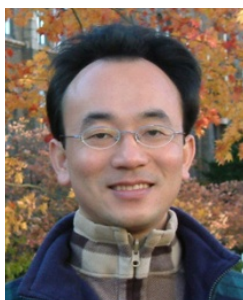


人工智慧文本分析基礎與應用 (Artificial Intelligence for Text Analytics: Foundations and Applications)

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)

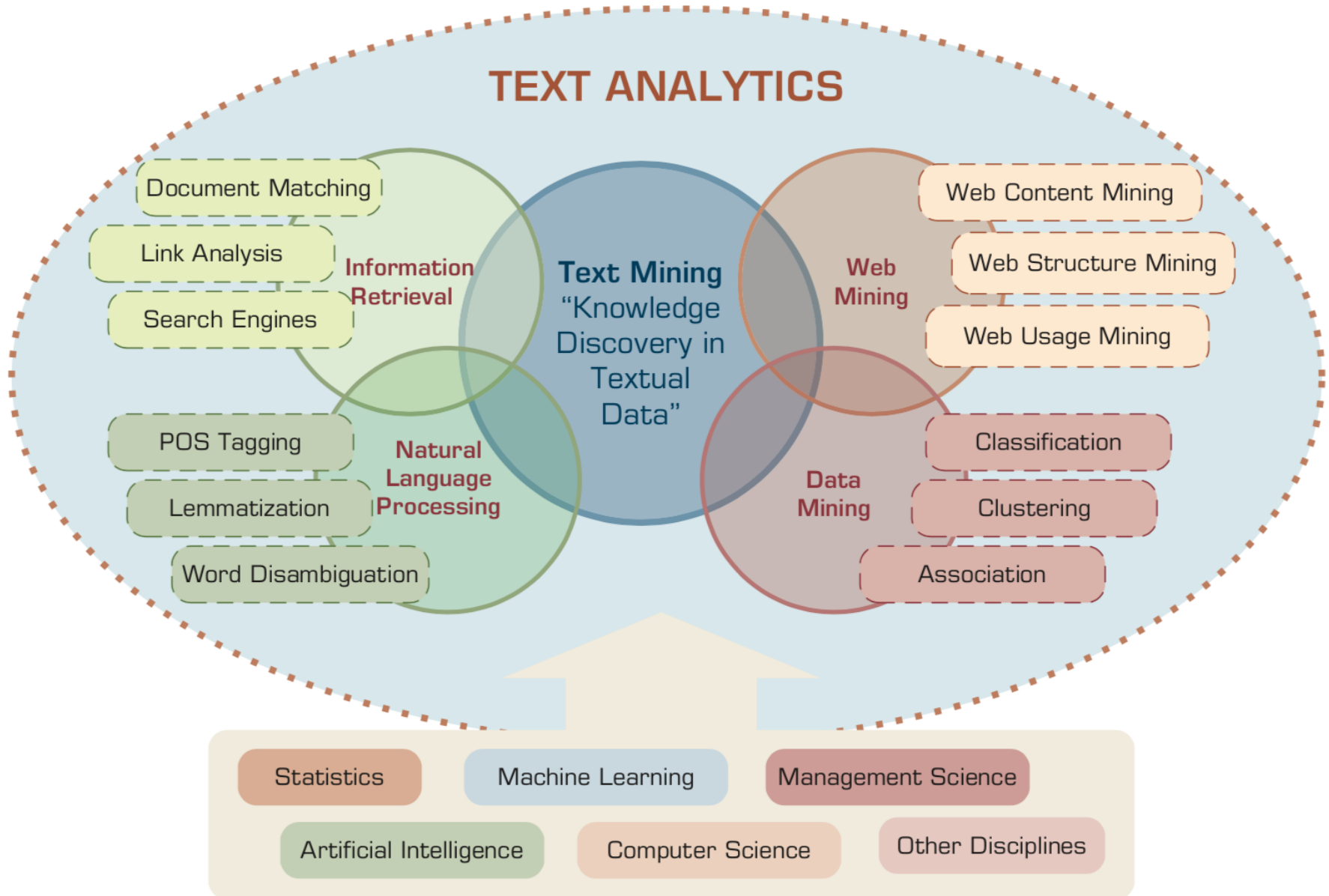
6. 問答系統與對話系統

(Question Answering and Dialogue Systems)

Outline

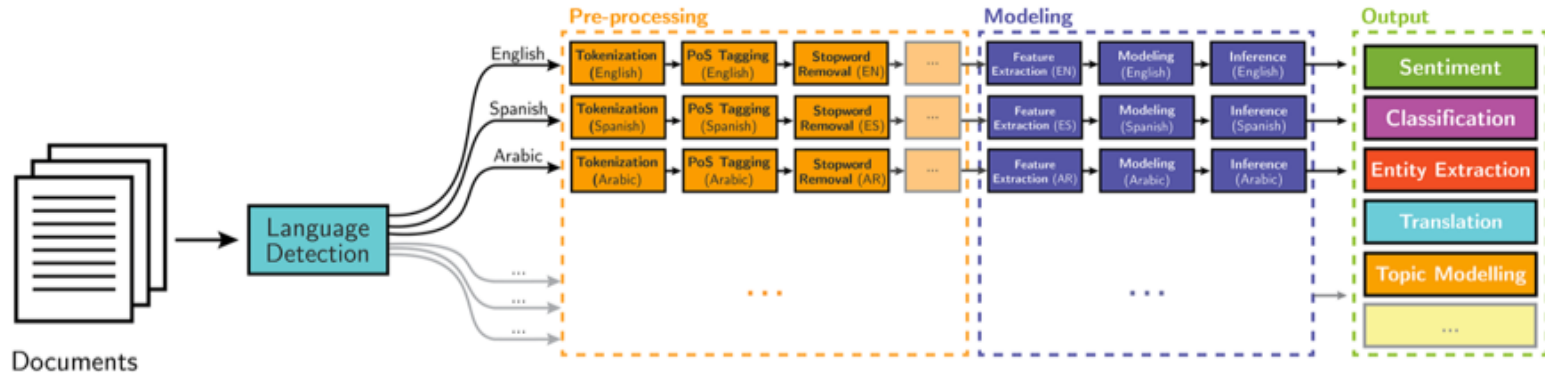
- **Python for
Natural Language Processing**
- **Processing Text and
Understanding Text**

Text Analytics and Text Mining

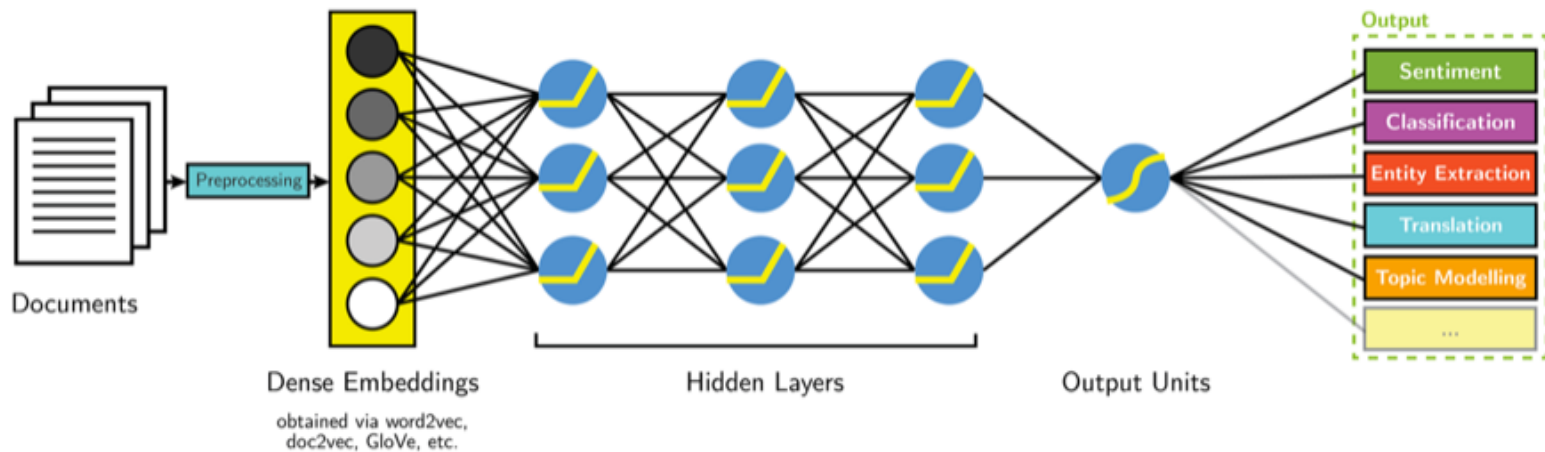


NLP

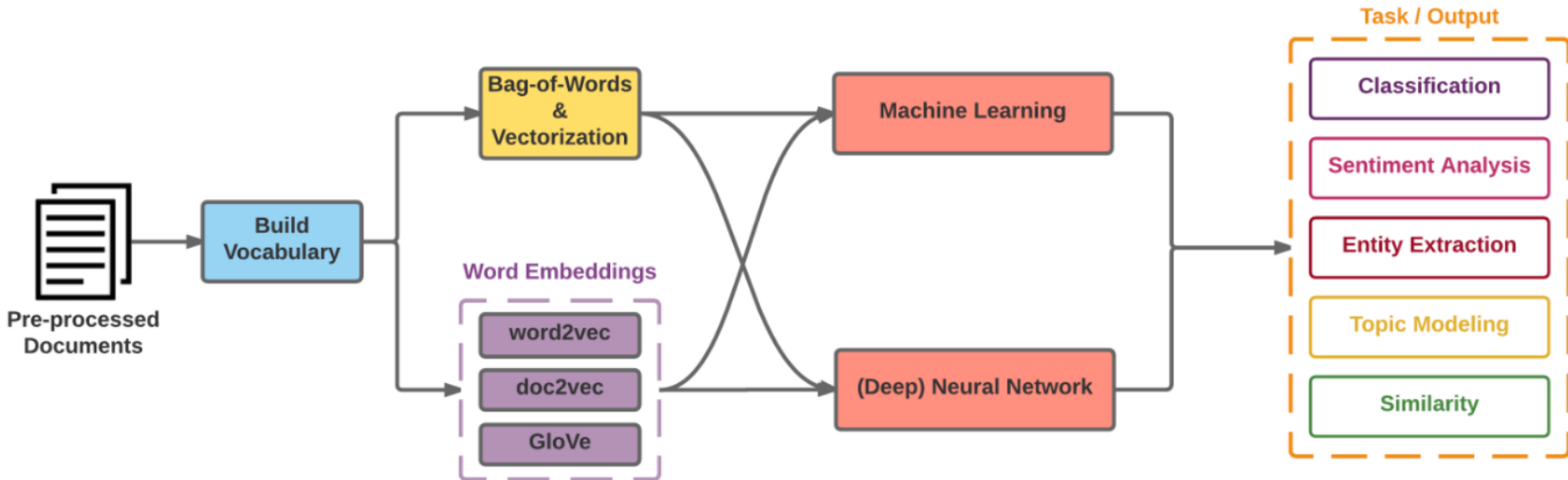
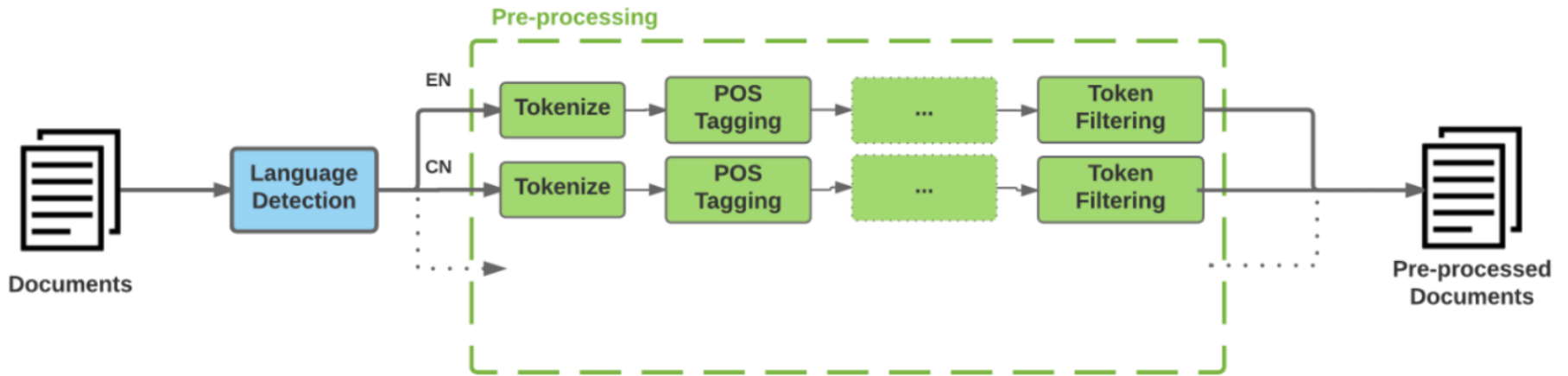
Classical NLP



