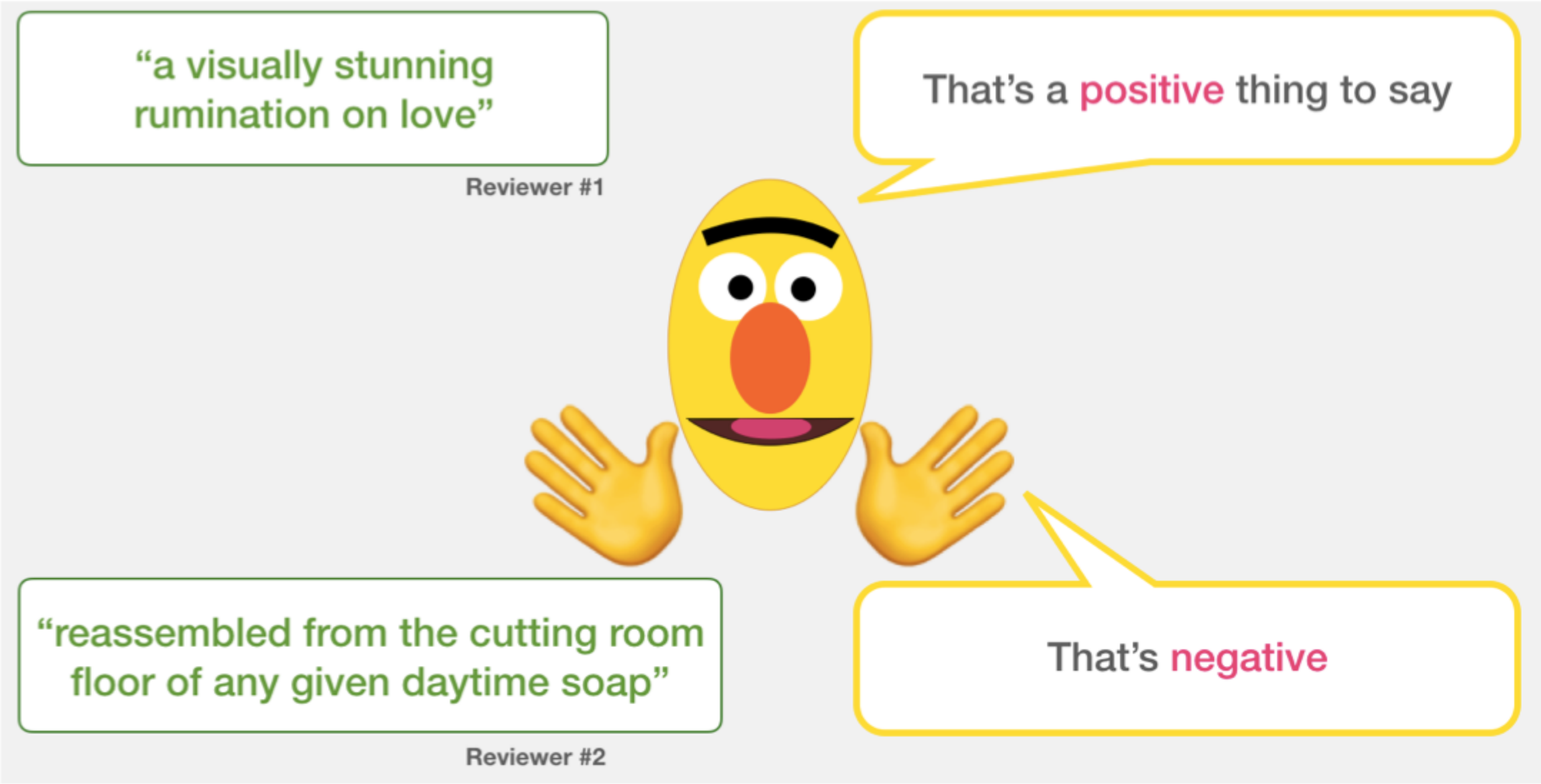


(b) Single Sentence Classification Tasks: SST-2, CoLA

Source: 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

A Visual Guide to Using BERT for the First Time

(Jay Alammar, 2019)



Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Sentiment Classification: SST2
Sentences from movie reviews

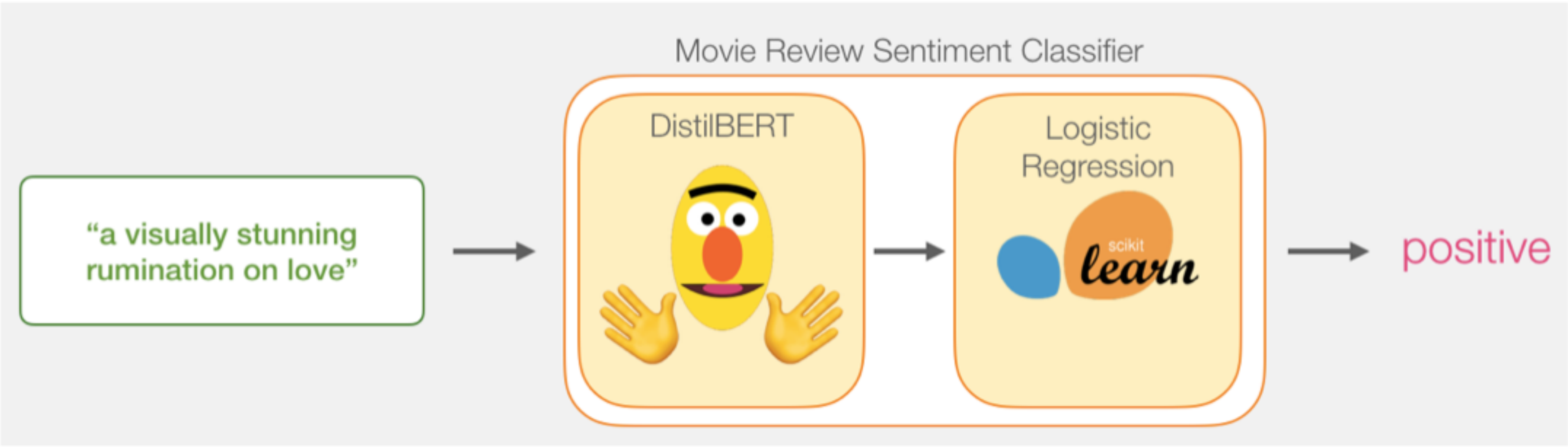
sentence	label
a stirring , funny and finally transporting re imagining of beauty and the beast and 1930s horror films	1
apparently reassembled from the cutting room floor of any given daytime soap	0
they presume their audience won't sit still for a sociology lesson	0
this is a visually stunning rumination on love , memory , history and the war between art and commerce	1
jonathan parker 's bartleby should have been the be all end all of the modern office anomie films	1

Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Movie Review Sentiment Classifier

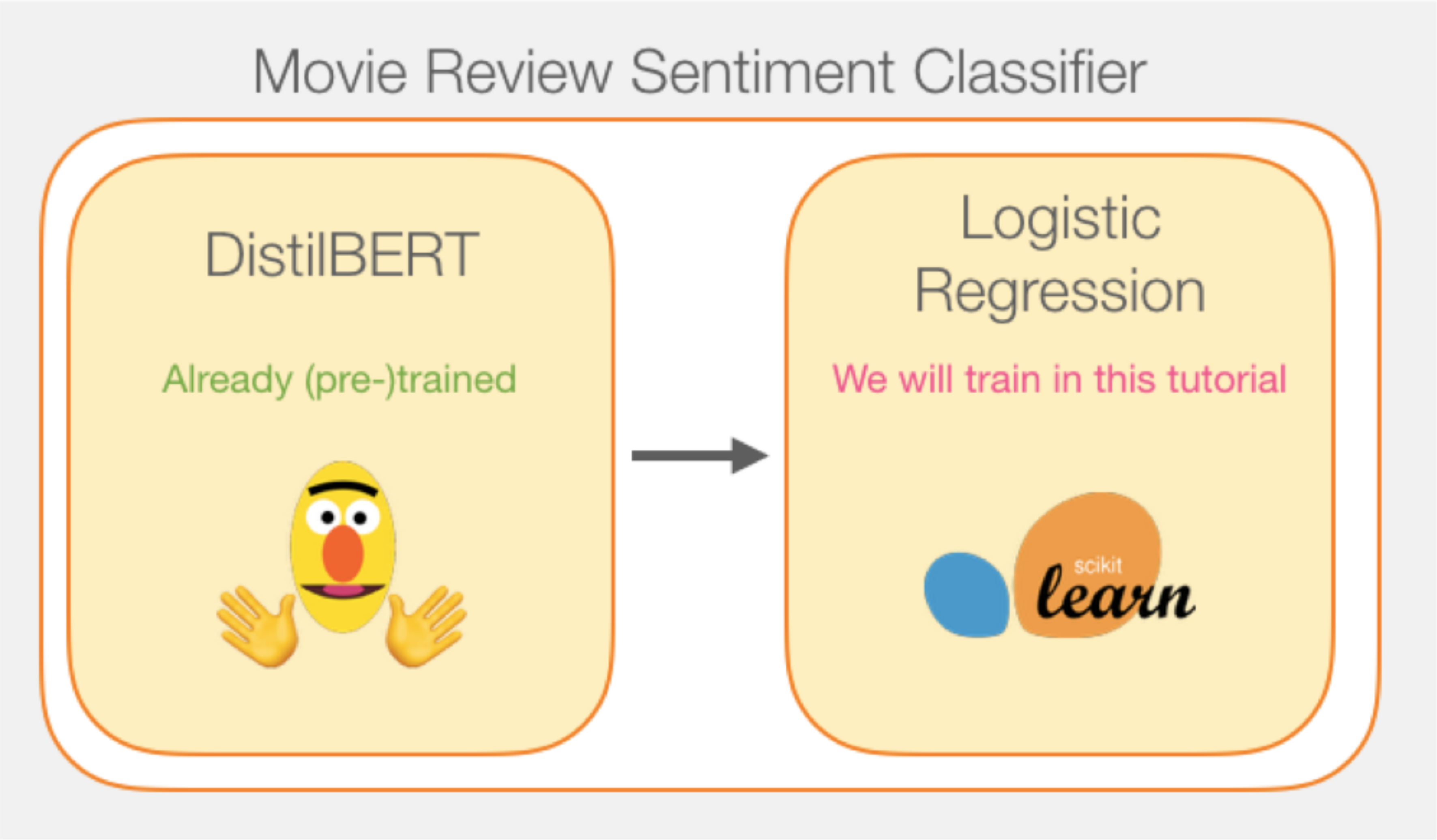
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Movie Review Sentiment Classifier