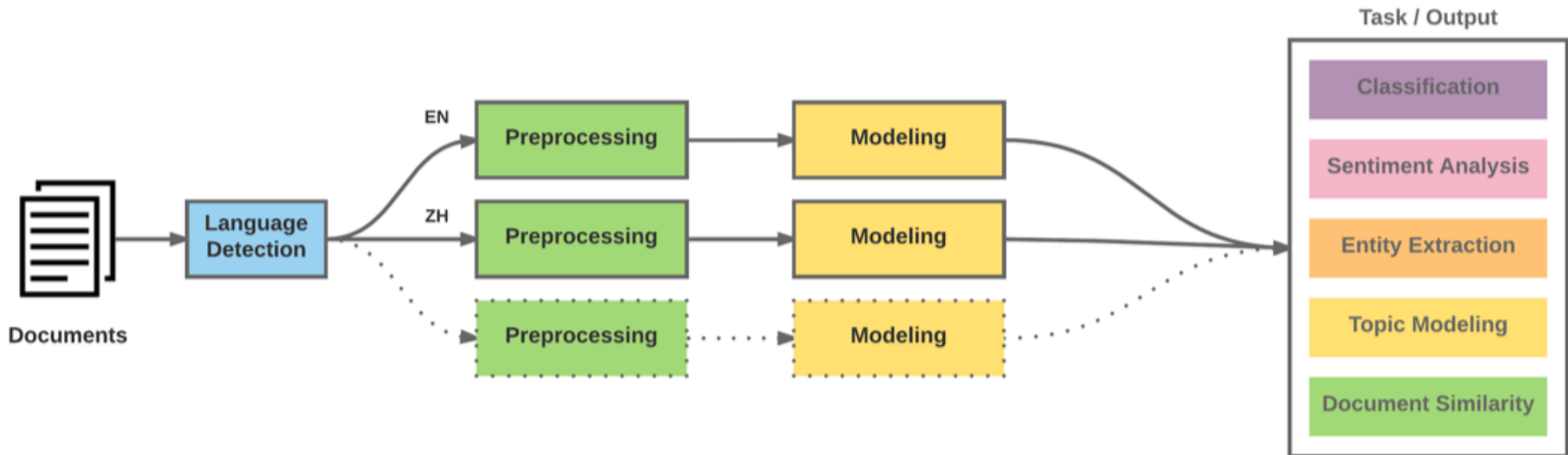
Deep Learning-based NLP



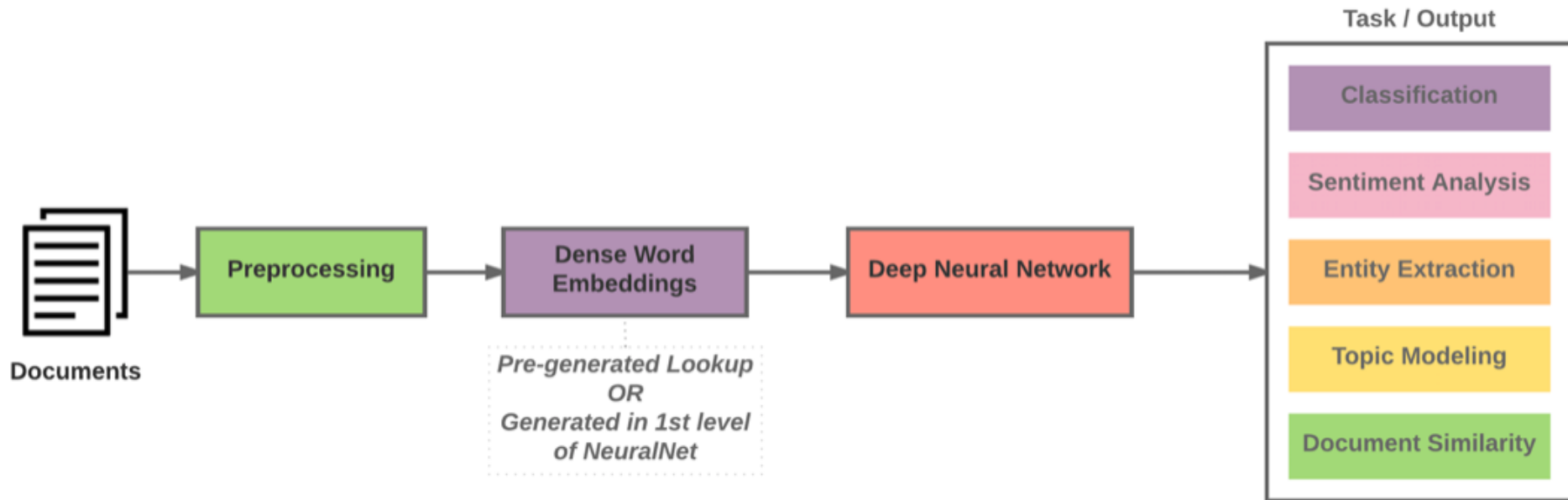
Modern NLP Pipeline



Modern NLP Pipeline



Deep Learning NLP







Papers with Code: NLP

[Browse](#) > Natural Language Processing

Natural Language Processing

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

Representation Learning

 <p>Representation Learning</p> <p>📄 7 leaderboards 548 papers with code</p>	 <p>Word Embeddings</p> <p>454 papers with code</p>	 <p>Graph Embedding</p> <p>116 papers with code</p>	 <p>Network Embedding</p> <p>62 papers with code</p>	<p>Sentence Embeddings</p> <p>📄 3 leaderboards 52 papers with code</p>
--	---	---	--	---

▶ [See all 17 tasks](#)

Machine Translation

 <p>Machine Translation</p> <p>📄 45 leaderboards 612 papers with code</p>	 <p>Transliteration</p> <p>17 papers with code</p>	 <p>Unsupervised Machine Translation</p> <p>📄 9 leaderboards 12 papers with code</p>	 <p>Low-Resource Neural Machine Translation</p> <p>8 papers with code</p>	 <p>Multimodal Machine Translation</p> <p>7 papers with code</p>
--	---	---	--	---

▶ [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/csmlm_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 NarrativeQA 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

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 'Storage' section shows '0 bytes of 15 GB used' and an 'UPGRADE STORAGE' button. A 'Files' section is partially visible at the bottom.

The 'New' button is highlighted with a red dashed border. A dropdown menu is open, 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.

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 'OK' button with a red dashed border. To the right of the notification card, there is a 'RATE IT' button with a star icon, followed by the text 'Productivity' and a star rating of five stars with '(195)' next to it. The background of the dialog box shows a list of apps, 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 'New' button is highlighted with a red dashed box, and its dropdown menu is open. The 'More' option in the dropdown is also highlighted with a red dashed box. The 'More' dropdown is open, showing options like Google Forms, Google Drawings, Google My Maps, Google Sites, and Colaboratory. The 'Colaboratory' option is highlighted with a red dashed box. The 'Storage' section shows 0 bytes of 15 GB used. The 'Files' section shows a grid of cards for 'Store safely', 'Sync seamlessly', 'Access anywhere', and 'Share easily'. A notification for 'Get Backup and Sync for Mac' is visible in the bottom left corner.

My Drive - Google Drive

Search Drive

My Drive

Quick Access

New

My Drive

Computers

Shared with me

Recent

Starred

Trash

Backups

Storage

0 bytes of 15 GB used

UPGRADE STORAGE

Files

Store safely

Sync seamlessly

Access anywhere

Share easily

Google Docs

Google Sheets

Google Slides

More

Google Forms

Google Drawings

Google My Maps

Google Sites

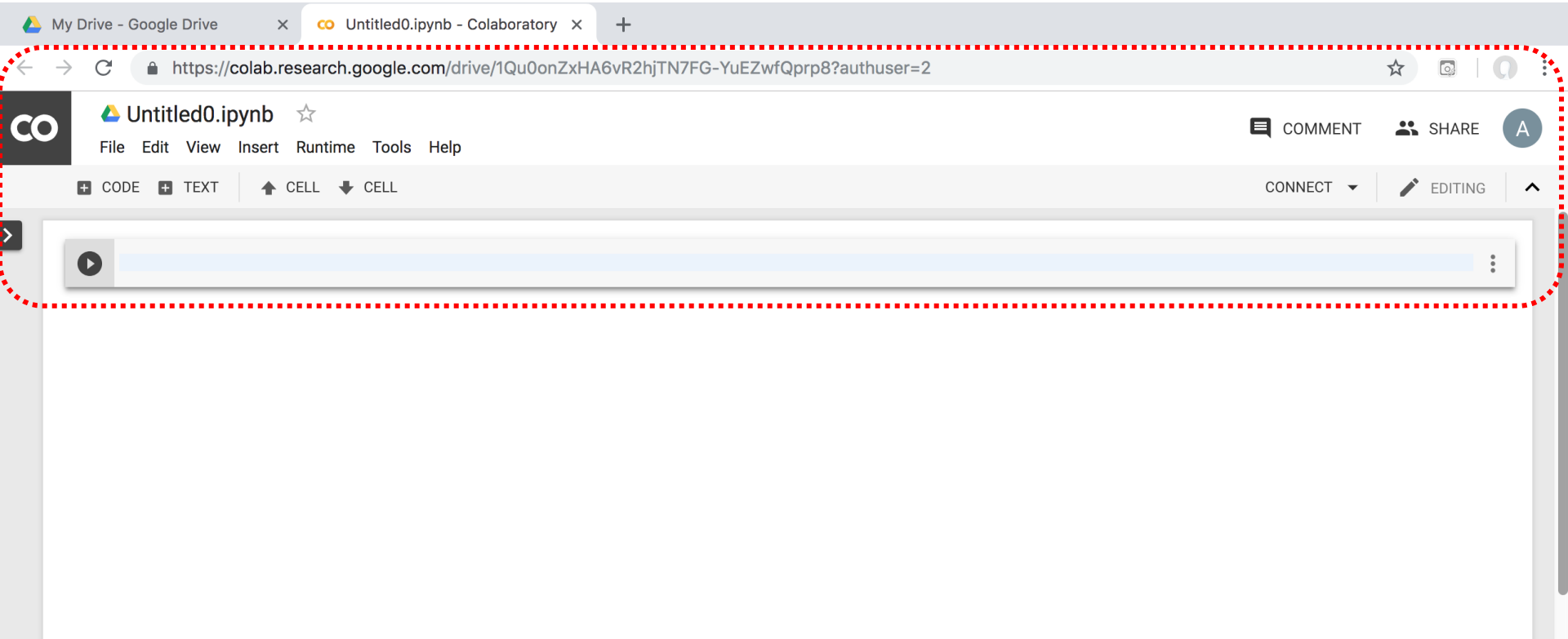
Colaboratory

Connect more apps

Name ↑

Get Backup and Sync for Mac

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 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 Colab interface includes a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". Below the menu bar 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 of the screen. It contains the following options:

- Runtime type:** A dropdown menu currently set to "Python 3".
- Hardware accelerator:** A dropdown menu currently set to "GPU".
- Omit code cell output when saving this notebook

At the bottom right of the dialog box are two buttons: "CANCEL" and "SAVE".

Google Colab Python Hello World

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



The screenshot displays the Google Colaboratory web interface. At the top, the browser tab is labeled "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 navigation menu includes "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". On the right side, there are buttons for "COMMENT" and "SHARE", along with a user profile icon labeled "A". Below the menu, the interface shows a toolbar with options: "+ CODE", "+ TEXT", "↑ CELL", and "↓ CELL". The status bar indicates "CONNECTED" with a green checkmark and "EDITING" with a pencil icon. The main workspace contains a code cell with a play button icon on the left and a vertical ellipsis on the right. The code in the cell is `print('Hello World')`. Below the code cell, the output is displayed as "Hello World" with a copy icon to its left.

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

RAM Disk Editing

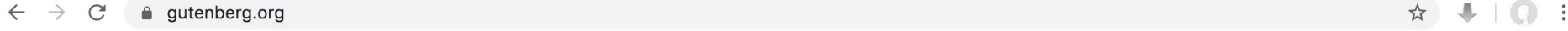
Table of contents

- Python101
 - Python File Input / Output
 - OS, IO, files, and Google Drive
 - Python Programming
 - Python String and Text
 - Python Numpy
 - Python Pandas
 - Deep Learning for Financial Time Series Forecasting
 - Text Analytics and Natural Language Processing (NLP)
 - Python for Natural Language Processing
 - spaCy Chinese Model
 - Open Chinese Convert (OpenCC, 開放中文轉換)
 - Jieba 結巴中文分詞
 - Natural Language Toolkit (NLTK)
 - Stanza: A Python NLP Library for Many Human Languages
 - Text Processing and Understanding
 - NLTK (Natural Language Processing with Python – Analyzing Text with the

<https://tinyurl.com/imtkupython101>

Processing and Understanding Text

Free eBooks - Project Gutenberg



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](#).



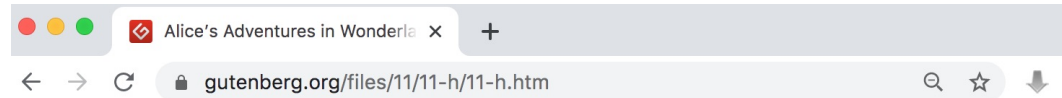
Free eBooks - Project Gutenberg

Alice in Wonderland

ALICE IN WONDERLAND



LEWIS CARROLL



The Project Gutenberg eBook of Alice's Adventures in Wonderland, by Lewis Carroll

This eBook is for the use of anyone anywhere at no cost and with almost no restrictions whatsoever. You may copy it, give it away or re-use it under the terms of the Project Gutenberg License included with this eBook or online at www.gutenberg.org

Title: Alice's Adventures in Wonderland

Author: Lewis Carroll

Release Date: June 25, 2008 [EBook #11]

Last Updated: February 22, 2020

Language: English

Character set encoding: UTF-8

*** START OF THIS PROJECT GUTENBERG EBOOK ALICE'S ADVENTURES IN WONDERLAND ***

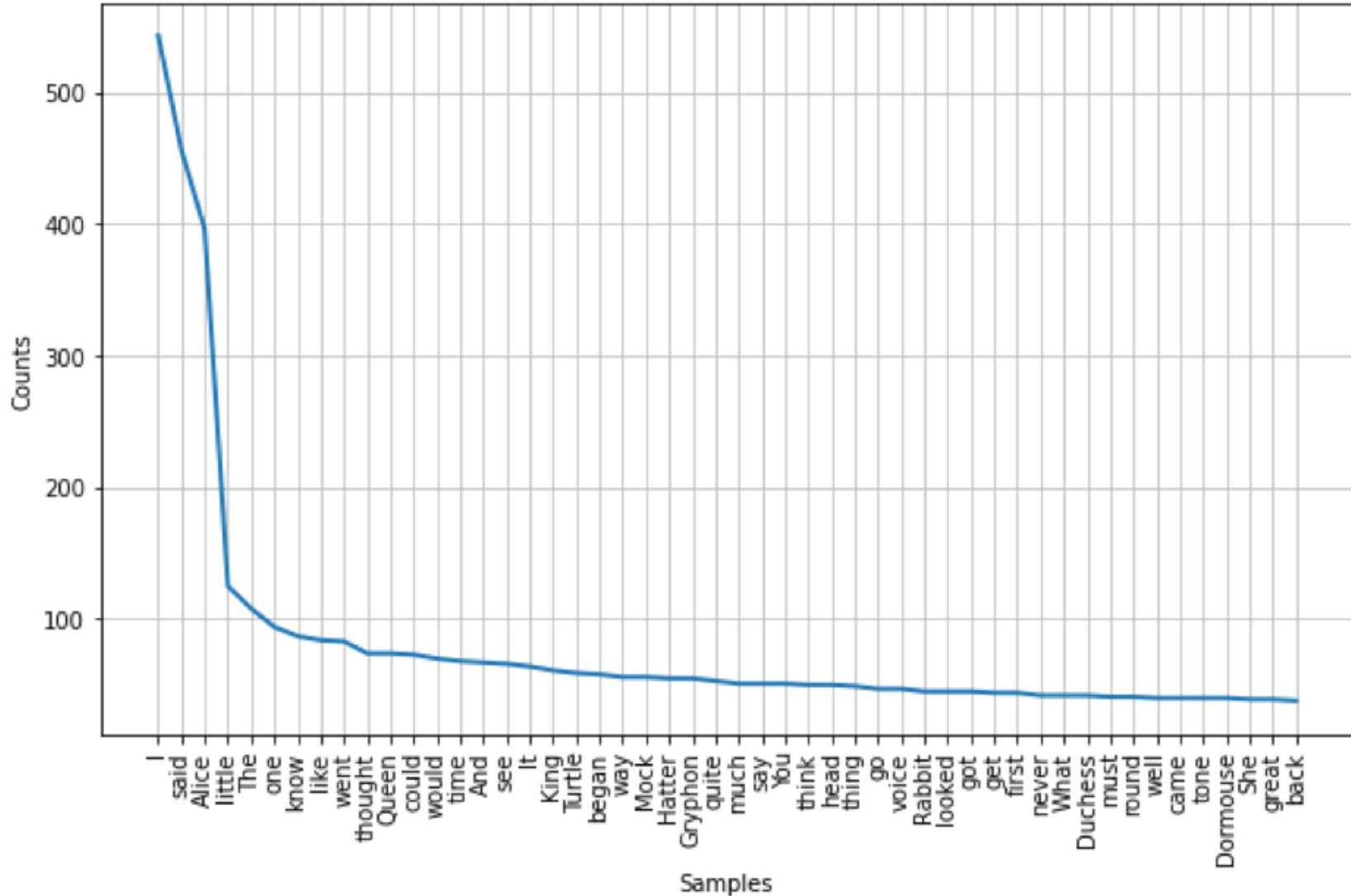
Produced by Arthur DiBianca and David Widger

ALICE IN WONDERLAND



Alice Top 50 Tokens

50 most common tokens (no stopwords or punctuation)

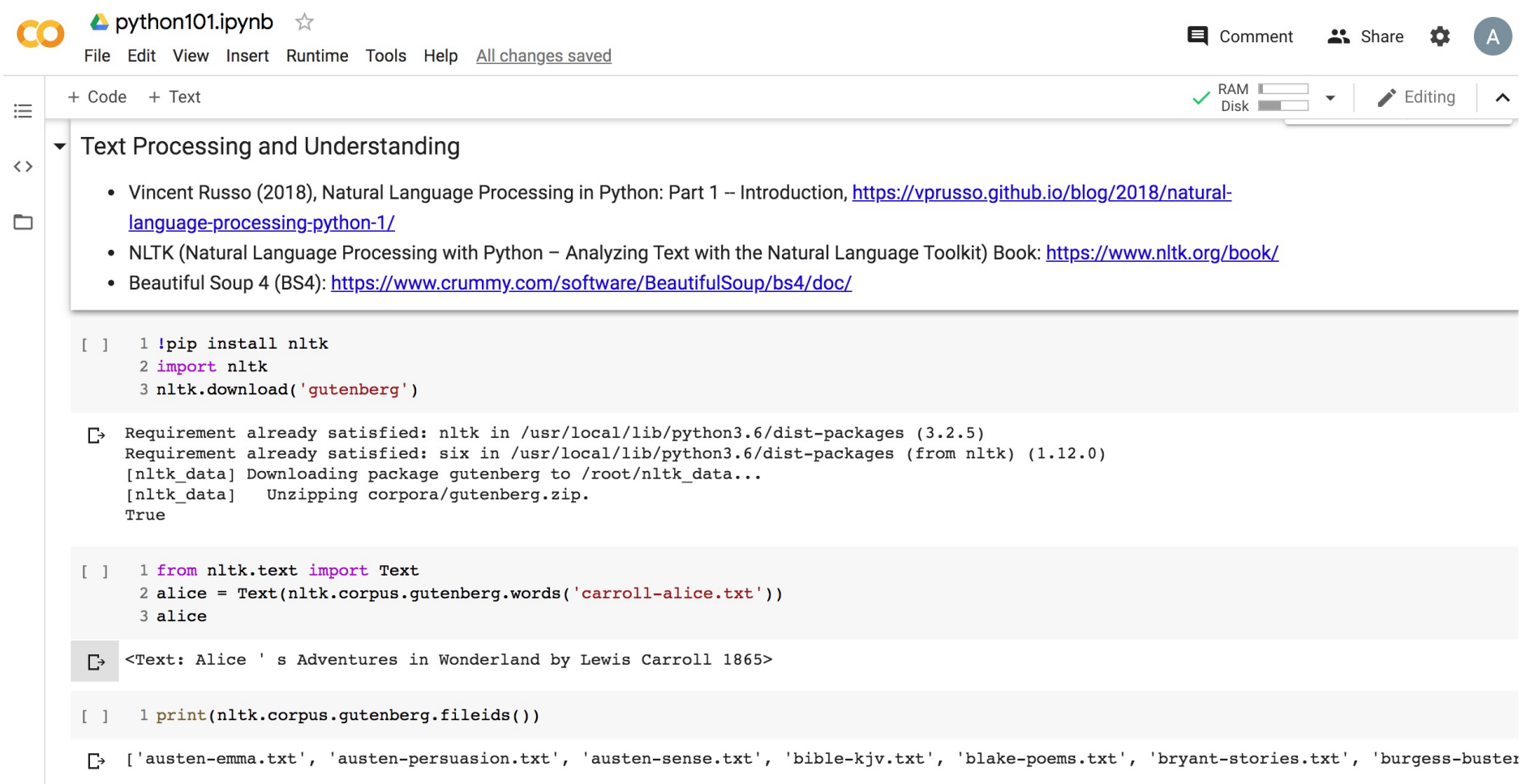


<https://tinyurl.com/imtkupython101>

Python in Google Colab (Python101)

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

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



The screenshot shows a Google Colab notebook titled 'python101.ipynb'. The interface includes a top navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help' menus, along with 'Comment', 'Share', and 'Settings' icons. The notebook content is organized into sections, with the current section being 'Text Processing and Understanding'. This section contains a list of references and two code cells. The first code cell shows the installation of NLTK and downloading the Gutenberg corpus. The second code cell shows the creation of a Text object from the downloaded corpus. The output of the second code cell is a text snippet: '<Text: Alice ' s Adventures in Wonderland by Lewis Carroll 1865>'. The third code cell shows the retrieval of file IDs from the Gutenberg corpus, with the output being a list of file names.

python101.ipynb ☆

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

Comment Share Settings 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/>

```
[ ] 1 !pip install nltk  
    2 import nltk  
    3 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
```

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

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

```
[ ] 1 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:

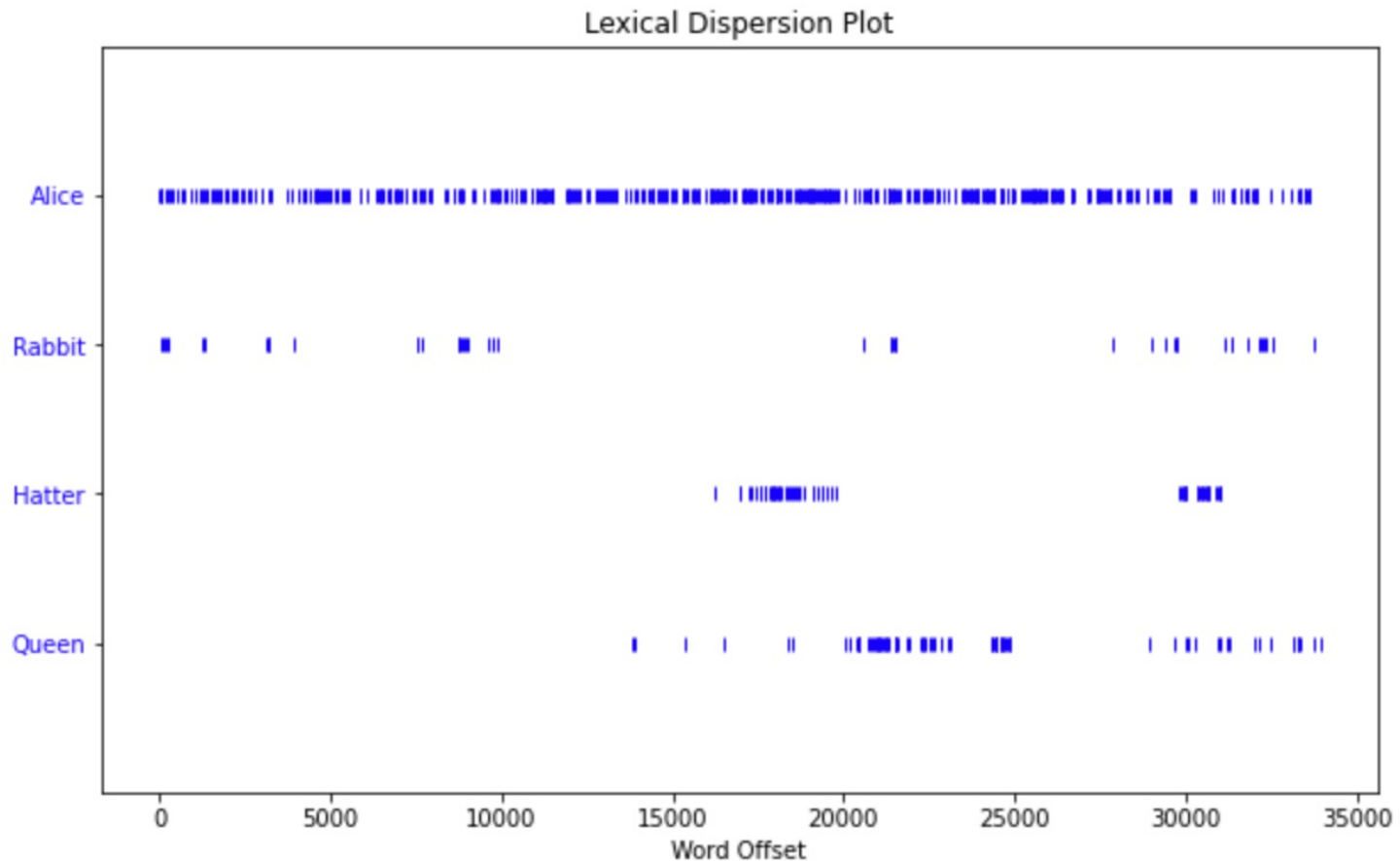
```