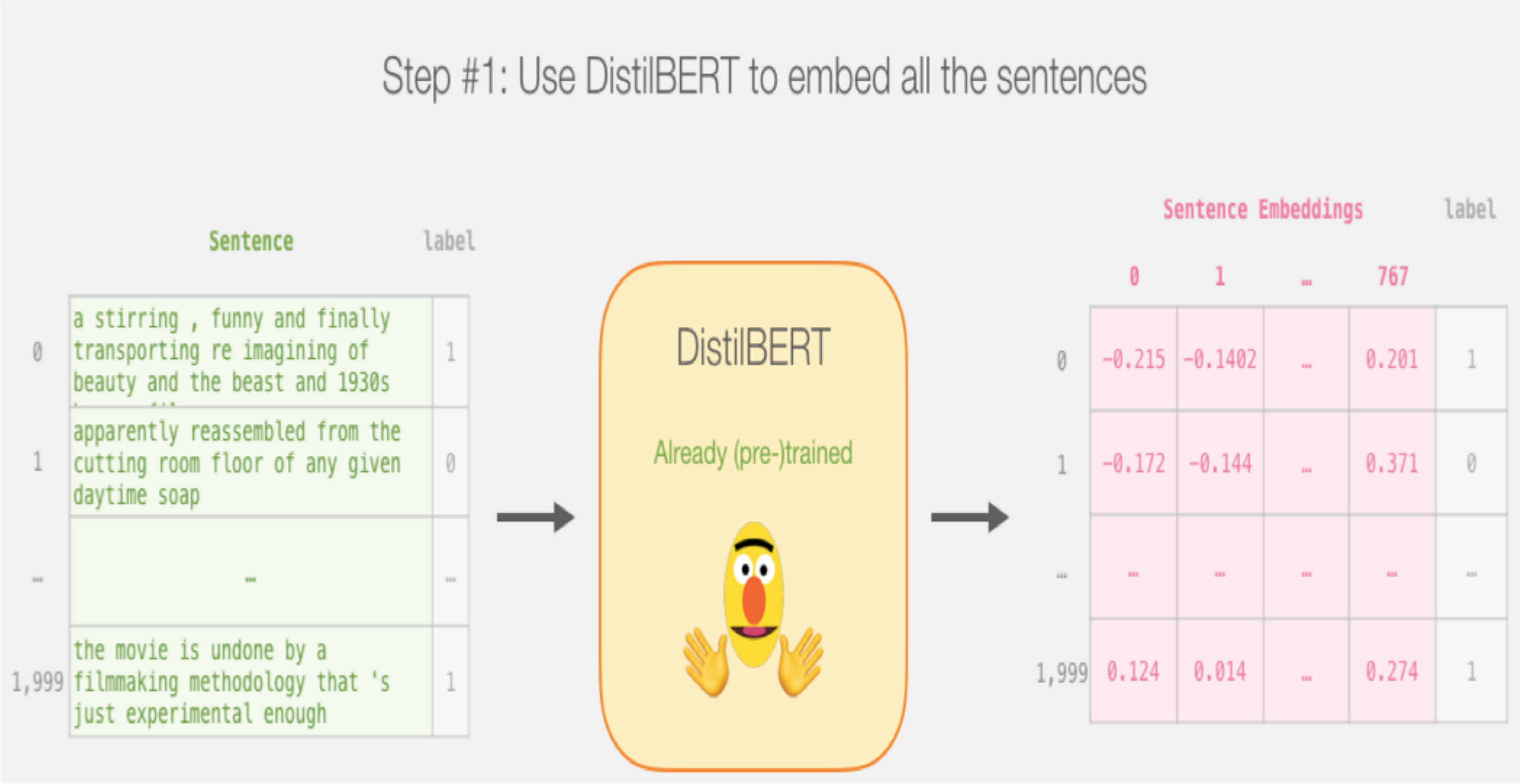
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Movie Review Sentiment Classifier Model Training



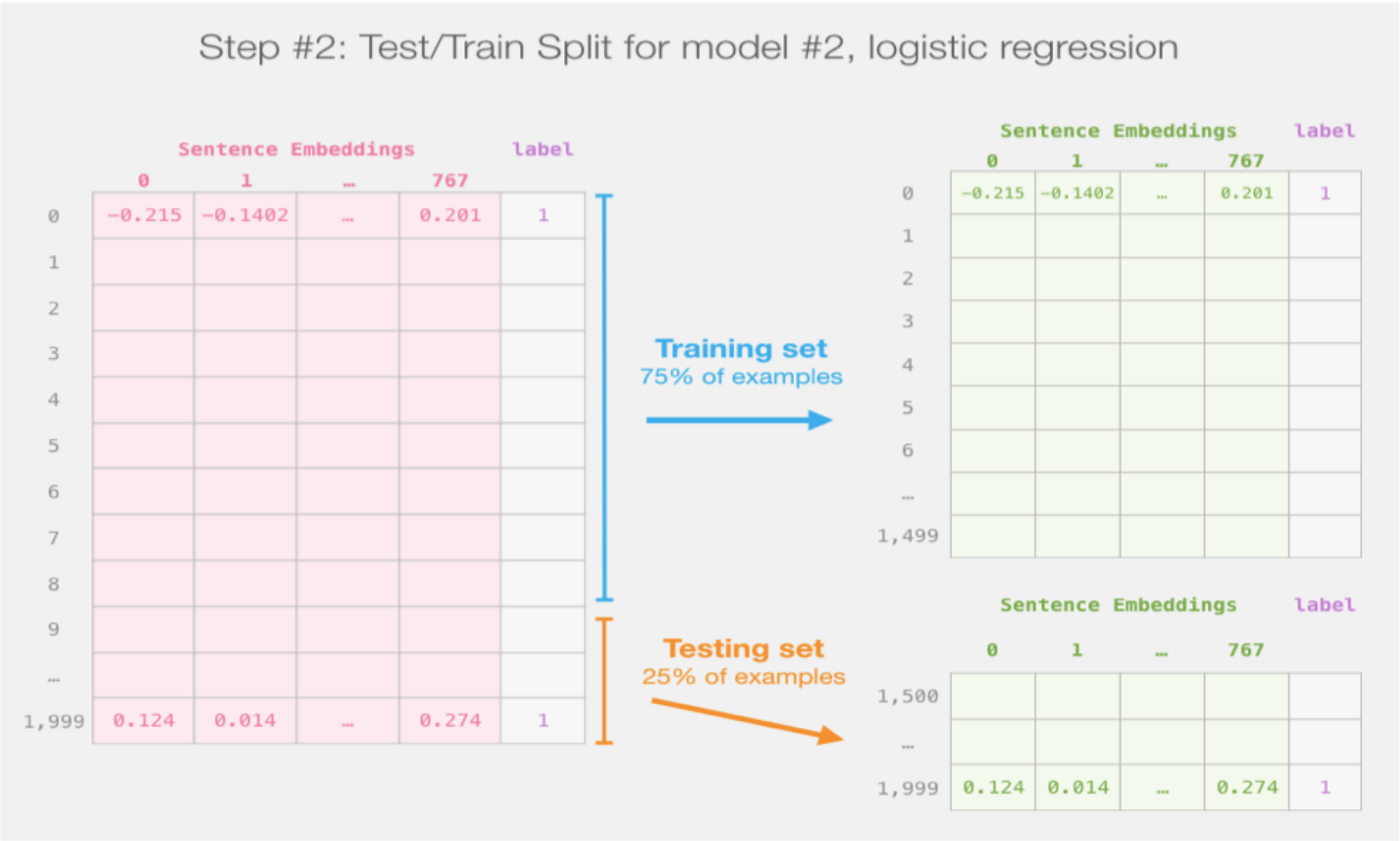
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Step # 1 Use distilBERT to Generate Sentence Embeddings



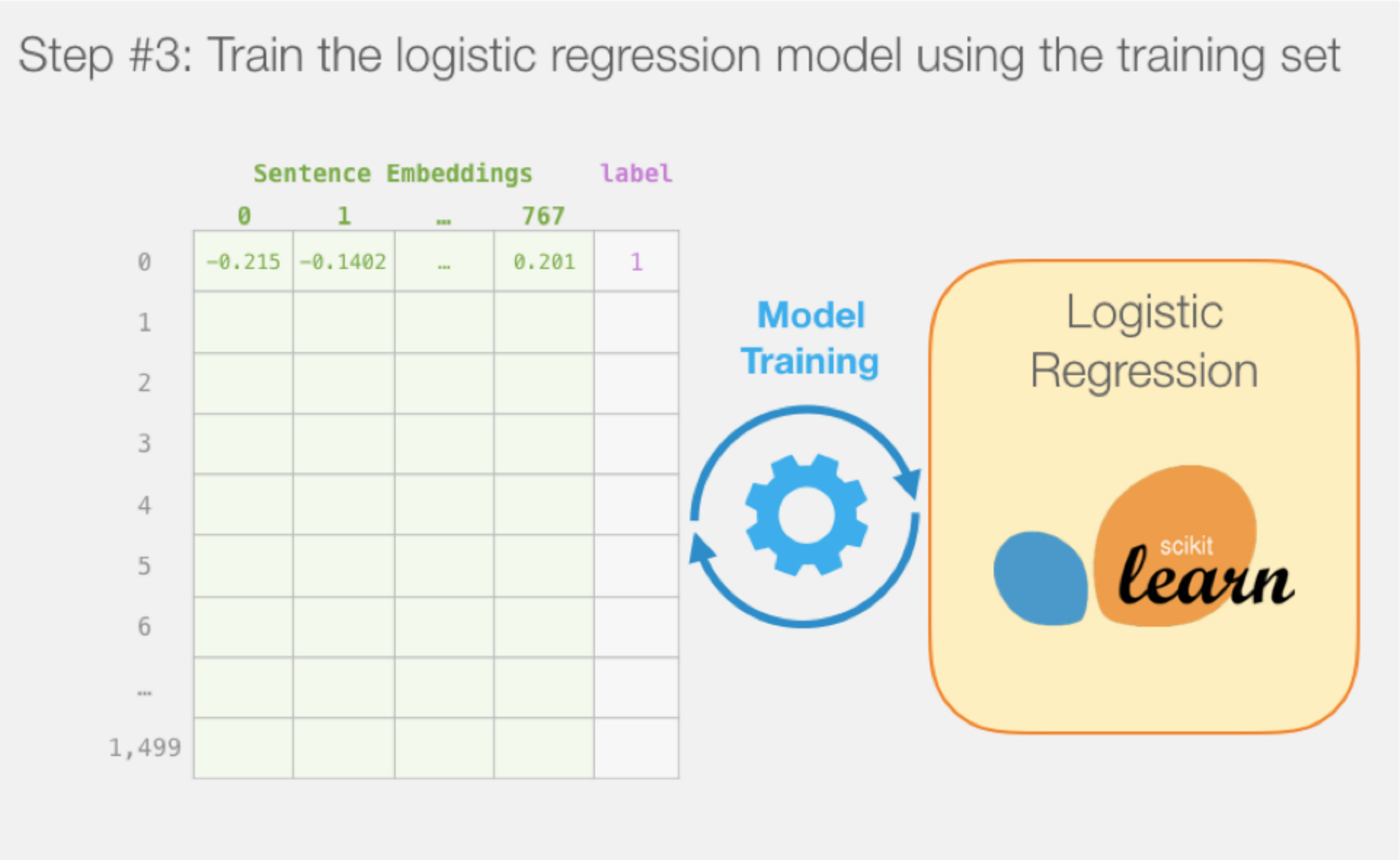
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Step #2:Test/Train Split for Model #2, Logistic Regression



Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Step #3 Train the logistic regression model using the training set

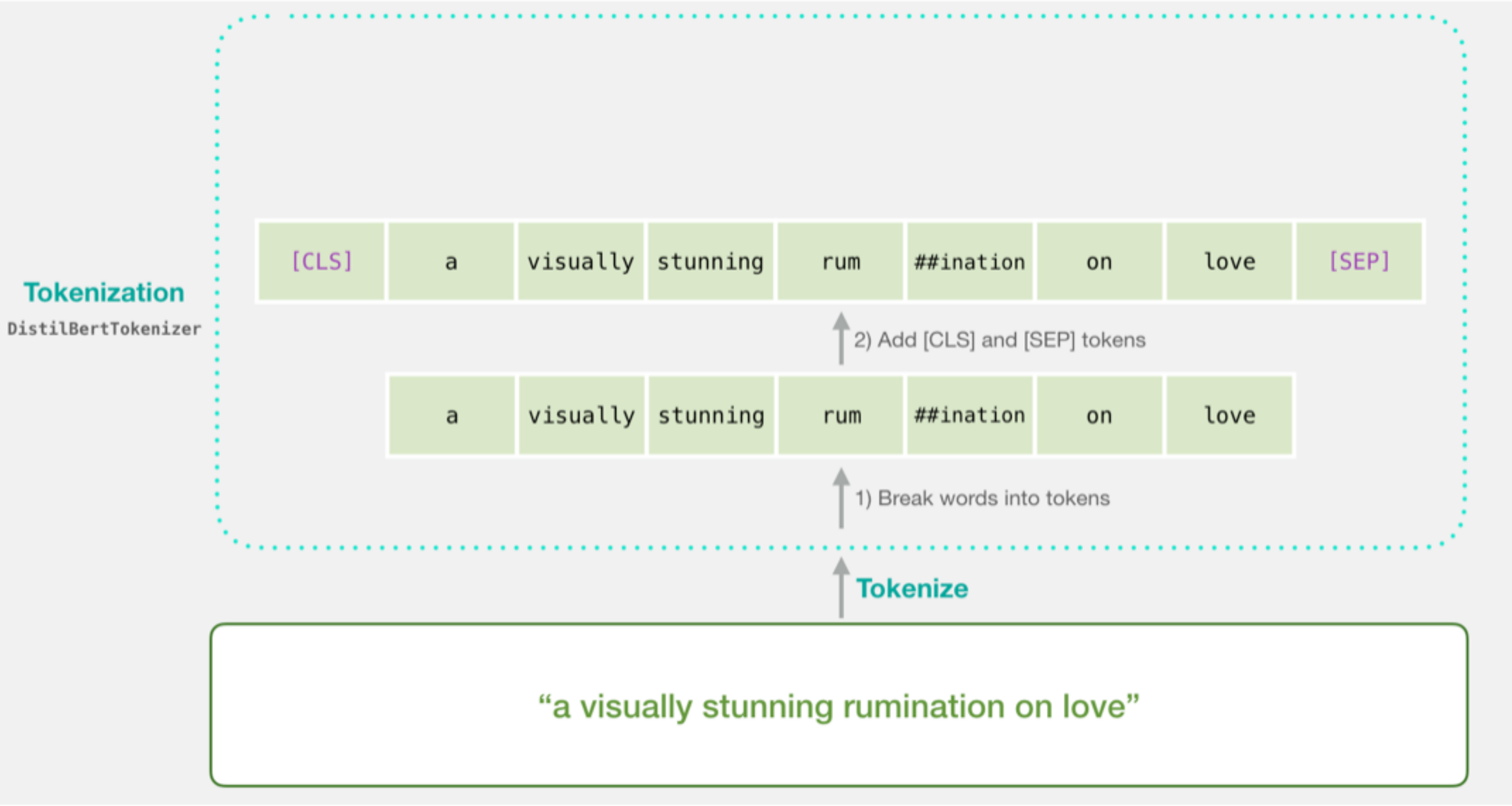


Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Tokenization

[CLS] a visually stunning rum ##ination on love [SEP]