] CHAPTER I . Down the Rabbit - Hole Alice ' s Adventures in Wonderland by Lewi
what is the use of a book , ' thought Alice was beginning to get very tired of s
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
```



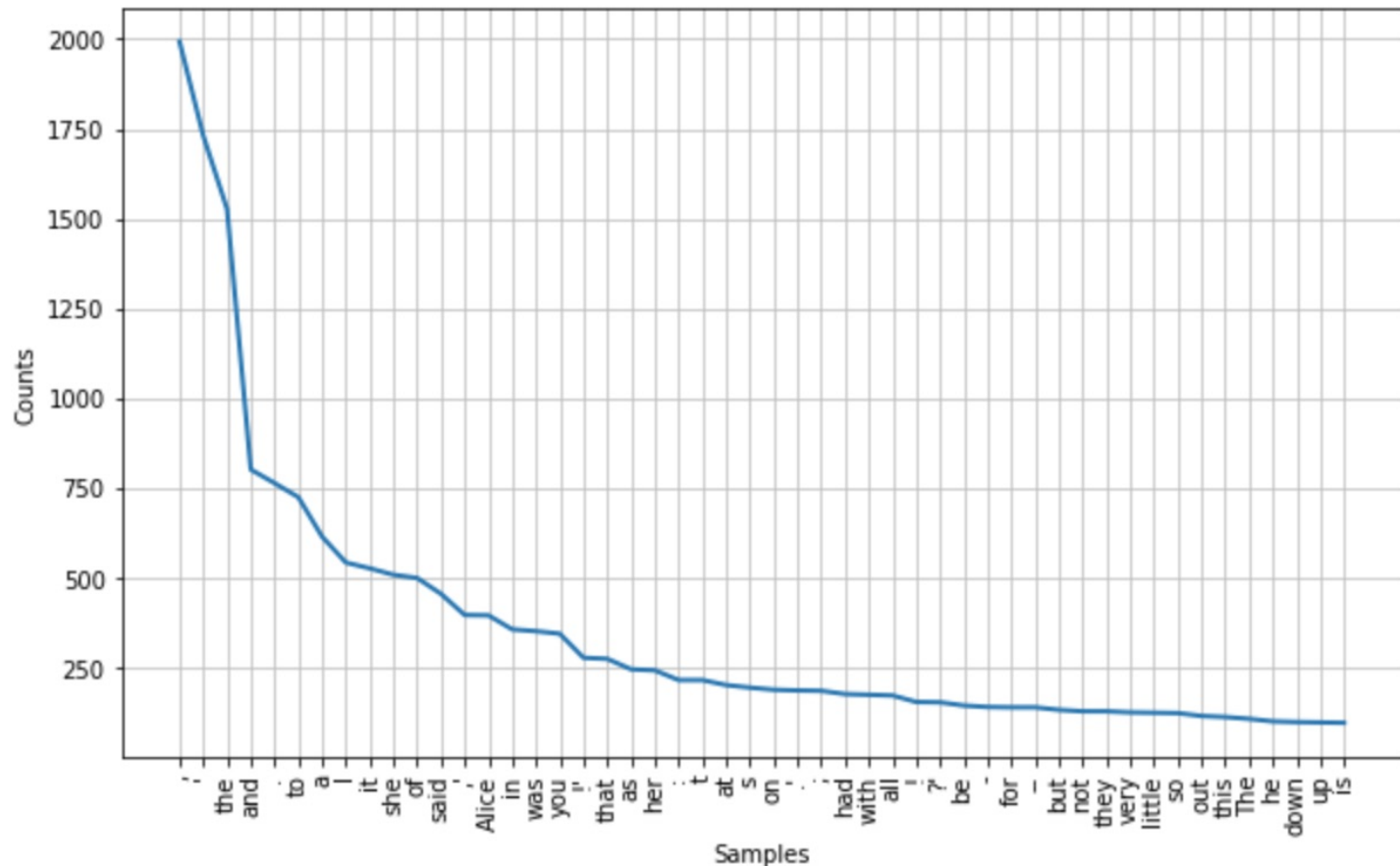
```
alice.dispersion_plot(["Alice", "Rabbit",  
                    "Hatter", "Queen"])
```

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



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

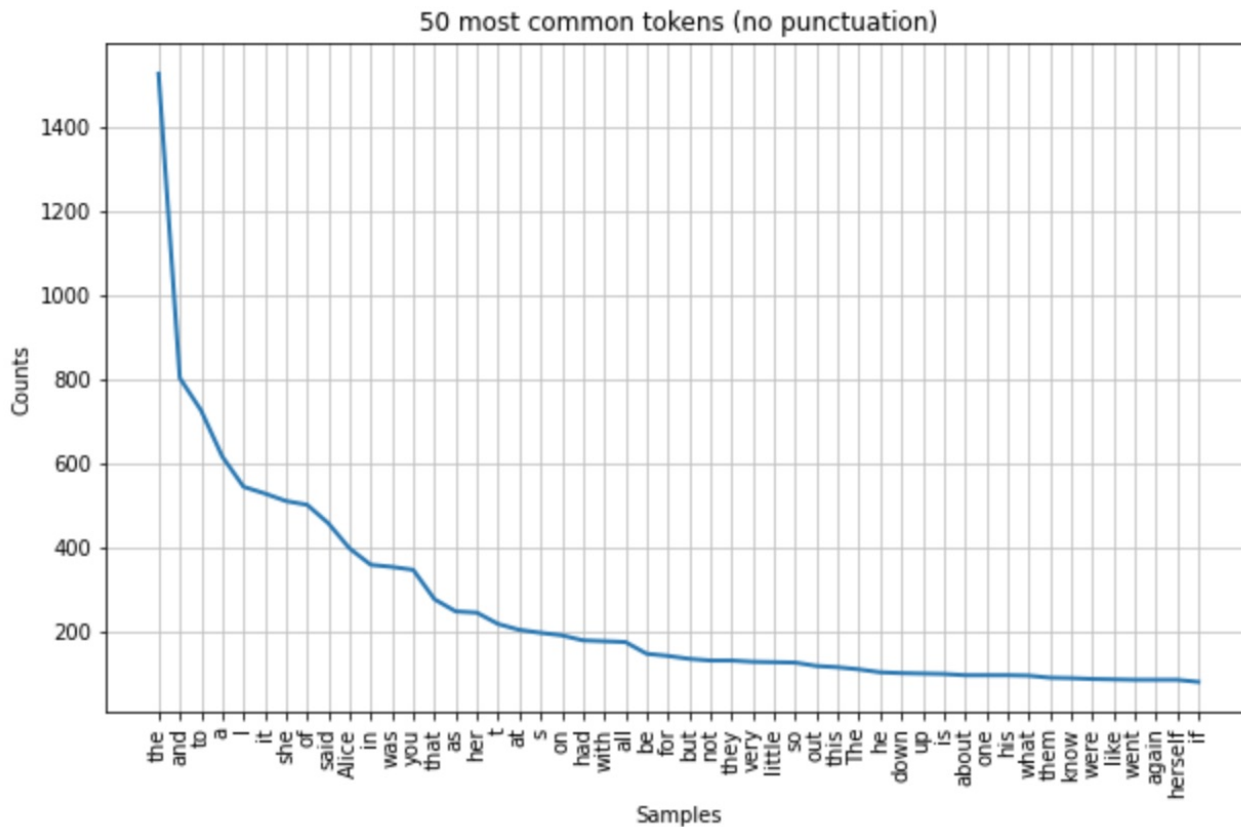
```
1 #import matplotlib.pyplot as plt
2 plt.figure(figsize=(10, 6))
3 fdist = nltk.FreqDist(alice)
4 fdist.plot(50)
```



<https://tinyurl.com/imtkupython101>

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

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



```
nlTK.download('stopwords')
```

```
stopwords = nlTK.corpus.stopwords.words('english')
```

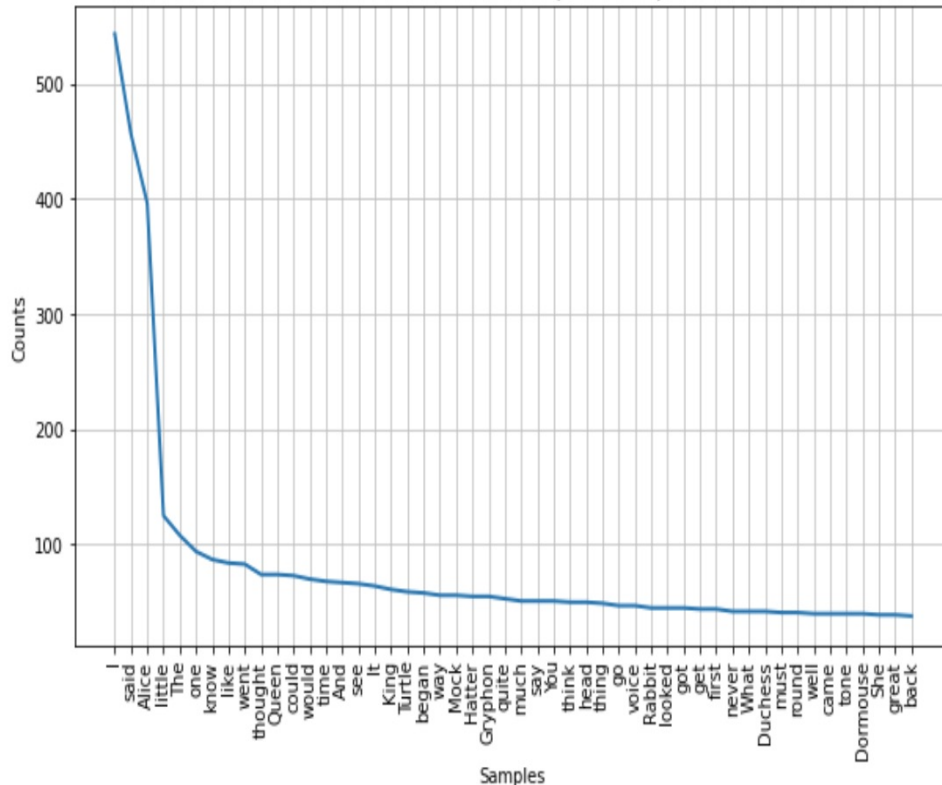
```
1 import nlTK
2 nlTK.download('stopwords')
3 stopwords = nlTK.corpus.stopwords.words('english')
4 stopwords
```

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

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

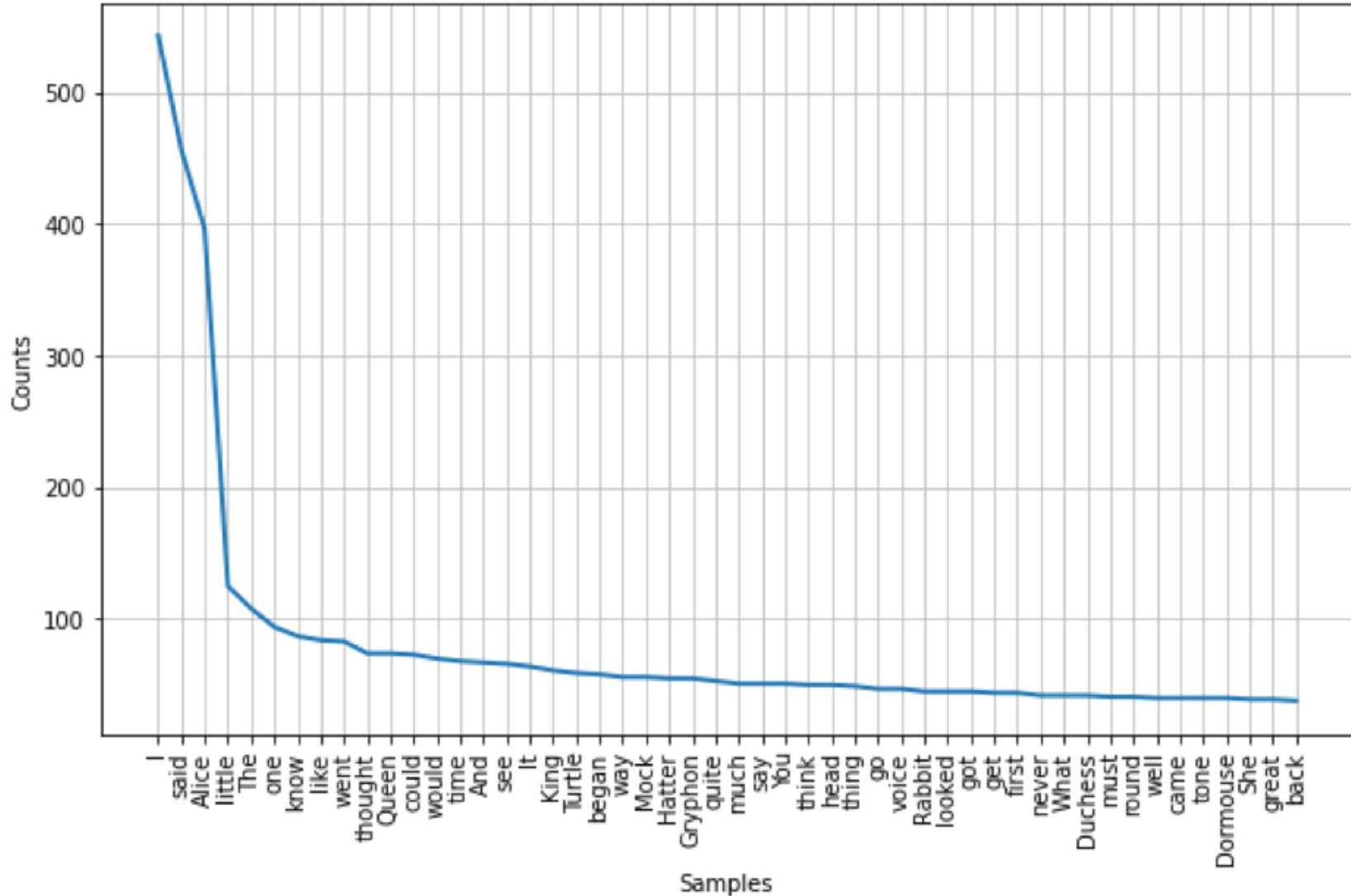
```
1 #import matplotlib.pyplot as plt
2 plt.figure(figsize=(10, 6))
3 fdist_no_punc_no_stopwords = nltk.FreqDist(dict((word, freq) for word, freq in fdist.items() if word not in stopwords and word.isalpha()))
4 fdist_no_punc_no_stopwords.plot(50, cumulative=False, title="50 most common tokens (no stopwords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