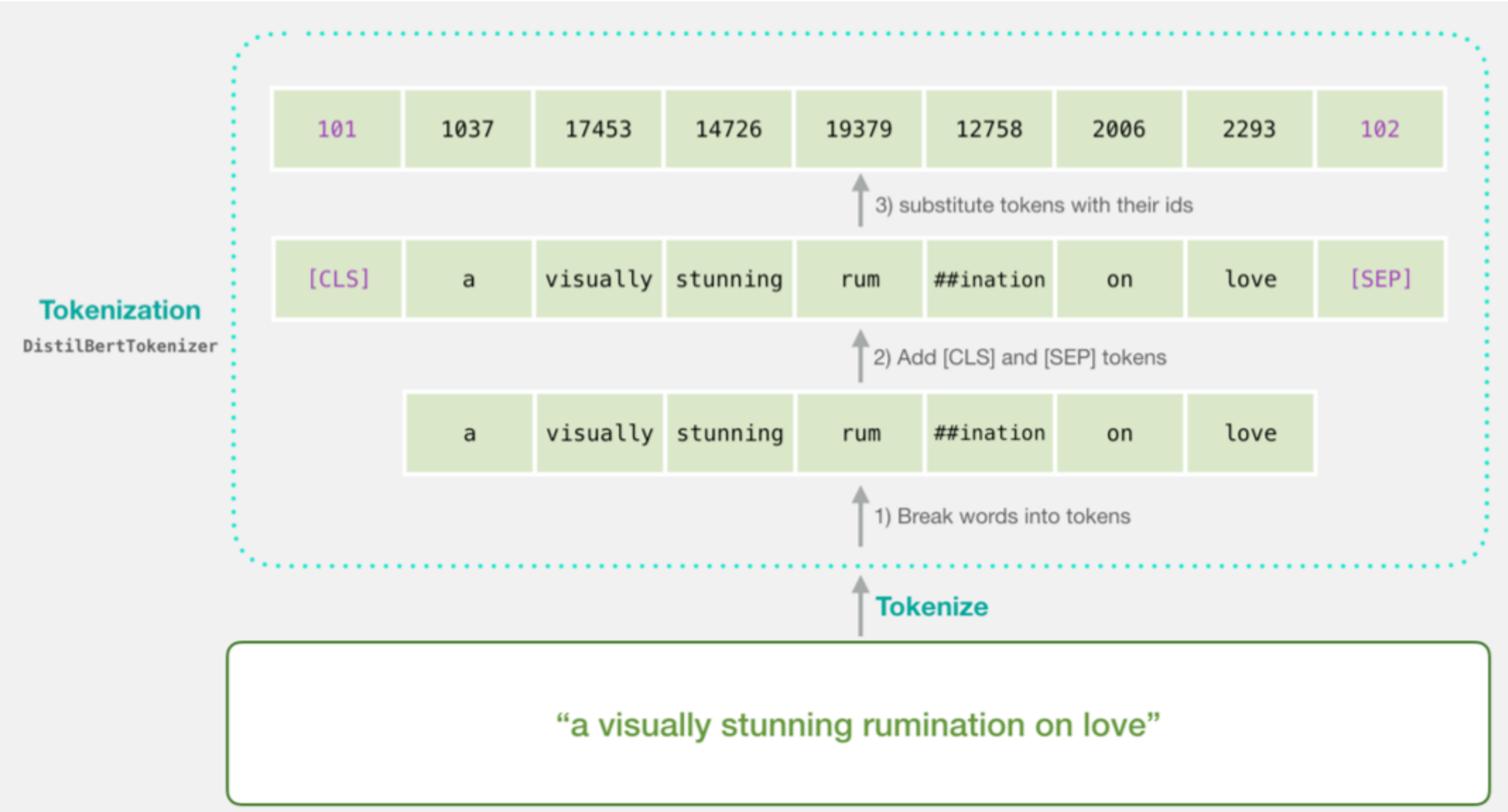
a visually stunning rumination on love



Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

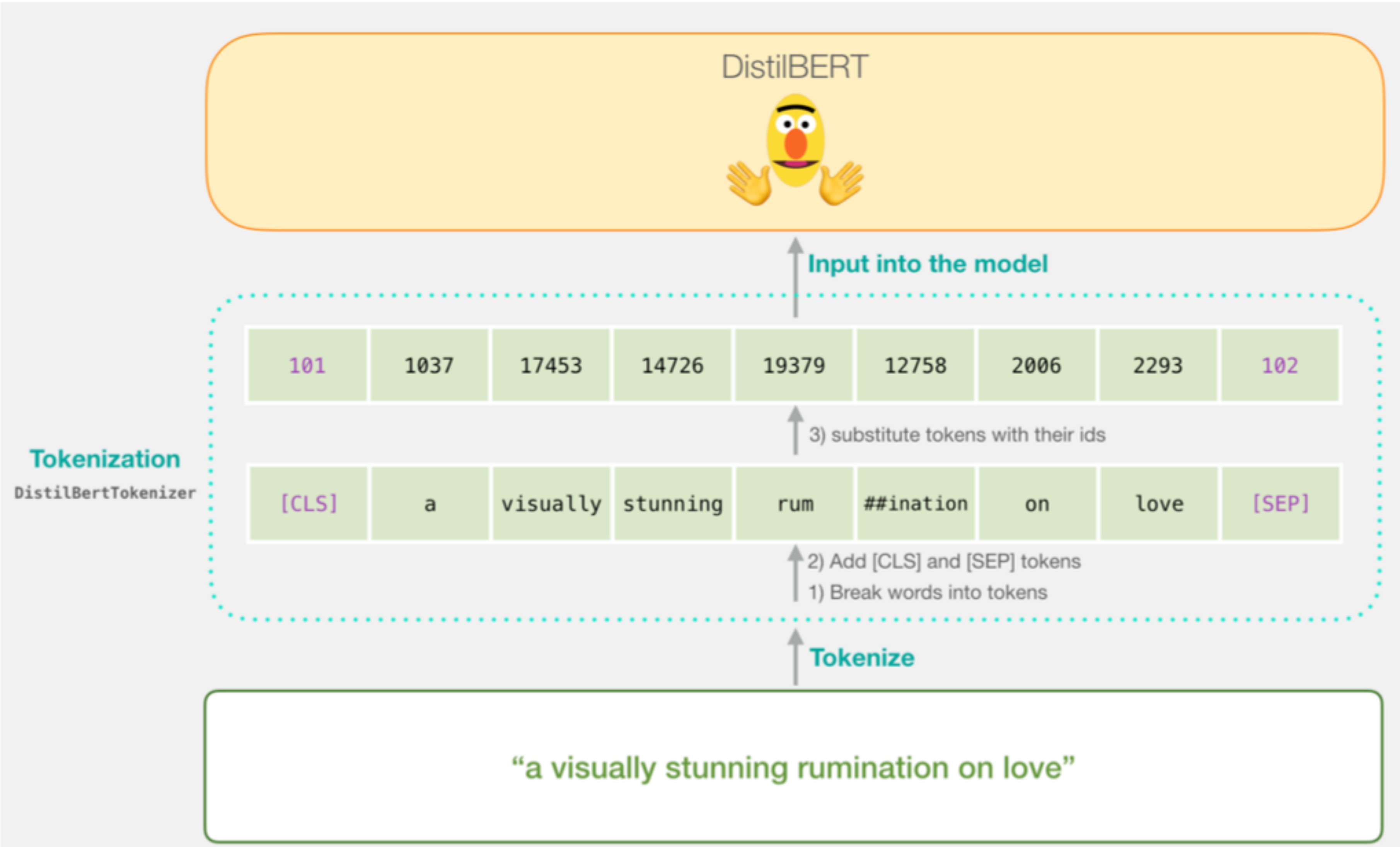
Tokenization

```
tokenizer.encode("a visually stunning rumination on love",
                 add_special_tokens=True)
```



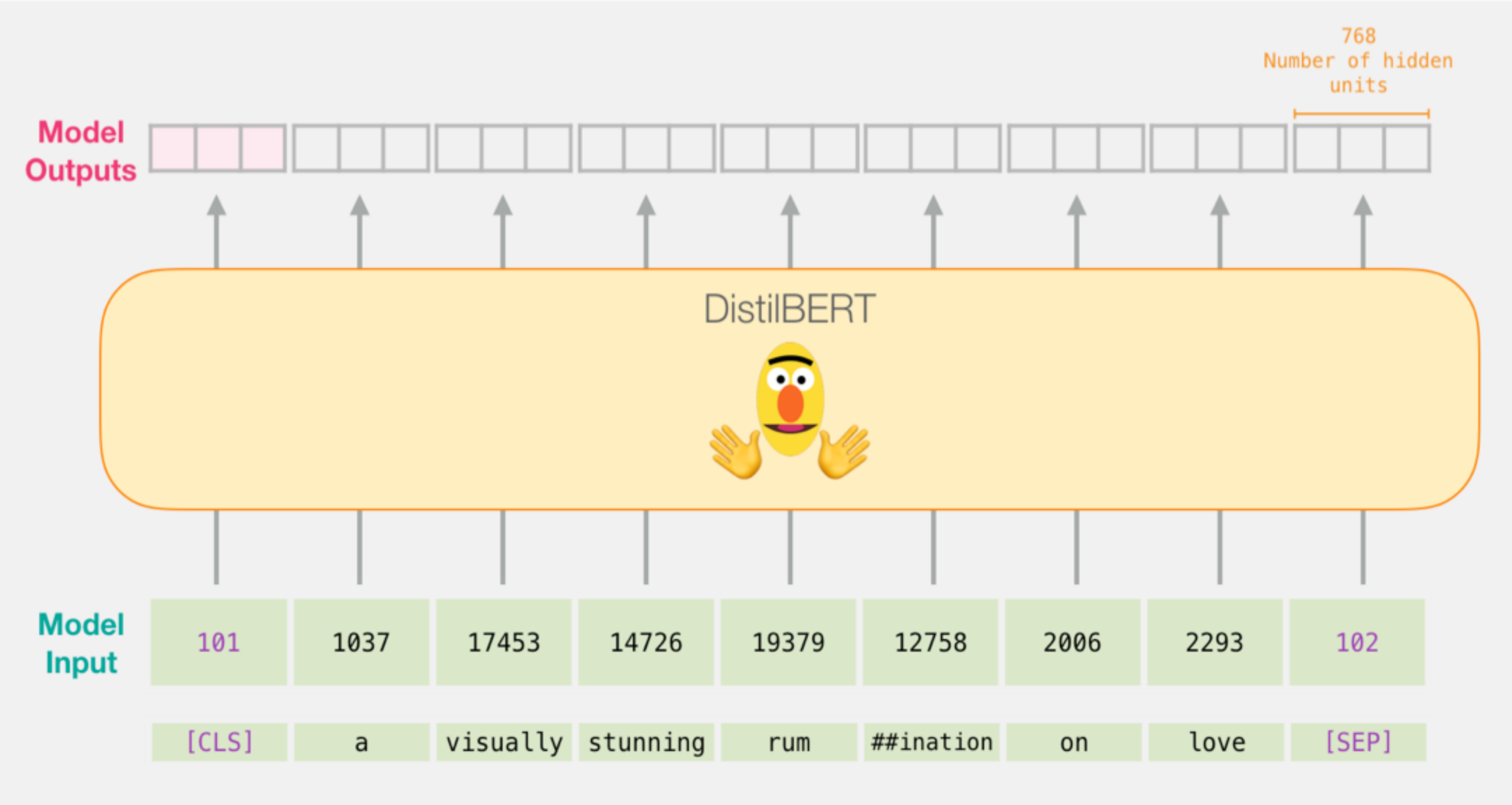
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Tokenization for BERT Model



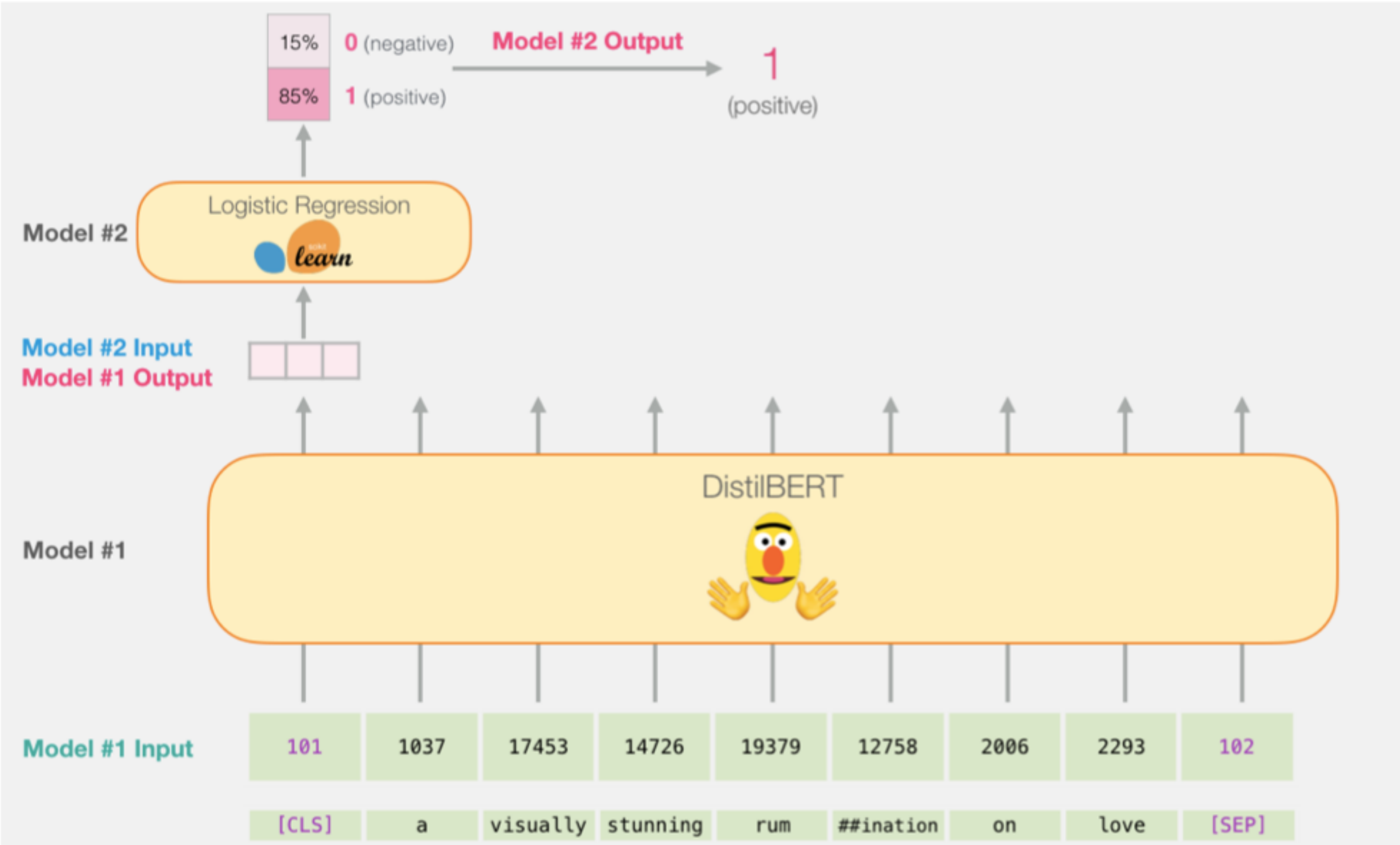
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Flowing Through DistilBERT (768 features)



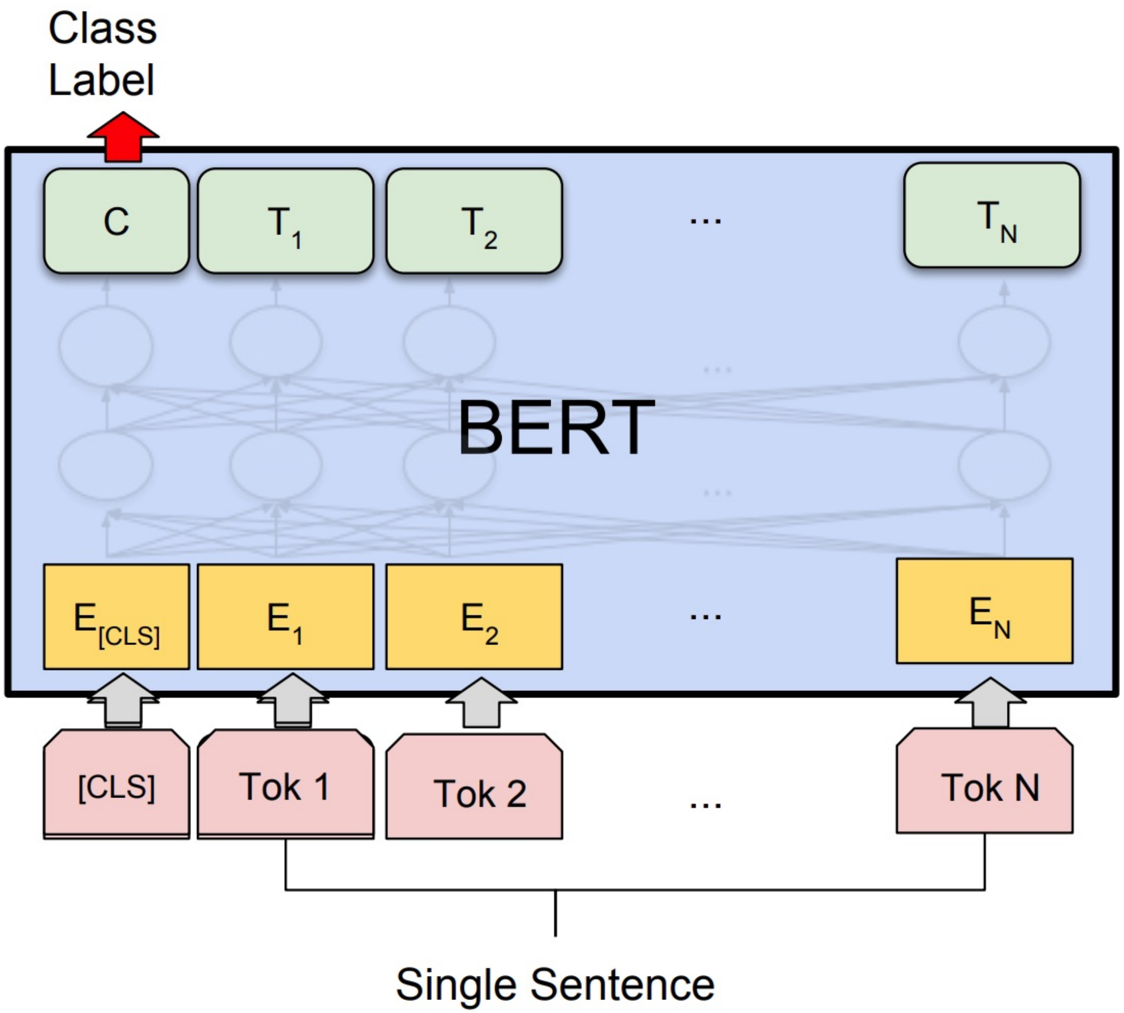
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Model #1 Output Class vector as Model #2 Input



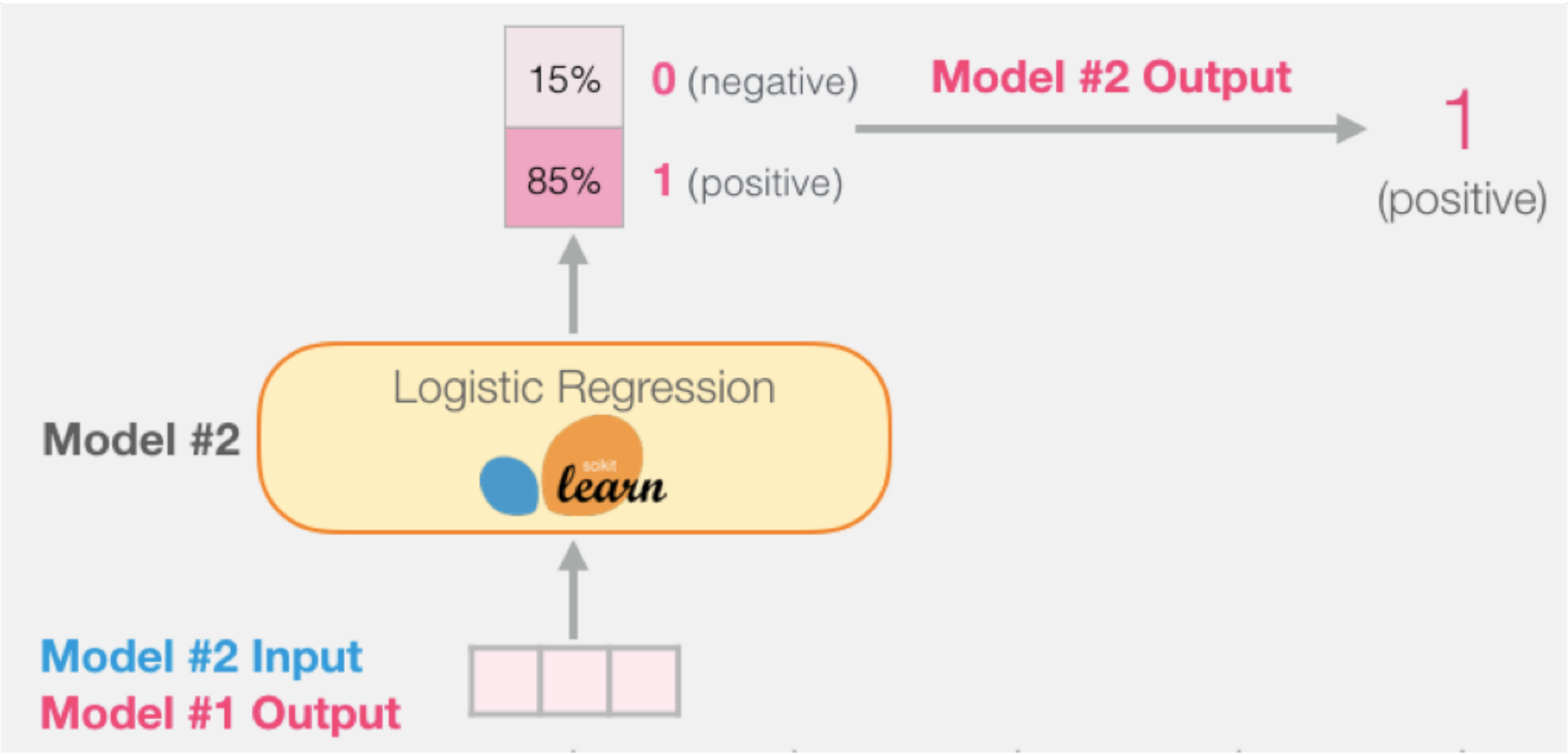
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Fine-tuning BERT on Single Sentence Classification Tasks