50 most common tokens (no stopwords or punctuation)



Alice Top 50 Tokens

50 most common tokens (no stopwords or punctuation)



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}
```



```
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)
```

```
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?']
```

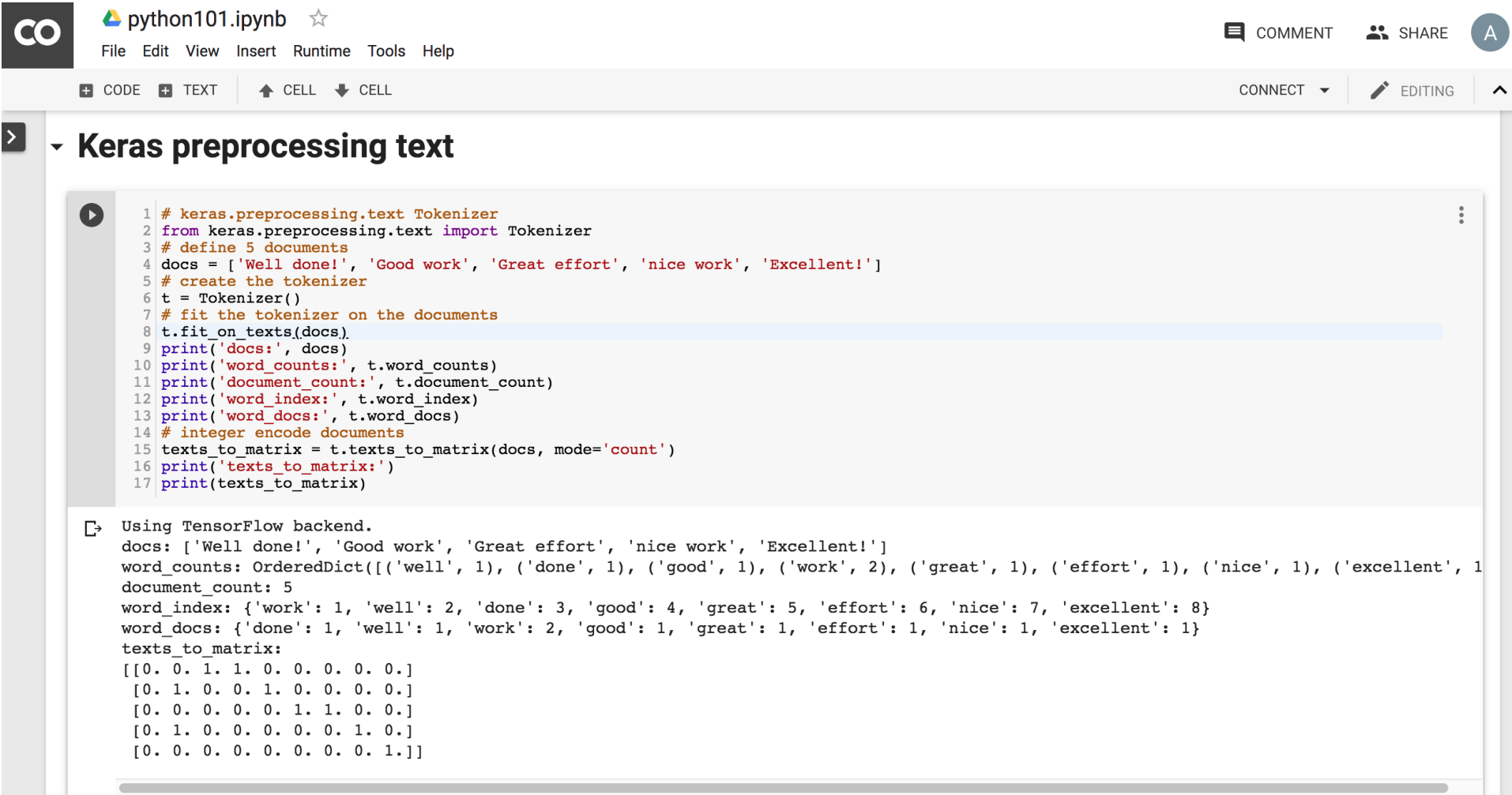
```
Word Index = {'<OOV>': 1, 'my': 2, 'love': 3,  
'dog': 4, 'i': 5, 'you': 6, 'cat': 7, 'do':  
8, 'think': 9, 'is': 10, 'amazing': 11}
```

```
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]]
```

Python in Google Colab

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



The screenshot shows a Google Colab notebook interface. At the top, the notebook is titled "python101.ipynb" and has a star icon. The menu bar includes "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". On the right, there are buttons for "COMMENT", "SHARE", and a user profile icon. Below the menu bar, there are tabs for "CODE", "TEXT", and "CELL", along with "CONNECT" and "EDITING" options.

The main content area is titled "Keras preprocessing text". It contains a code cell with the following Python code:

```
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)
```

Below the code cell, the output is displayed:

```
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.]]
```

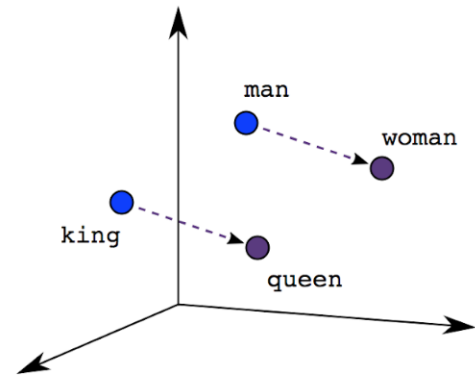
<https://tinyurl.com/imtkupython101>

One-hot encoding

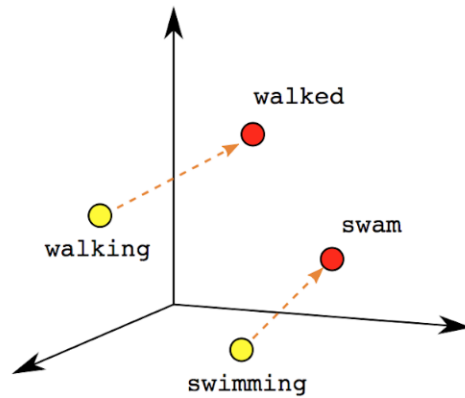
'The mouse ran up the clock' =

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]	

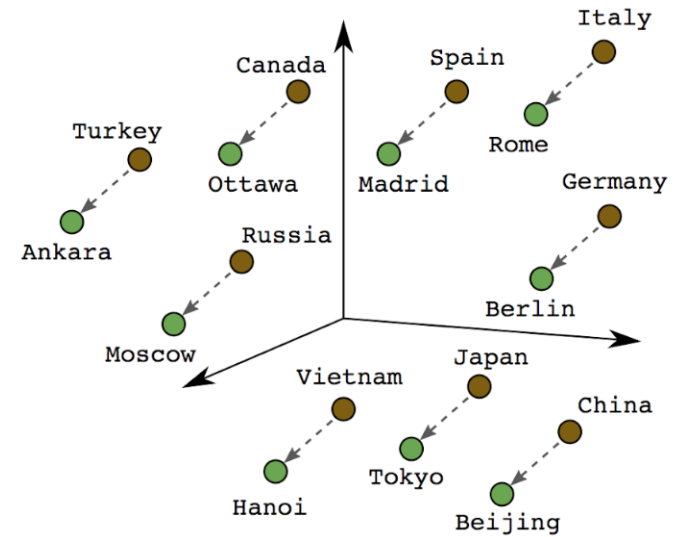
Word embeddings



Male-Female

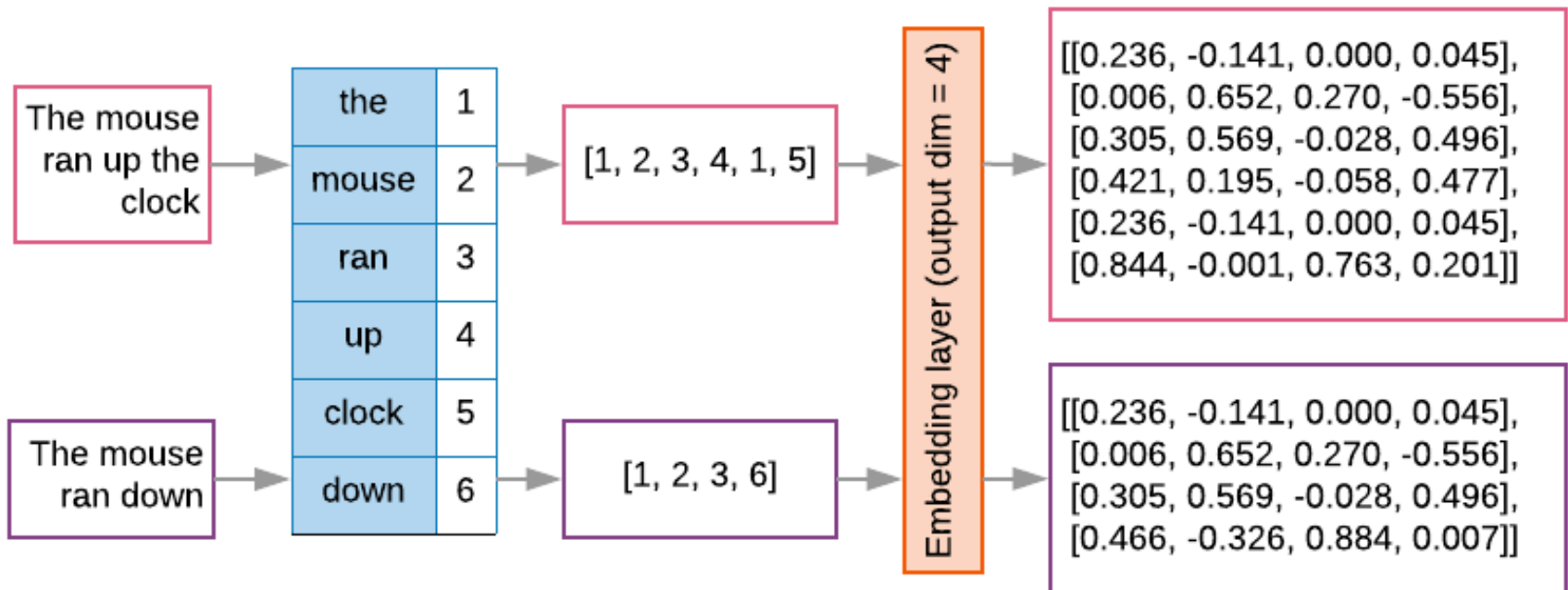


Verb Tense



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')
```