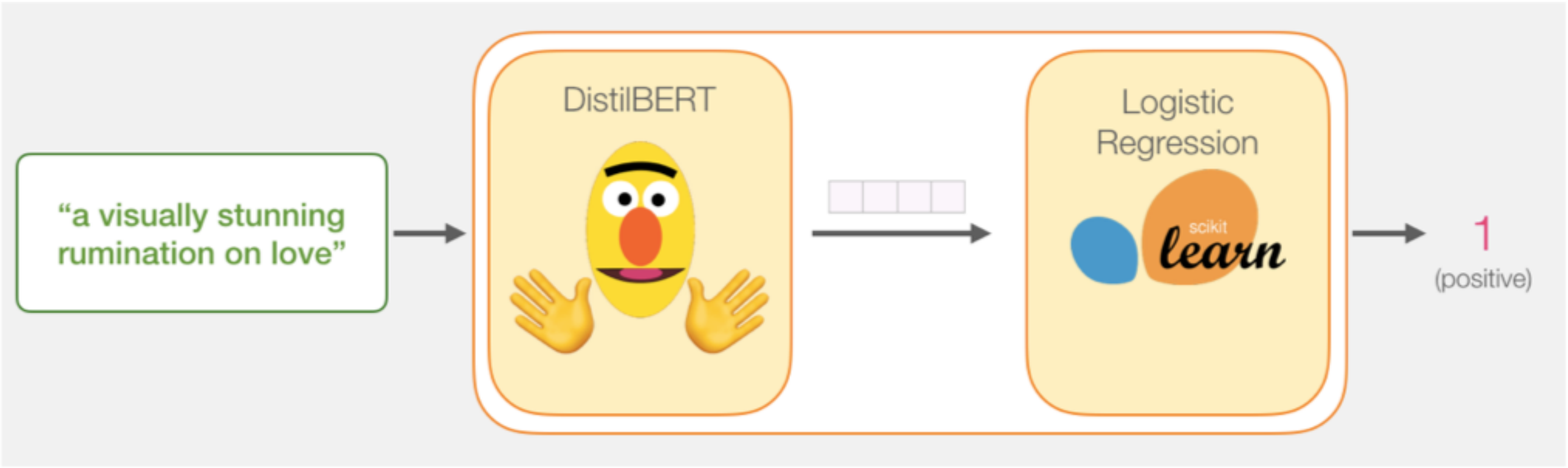
Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

Model #1 Output Class vector as Model #2 Input



Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Logistic Regression Model to classify Class vector



Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

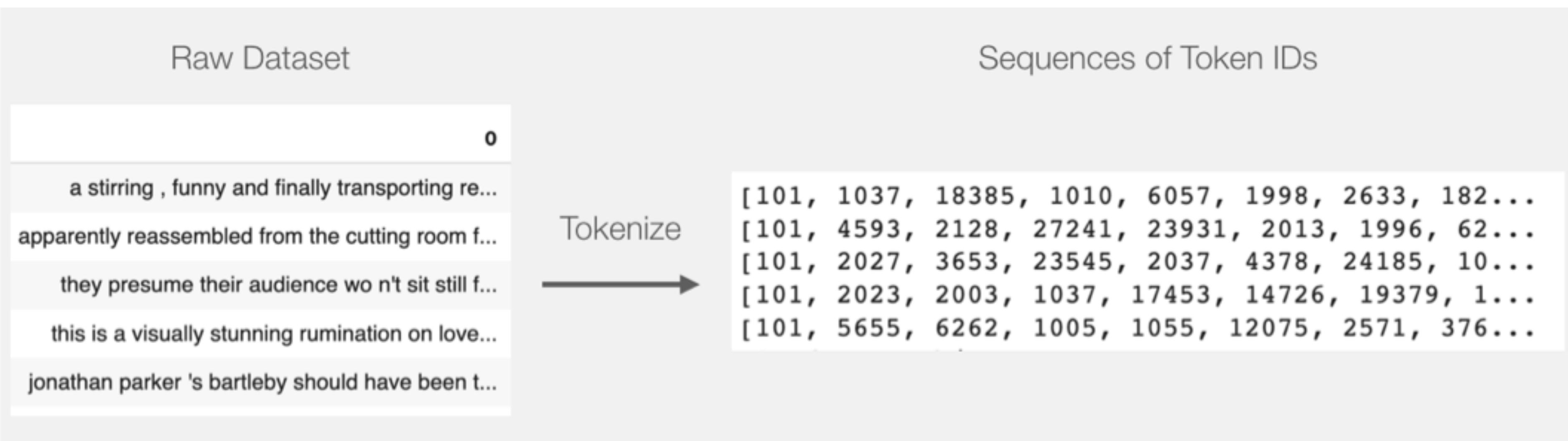
```python
df = pd.read_csv('https://github.com/clairett/pytorch-sentiment-classification/raw/master/data/SST2/train.tsv', delimiter='\t', header=None)

df.head()
```

	0	1
0	a stirring , funny and finally transporting re...	1
1	apparently reassembled from the cutting room f...	0
2	they presume their audience wo n't sit still f...	0
3	this is a visually stunning rumination on love...	1
4	jonathan parker 's bartleby should have been t...	1

Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Tokenization

```
tokenized = df[0].apply((lambda x: tokenizer.encode(x, add_special_tokens=True)))
```



Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

BERT Input Tensor

BERT/DistilBERT Input Tensor

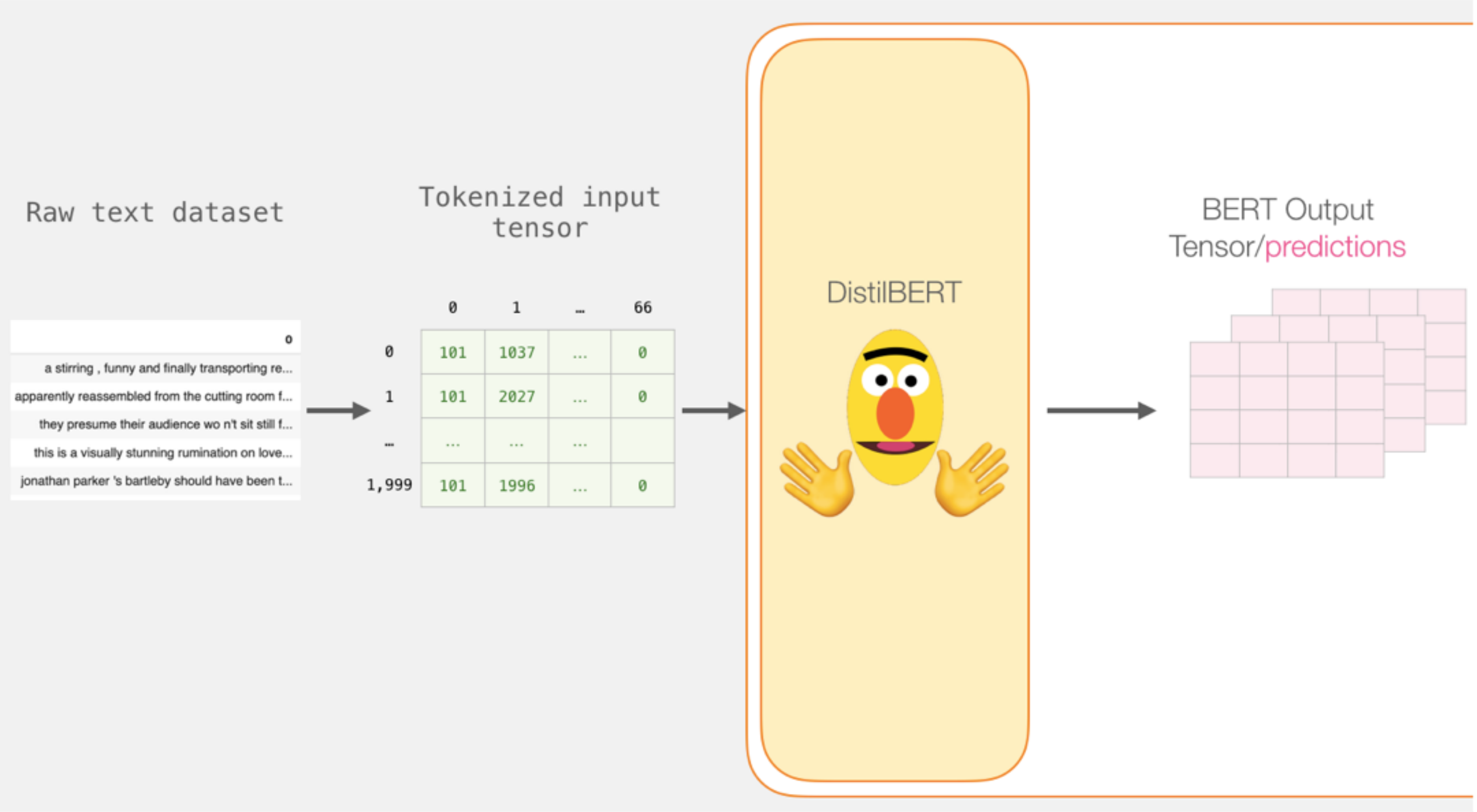
Tokens in each sequence

Input sequences (reviews)		0	1	…	66
	0	101	1037	...	0
	1	101	2027	...	0
	…	...	...	...	
	1,999	101	1996	...	0

Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

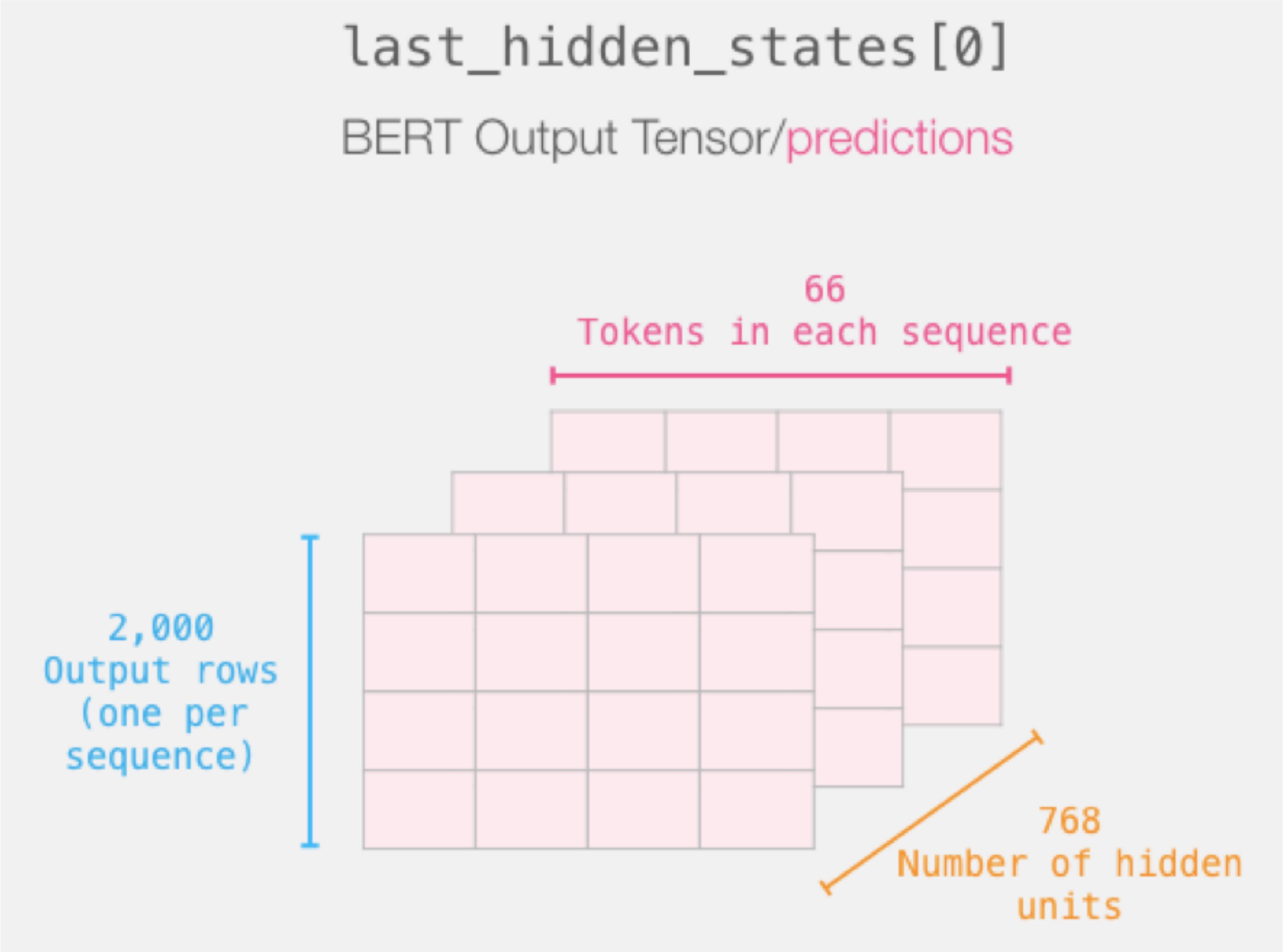
Processing with DistilBERT

```
input_ids = torch.tensor(np.array(padded))
last_hidden_states = model(input_ids)
```