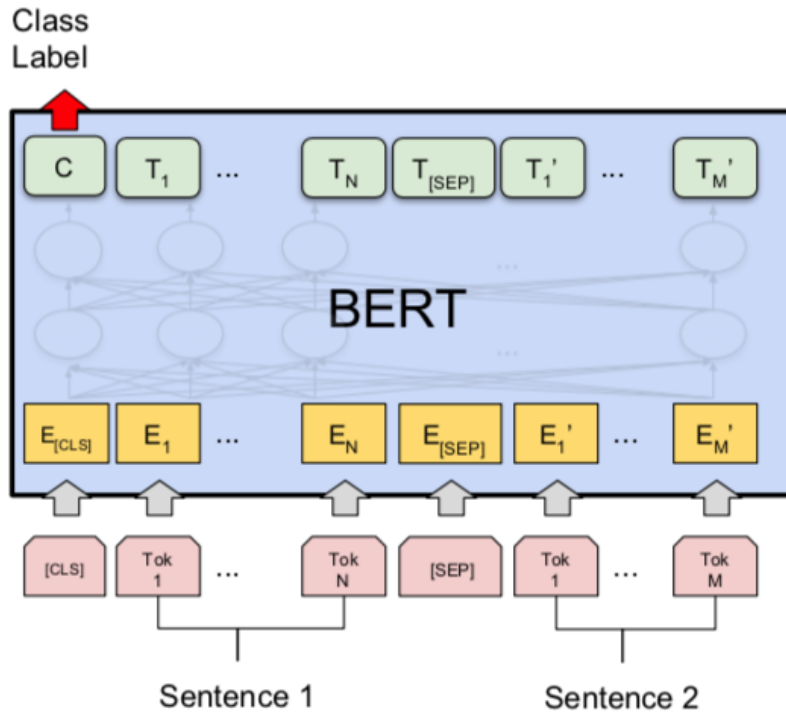
```
docs: ['Well done!', 'Good work', 'Great effort',  
'nice work', 'Excellent!']  
word_counts: OrderedDict([('well', 1), ('done', 1),  
('good', 1), ('work', 2), ('great', 1), ('effort', 1),  
('nice', 1), ('excellent', 1)])  
document_count: 5  
word_index: {'work': 1, 'well': 2, 'done': 3, 'good':  
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')`

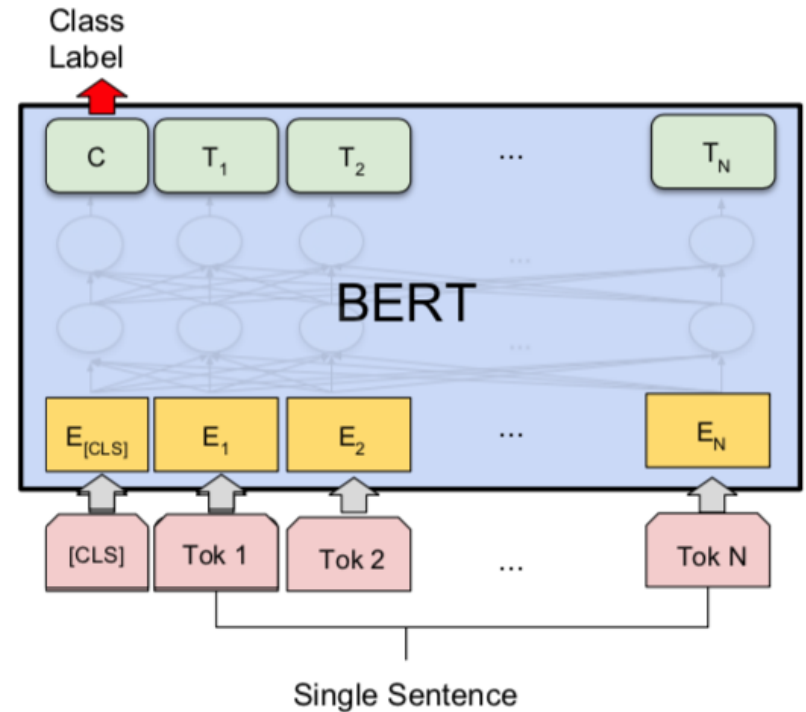
```
from keras.preprocessing.text import Tokenizer
# define 5 documents
docs = ['Well done!', 'Good work', 'Great effort', 'nice work',
        'Excellent!']
# create the tokenizer
t = Tokenizer()
# fit the tokenizer on the documents
t.fit_on_texts(docs)
print('docs:', docs)
print('word_counts:', t.word_counts)
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.  0.  1.25276297  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  1.25276297]]
```

BERT Sequence-level tasks

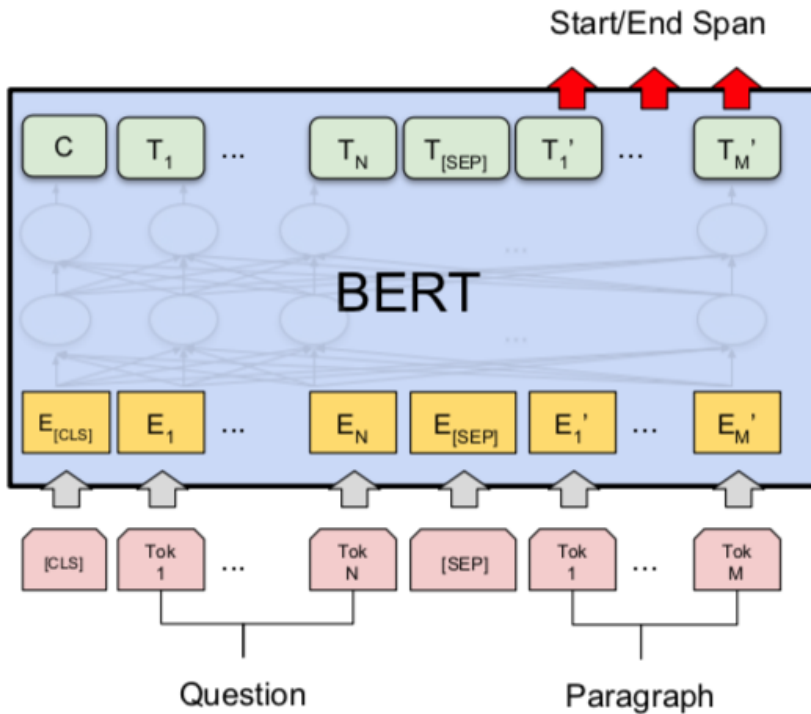


(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG

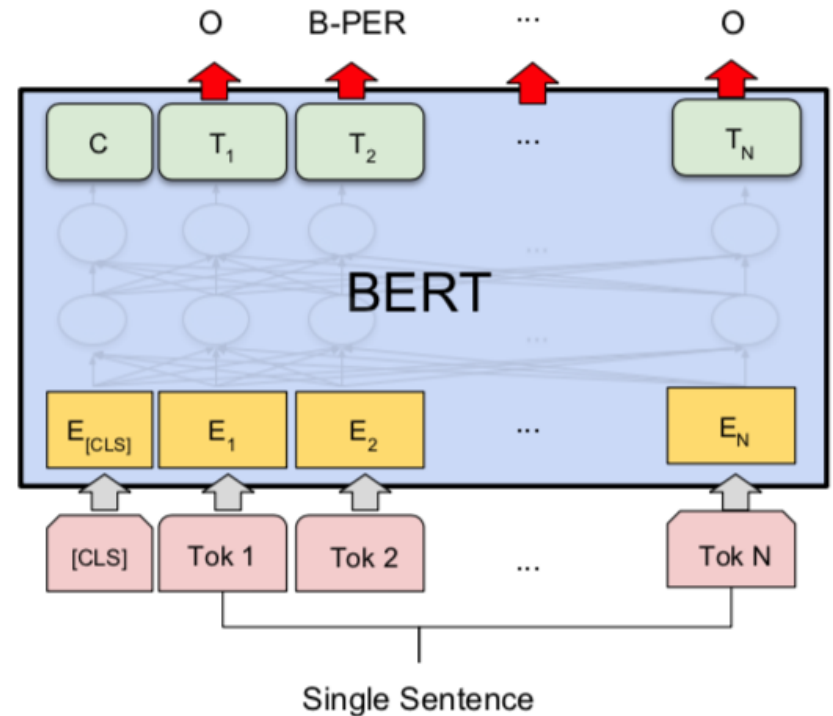


(b) Single Sentence Classification Tasks:
SST-2, CoLA

BERT Token-level tasks

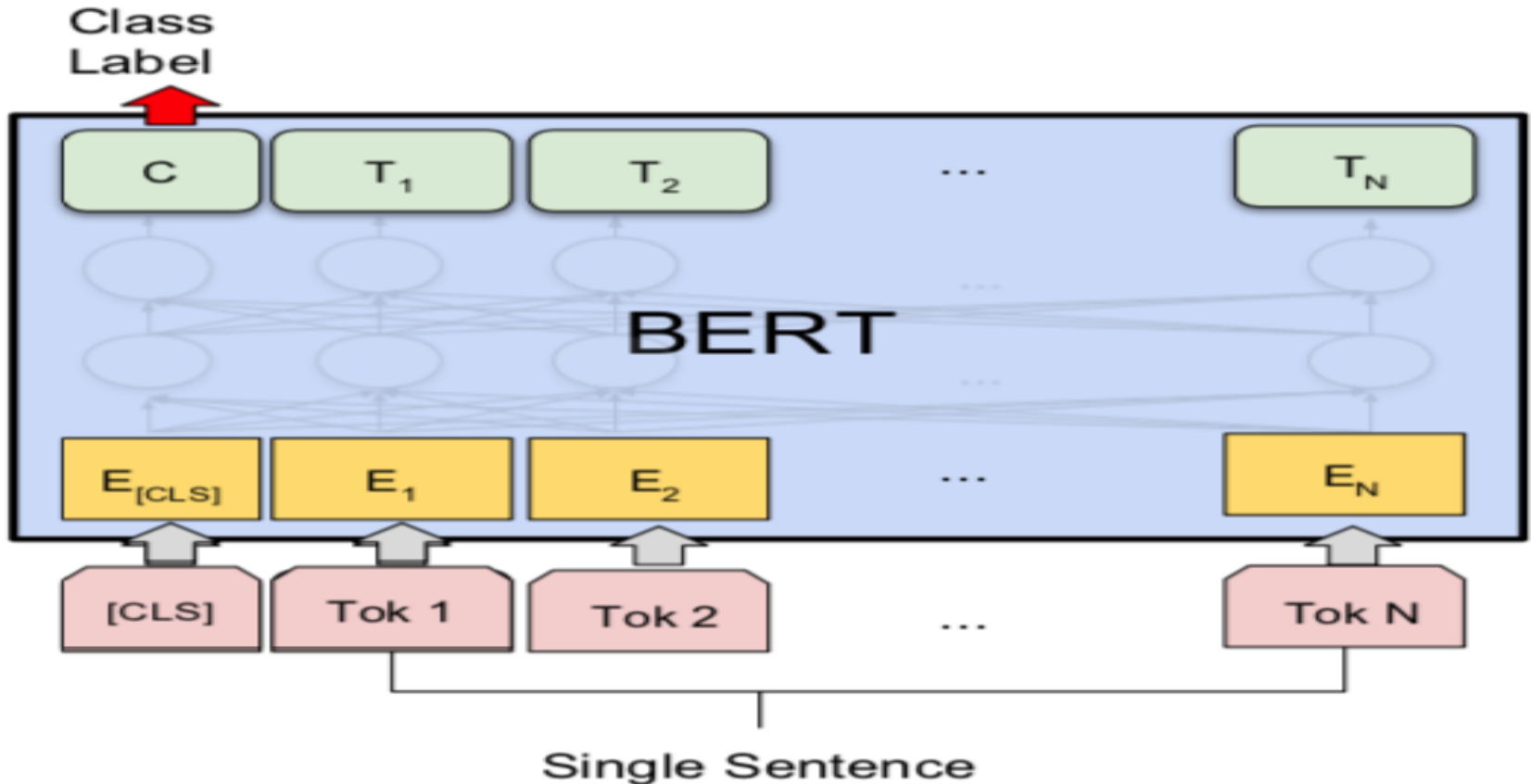


(c) Question Answering Tasks:
SQuAD v1.1



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

Sentiment Analysis: Single Sentence Classification



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

A Visual Guide to Using BERT for the First Time

(Jay Alammar, 2019)

“a visually stunning
ruminant on love”

Reviewer #1

That’s a **positive** thing to say



“reassembled from the cutting room
floor of any given daytime soap”

Reviewer #2

That’s **negative**

Sentiment Classification: SST2

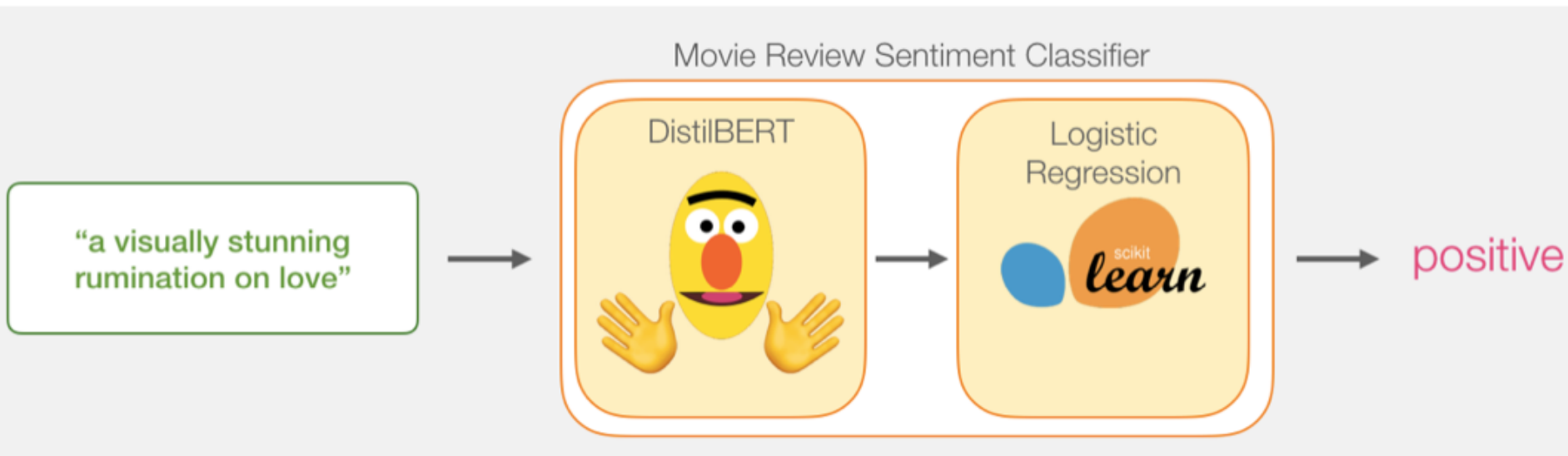
Sentences from movie reviews

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

Movie Review Sentiment Classifier



Movie Review Sentiment Classifier



Movie Review Sentiment Classifier

Model Training

Movie Review Sentiment Classifier

DistilBERT

Already (pre-)trained



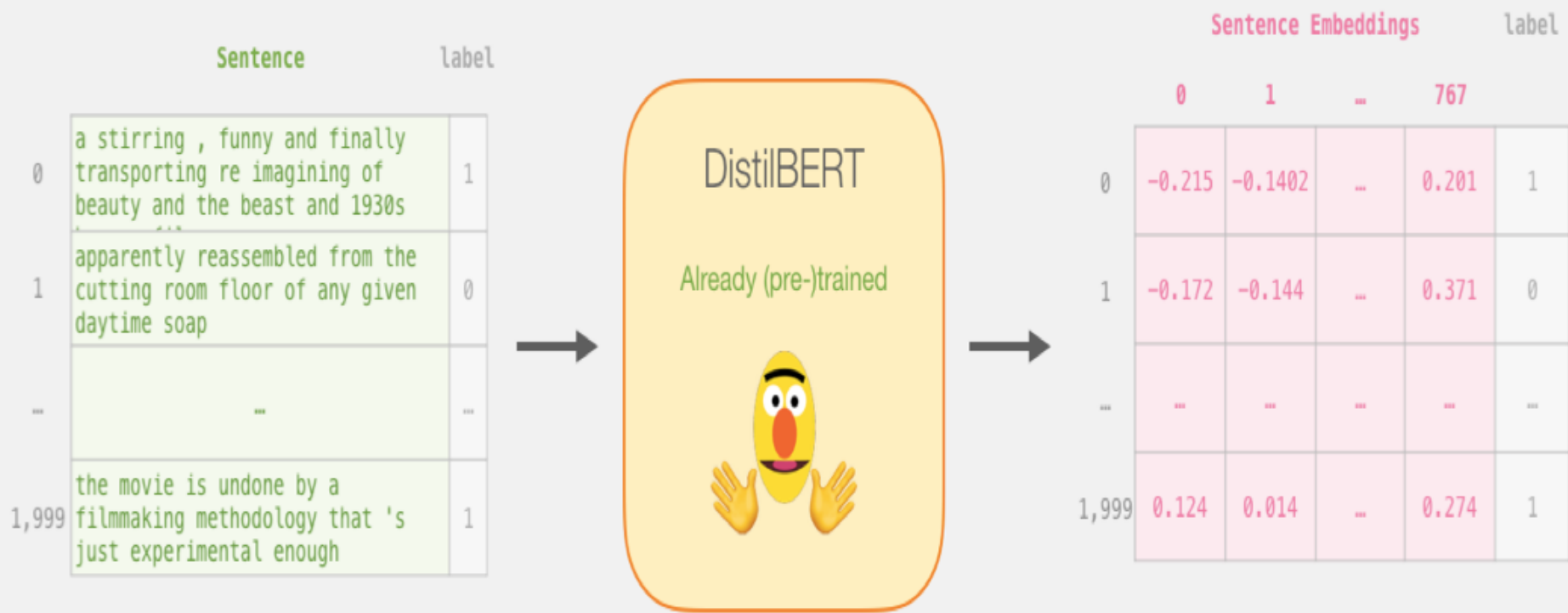
Logistic
Regression

We will train in this tutorial