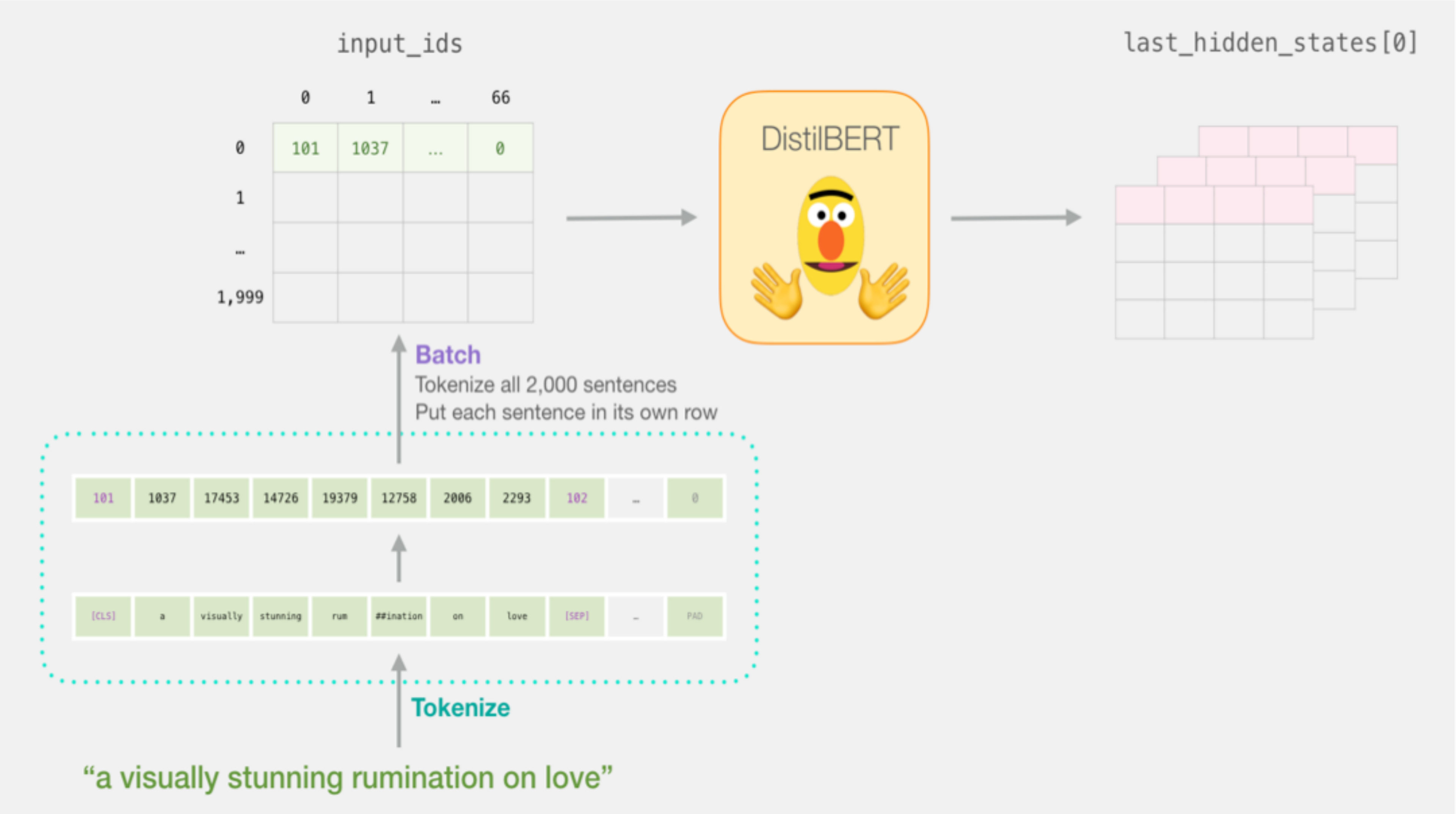
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Unpacking the BERT output tensor



Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

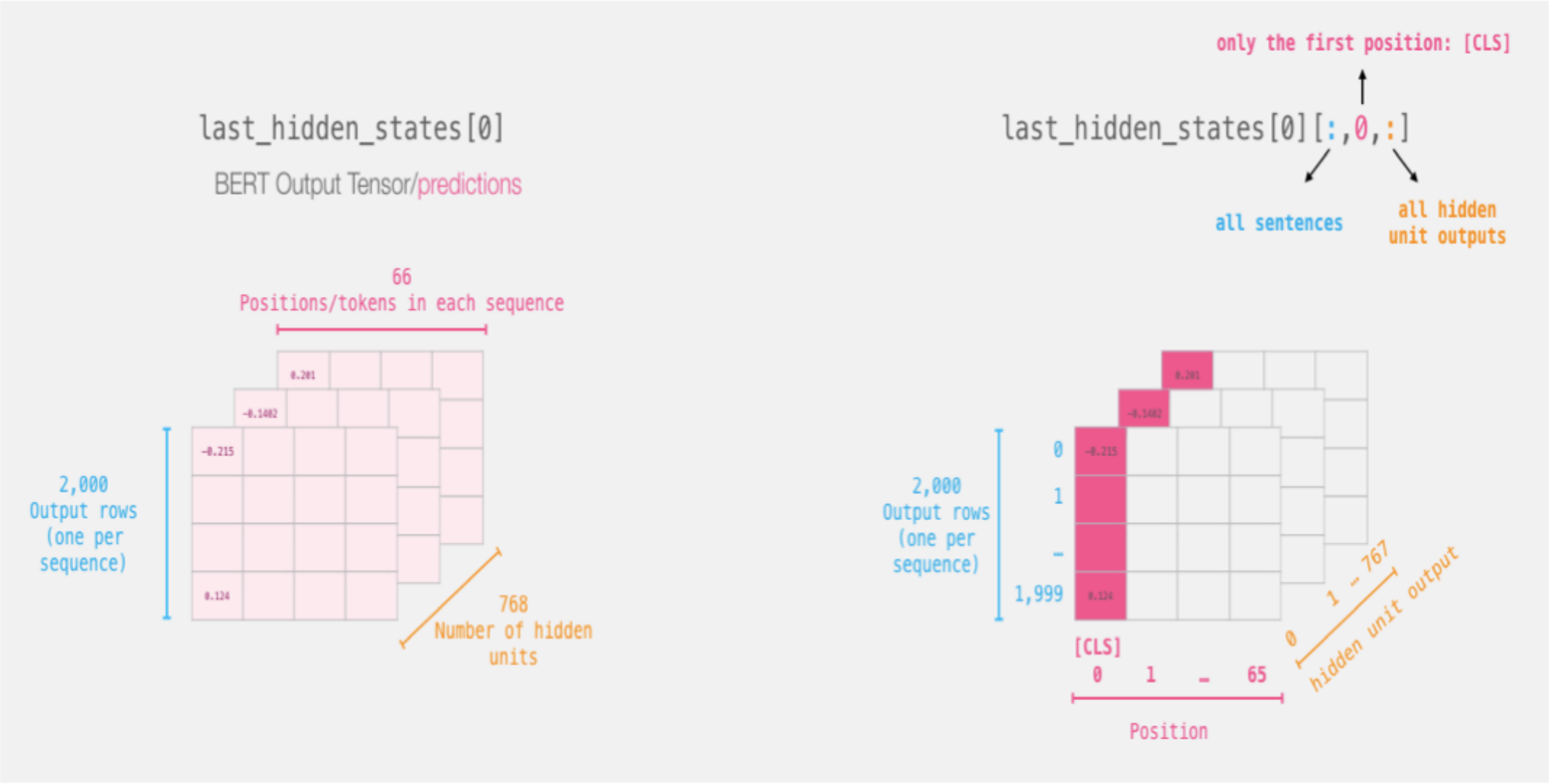
Sentence to last_hidden_state[0]



Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

BERT's output for the [CLS] tokens

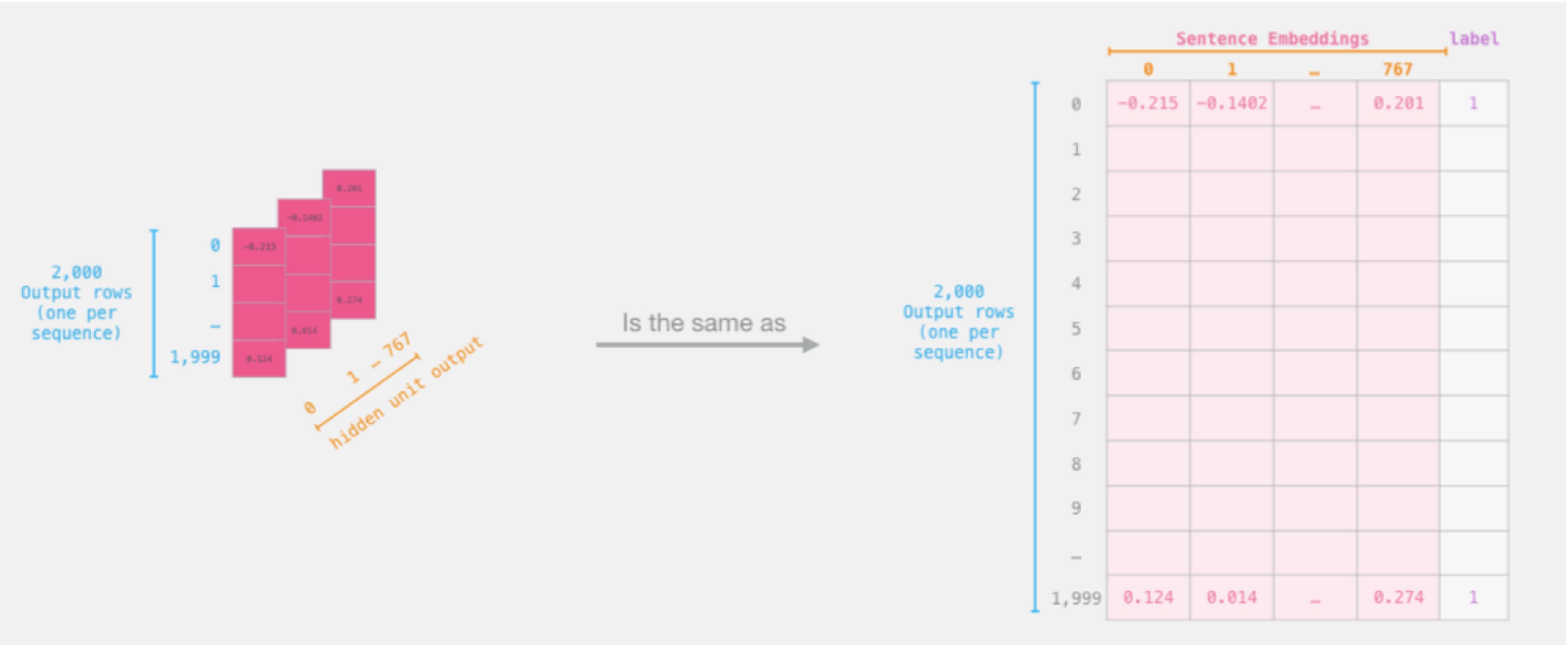
```python
# Slice the output for the first position for all the sequences, take all hidden unit outputs
features = last_hidden_states[0][:,0,:].numpy()
```



Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

The tensor sliced from BERT's output

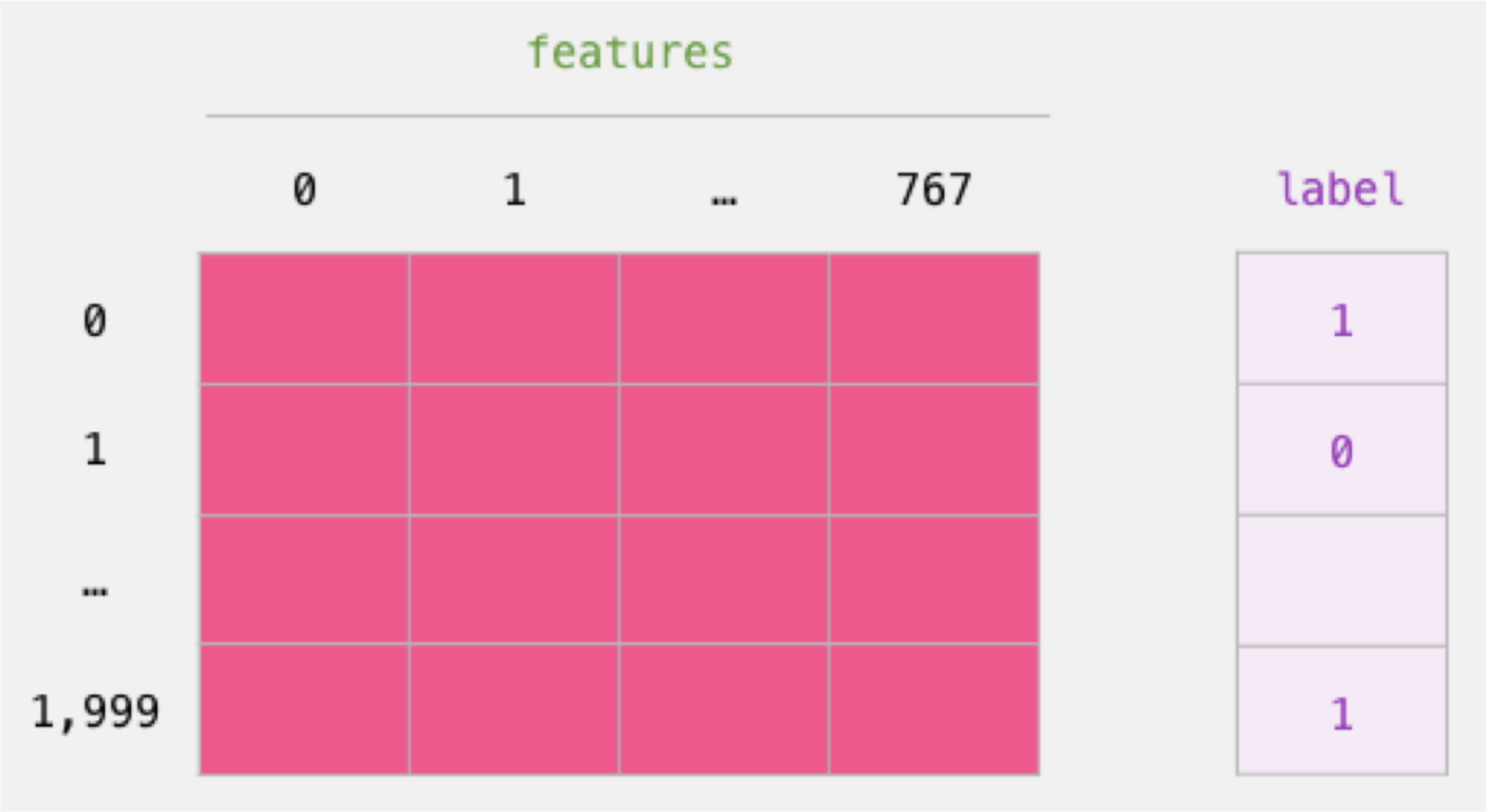
Sentence Embeddings



Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

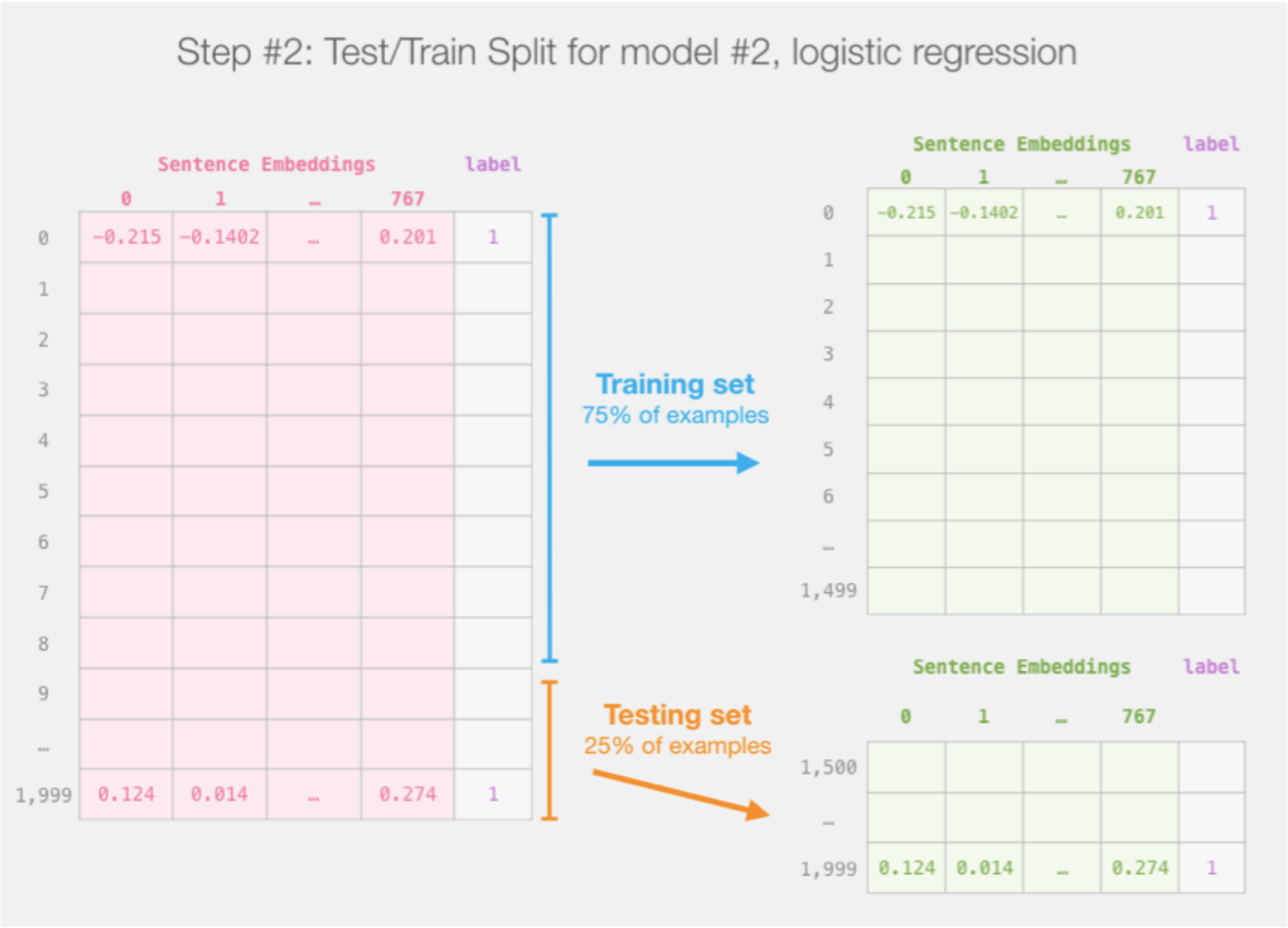
Dataset for Logistic Regression (768 Features)

The features are the output vectors of BERT for the [CLS] token (position #0)



Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

```
labels = df[1]
train_features, test_features, train_labels, test_labels =
train_test_split(features, labels)
```



Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Score Benchmarks Logistic Regression Model on SST-2 Dataset

```
# Training
lr_clf = LogisticRegression()
lr_clf.fit(train_features, train_labels)

#Testing
lr_clf.score(test_features, test_labels)

# Accuracy: 81%
# Highest accuracy: 96.8%
# Fine-tuned DistilBERT: 90.7%
# Full size BERT model: 94.9%
```

Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Sentiment Classification: SST2 Sentences from movie reviews

sentence	label
a stirring , funny and finally transporting re imagining of beauty and the beast and 1930s horror films	1
apparently reassembled from the cutting room floor of any given daytime soap	0
they presume their audience won't sit still for a sociology lesson	0
this is a visually stunning rumination on love , memory , history and the war between art and commerce	1
jonathan parker 's bartleby should have been the be all end all of the modern office anomie films	1

Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

A Visual Notebook to Using BERT for the First Time

A Visual Notebook to Using BERT for the First Time.ipynb

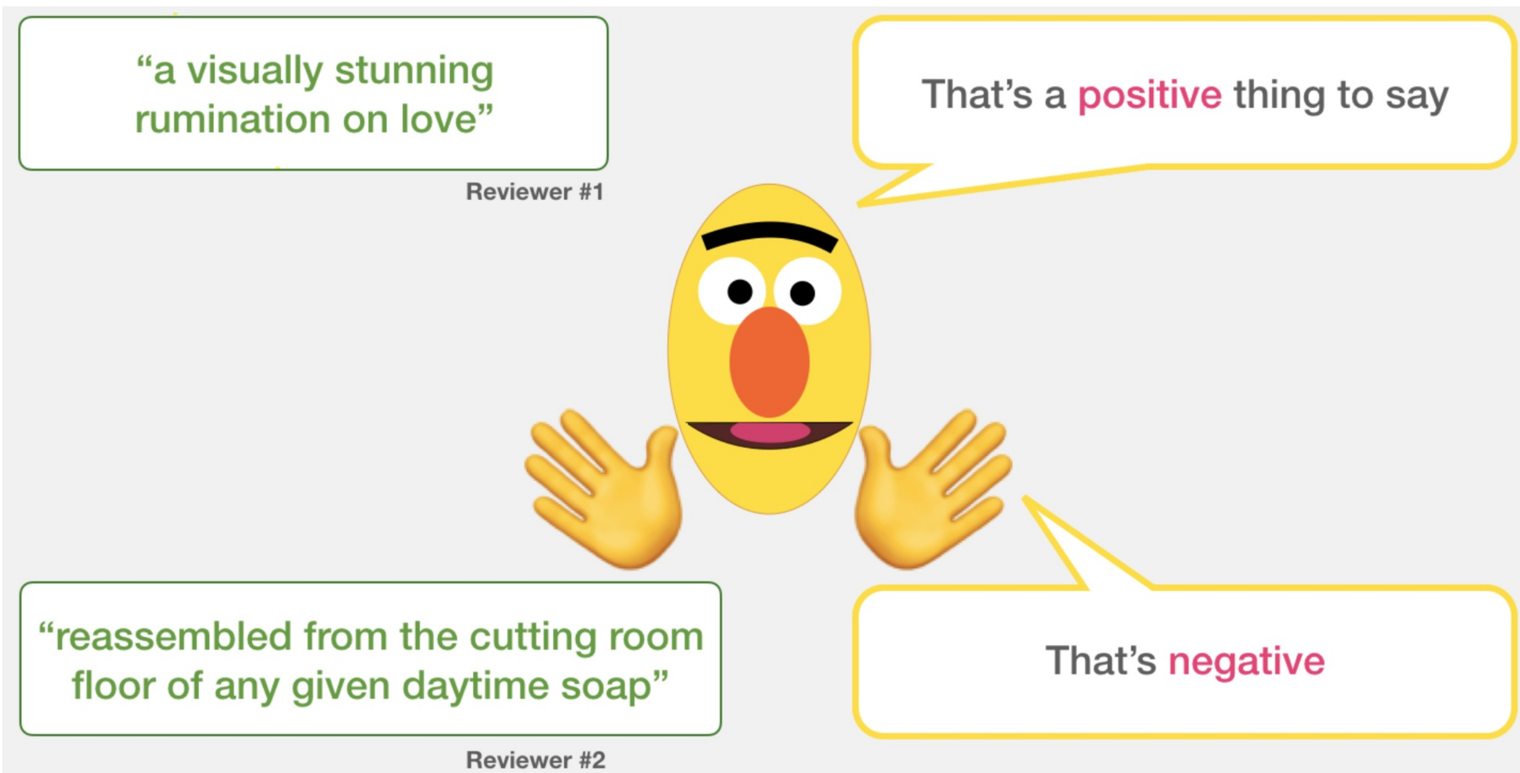
File Edit View Insert Runtime Tools Help Last edited on Nov 26, 2019

Share

+ Code + Text Copy to Drive

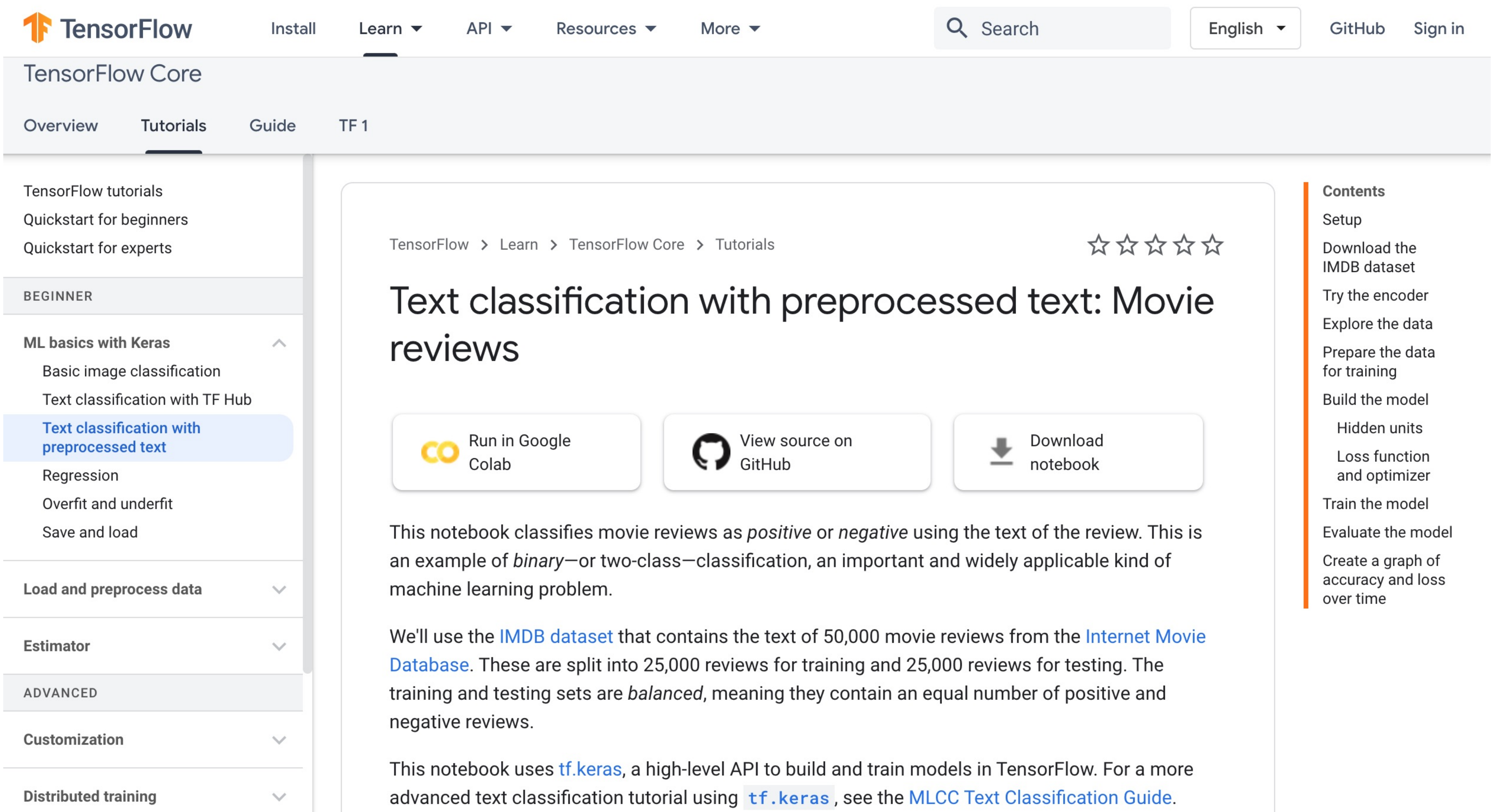
Connect Editing

A Visual Notebook to Using BERT for the First TIme.ipynb



https://colab.research.google.com/github/jalammar/jalammar.github.io/blob/master/notebooks/bert/A_Visual_Notebook_to_Using_BERT_for_the_First_Time.ipynb

Text classification with preprocessed text: Movie reviews



TensorFlow | Install | Learn | API | Resources | More | Search | English | GitHub | Sign in

TensorFlow Core

Overview | Tutorials | Guide | TF 1

TensorFlow tutorials
Quickstart for beginners
Quickstart for experts

BEGINNER

ML basics with Keras
Basic image classification
Text classification with TF Hub
Text classification with preprocessed text
Regression
Overfit and underfit
Save and load

Load and preprocess data

Estimator

ADVANCED

Customization

Distributed training

TensorFlow > Learn > TensorFlow Core > Tutorials

Text classification with preprocessed text: Movie reviews

Run in Google Colab | View source on GitHub | Download notebook

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.

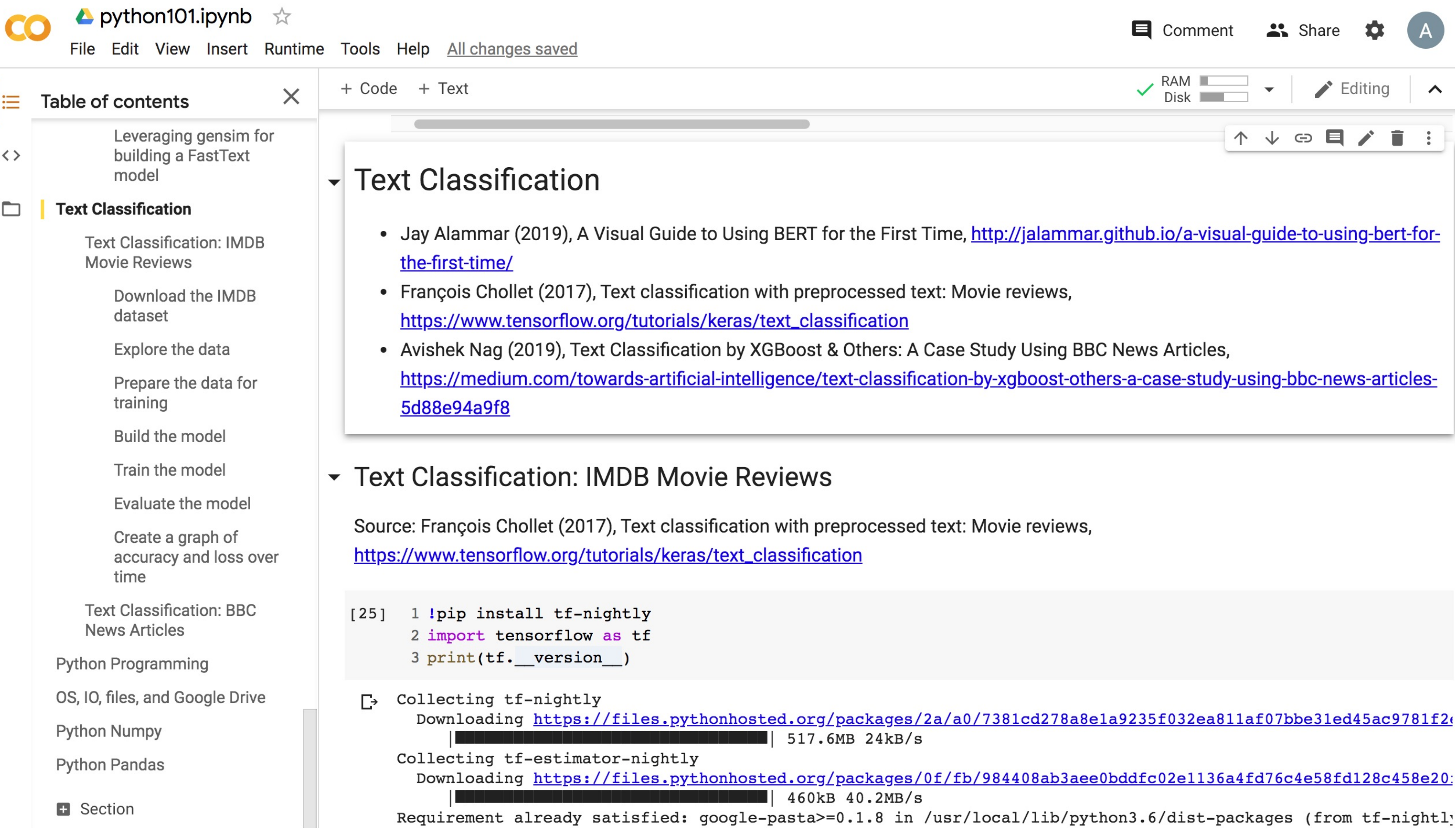
This notebook uses tf.keras, a high-level API to build and train models in TensorFlow. For a more advanced text classification tutorial using `tf.keras`, see the MLCC Text Classification Guide.

Contents

https://www.tensorflow.org/tutorials/keras/text_classification

Python in Google Colab (Python101)

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



https://tinyurl.com/imtkupython101

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 A

+ Code + Text RAM Disk Editing

Table of contents

Sentiment Analysis

- Source: Dipanjan Sarkar (2019), Text Analytics with Python: A Practitioner's Guide to Natural Language Processing, Second Edition. APress.

Sentiment Analysis - Unsupervised Lexical

[2]

```
#!wget http://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz
!wget 'http://mail.tku.edu.tw/myday/data/example/movie_reviews.csv'
!ls
```

[3]

```
import numpy as np
import pandas as pd
import tensorflow as tf
import tensorflow_hub as hub

df = pd.read_csv('http://mail.tku.edu.tw/myday/data/example/movie_reviews.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   review     50000 non-null  object
 1   sentiment  50000 non-null  object
dtypes: object(2)
```

https://tinyurl.com/imtkupython101

Summary

- Python for
  Natural Language Processing
- Processing Text and
  Understanding Text

References

- Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson.
- Dipanjan Sarkar (2019), Text Analytics with Python: A Practitioner's Guide to Natural Language Processing, Second Edition. APress.
- Benjamin Bengfort, Rebecca Bilbro, and Tony Ojeda (2018), Applied Text Analysis with Python: Enabling Language-Aware Data Products with Machine Learning, O'Reilly.
- Gabe Ignatow and Rada F. Mihalcea (2017), An Introduction to Text Mining: Research Design, Data Collection, and Analysis, SAGE Publications.
- 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.
- 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/
- Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/
- François Chollet (2017), Text classification with preprocessed text: Movie reviews, https://www.tensorflow.org/tutorials/keras/text_classification
- Google Developers (2020), Machine Learning Guides: Text Classification, https://developers.google.com/machine-learning/guides/text-classification
- Avishek Nag (2019), Text Classification by XGBoost & Others: A Case Study Using BBC News Articles, https://medium.com/towards-artificial-intelligence/text-classification-by-xgboost-others-a-case-study-using-bbc-news-articles-5d88e94a9f8
- Min-Yuh Day (2020), Python 101, https://tinyurl.com/imtkupython101

國立臺北護理健康大學
National Taipei University of Nursing and Health Sciences

Q & A

Tamkang University
淡江大學

人工智慧文本分析基礎與應用
(Artificial Intelligence for Text Analytics: Foundations and Applications)

Time: 2020/05/22 (Fri) (9:10 -12:00)
Place: 國立臺北護理健康大學 (台北市明德路365號) G210
Host: 祝國忠 院長 (健康科技學院院長)

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/

2020-05-22