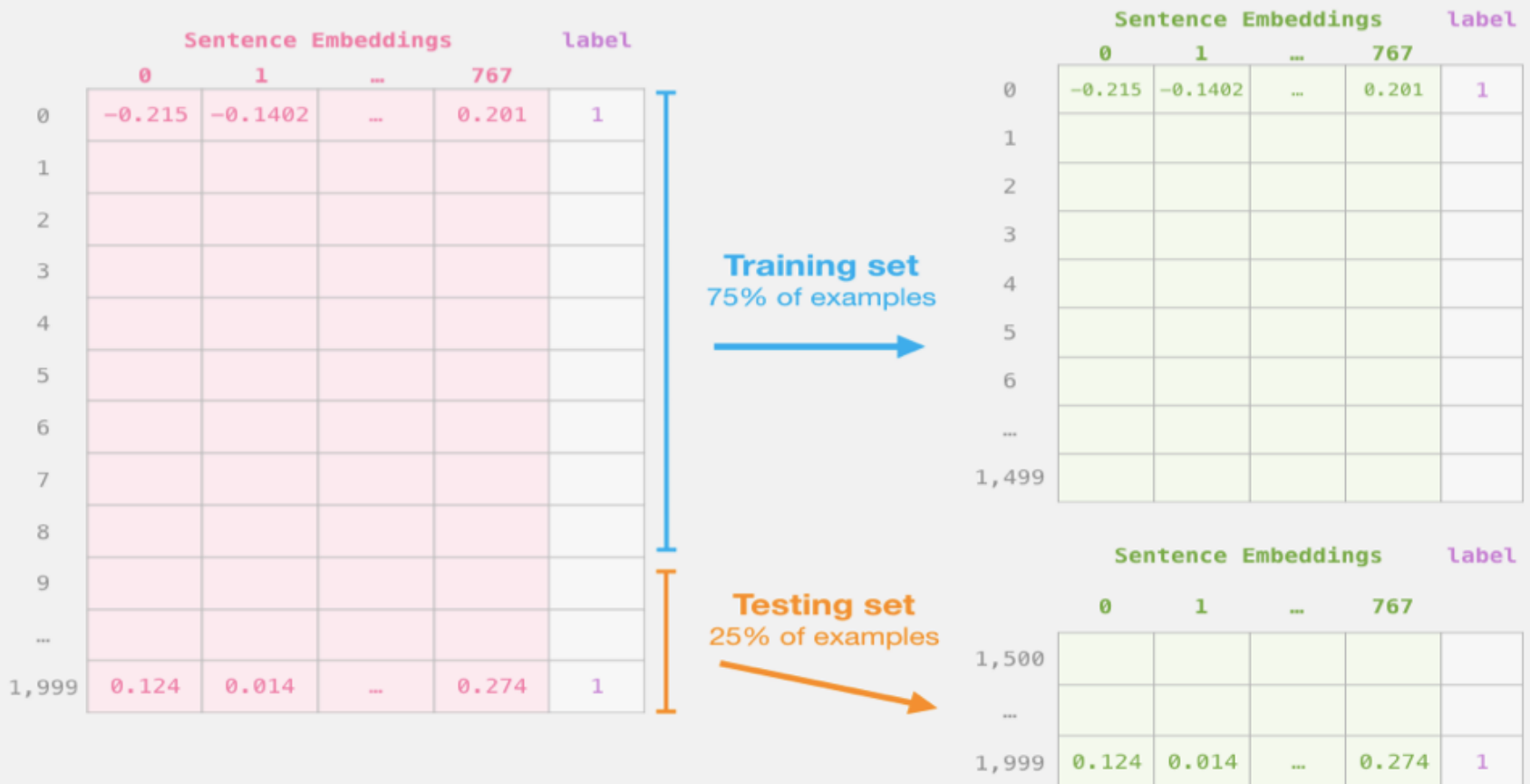
Step # 1 Use distilBERT to Generate Sentence Embeddings

Step #1: Use DistilBERT to embed all the sentences



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

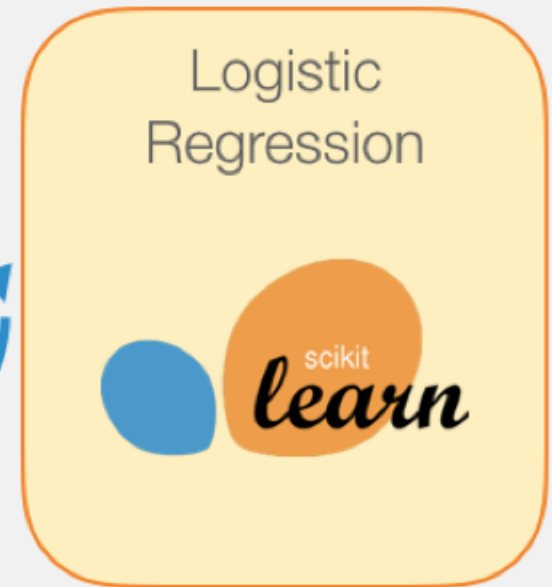
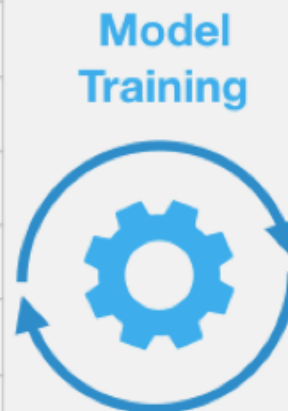
Step #2: Test/Train Split for model #2, logistic regression



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

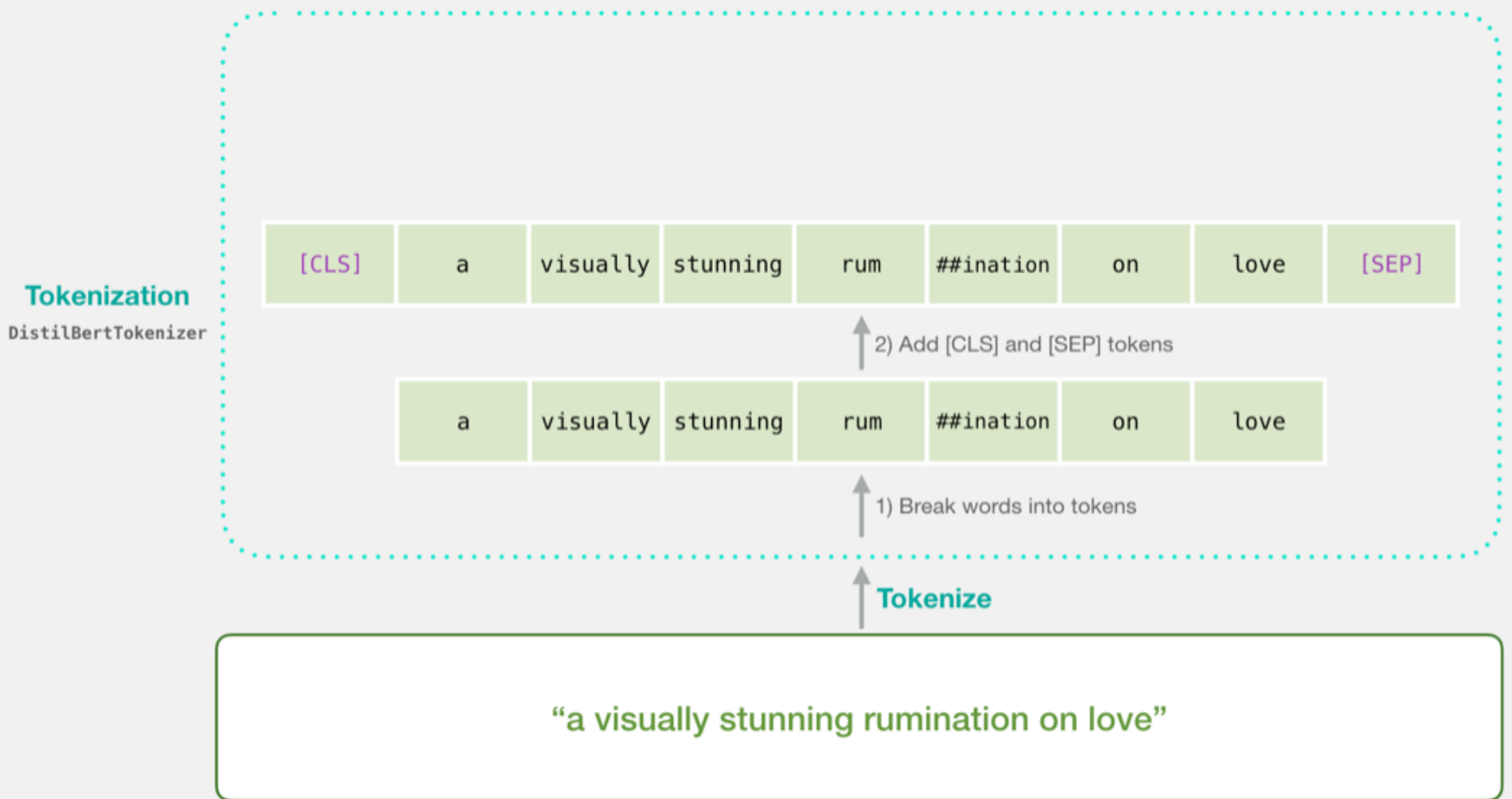
Step #3: Train the logistic regression model using the training set

	Sentence Embeddings				label
	0	1	...	767	
0	-0.215	-0.1402	...	0.201	1
1					
2					
3					
4					
5					
6					
...					
1,499					



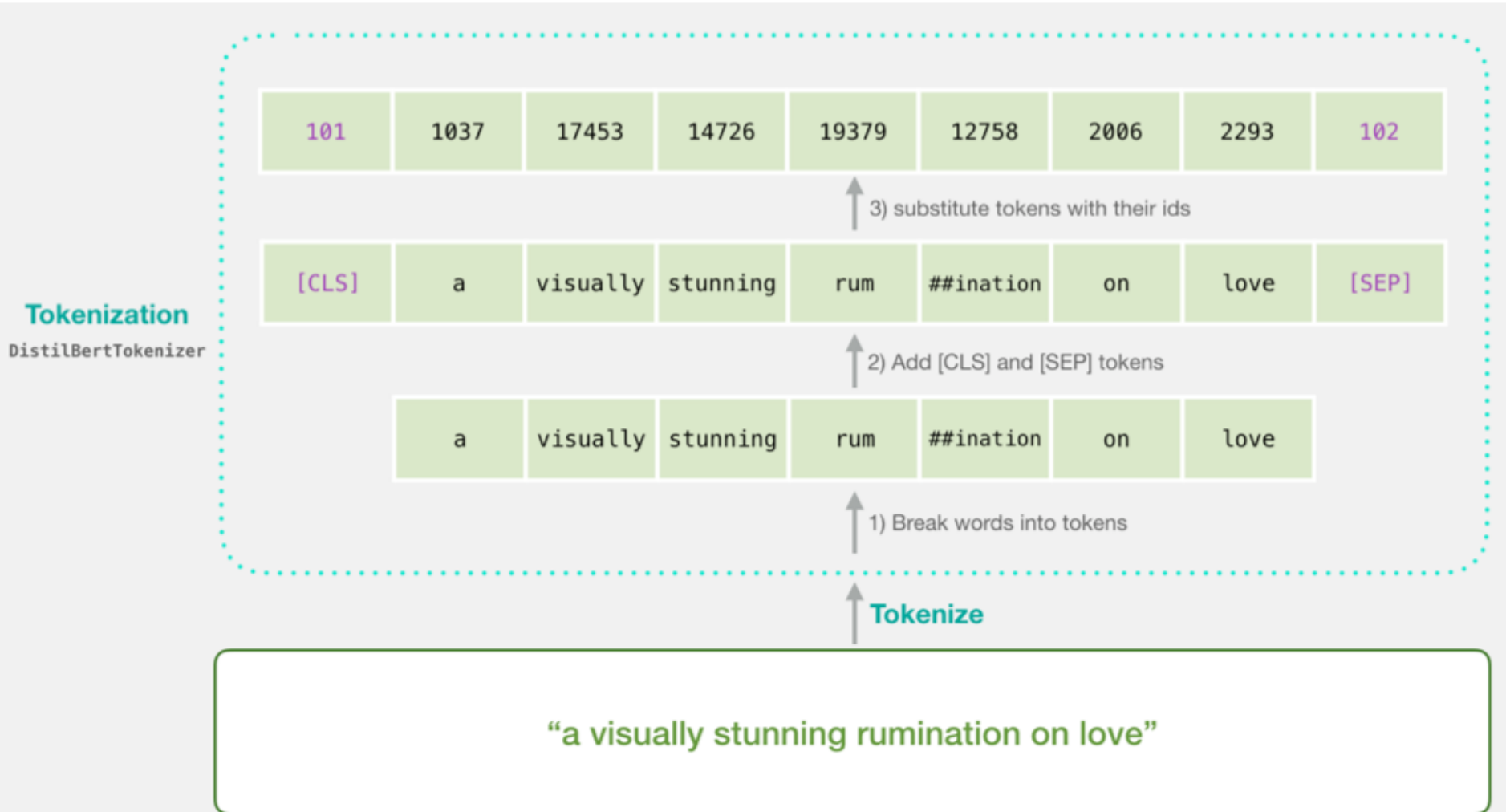
Tokenization

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

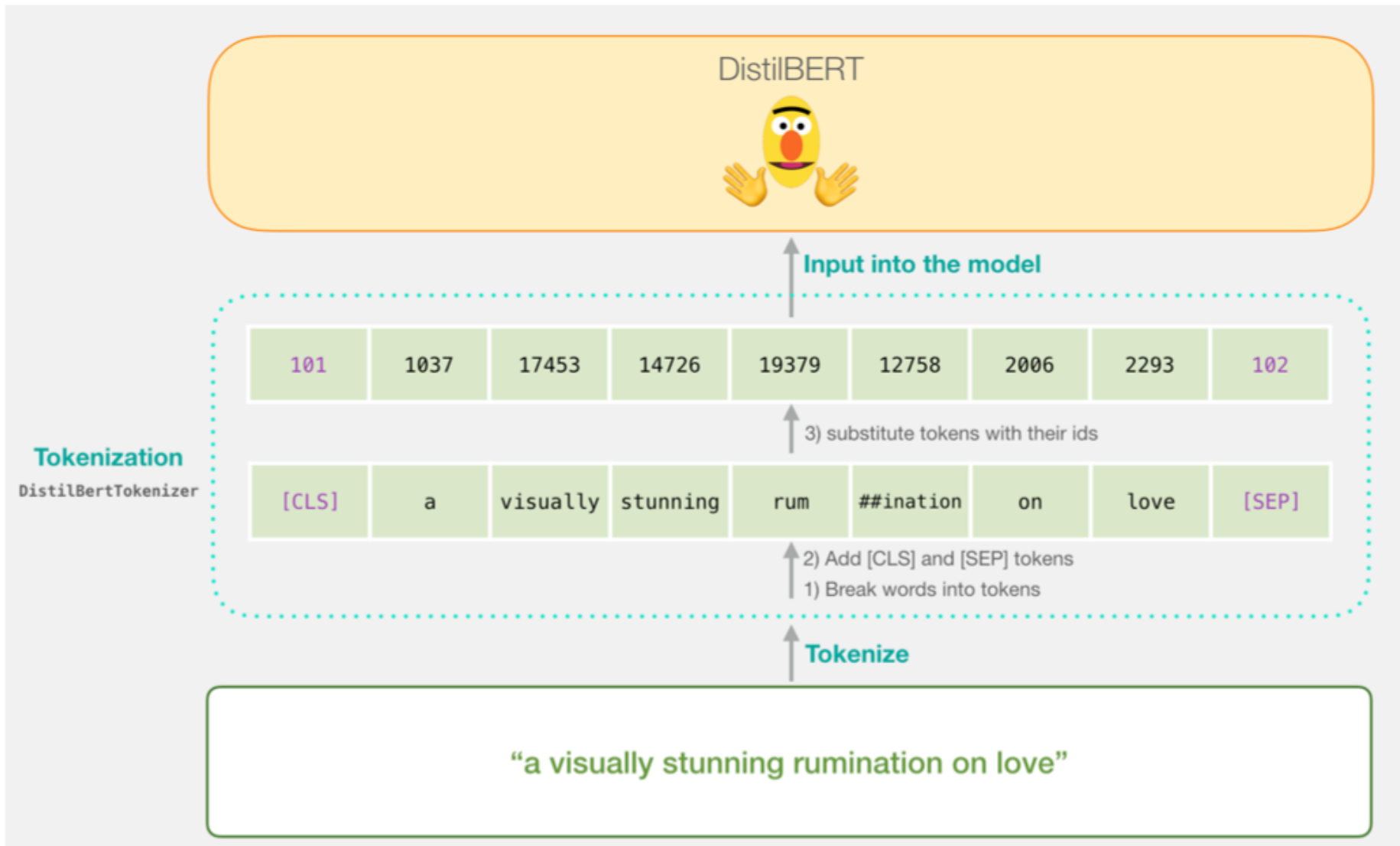


Tokenization

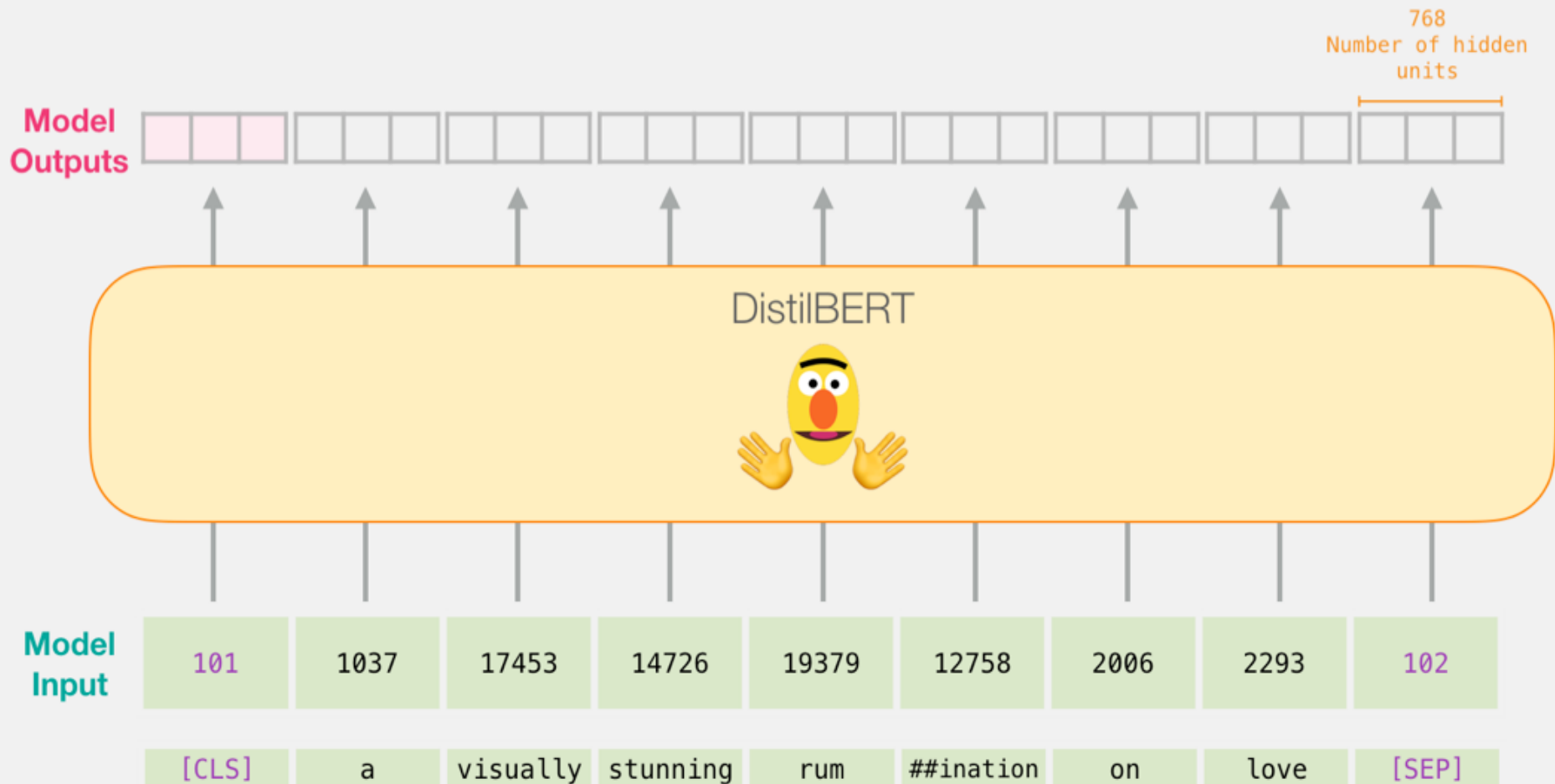
```
tokenizer.encode("a visually stunning ruminaton on love",  
                add_special_tokens=True)
```



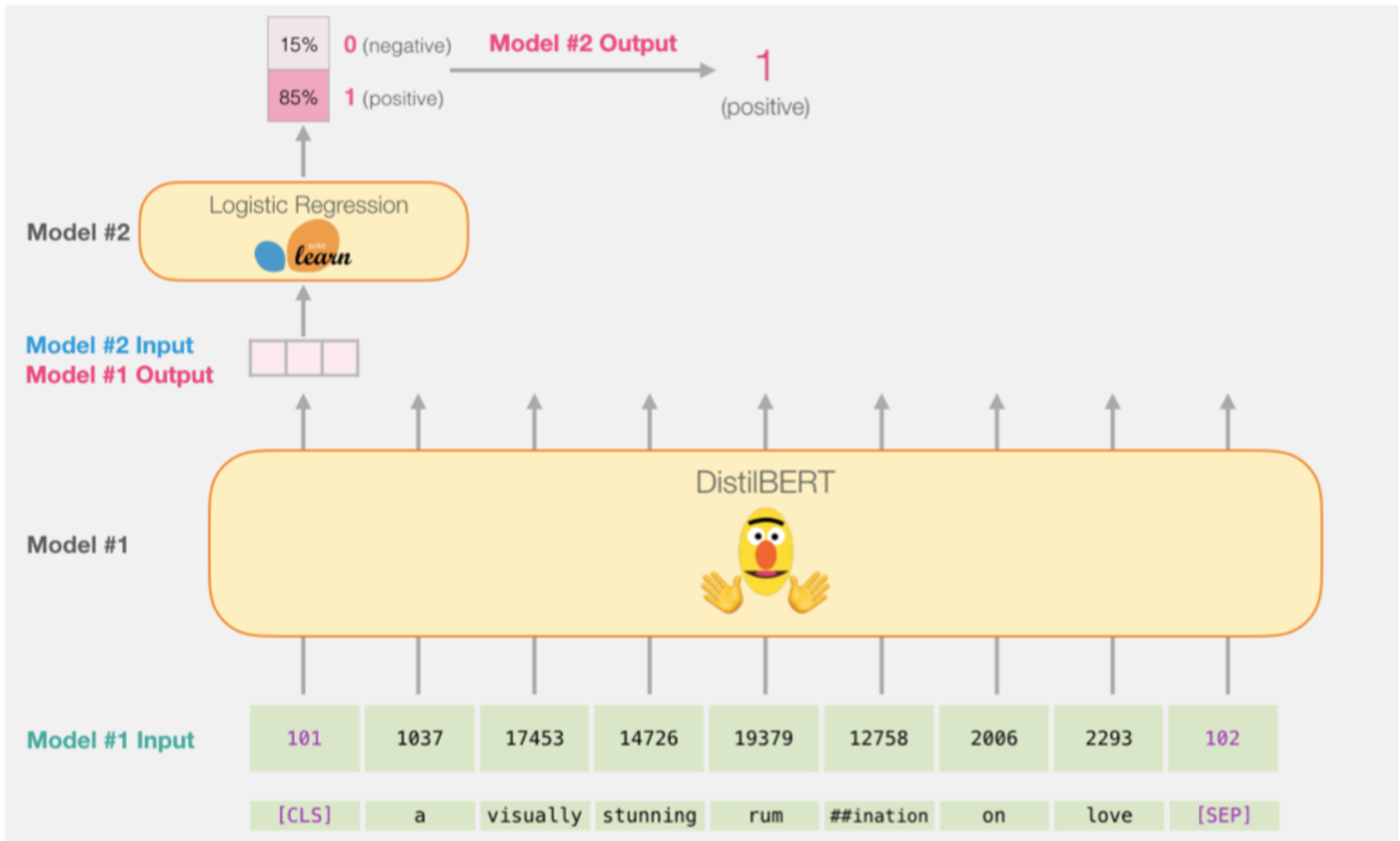
Tokenization for BERT Model



Flowing Through DistilBERT (768 features)

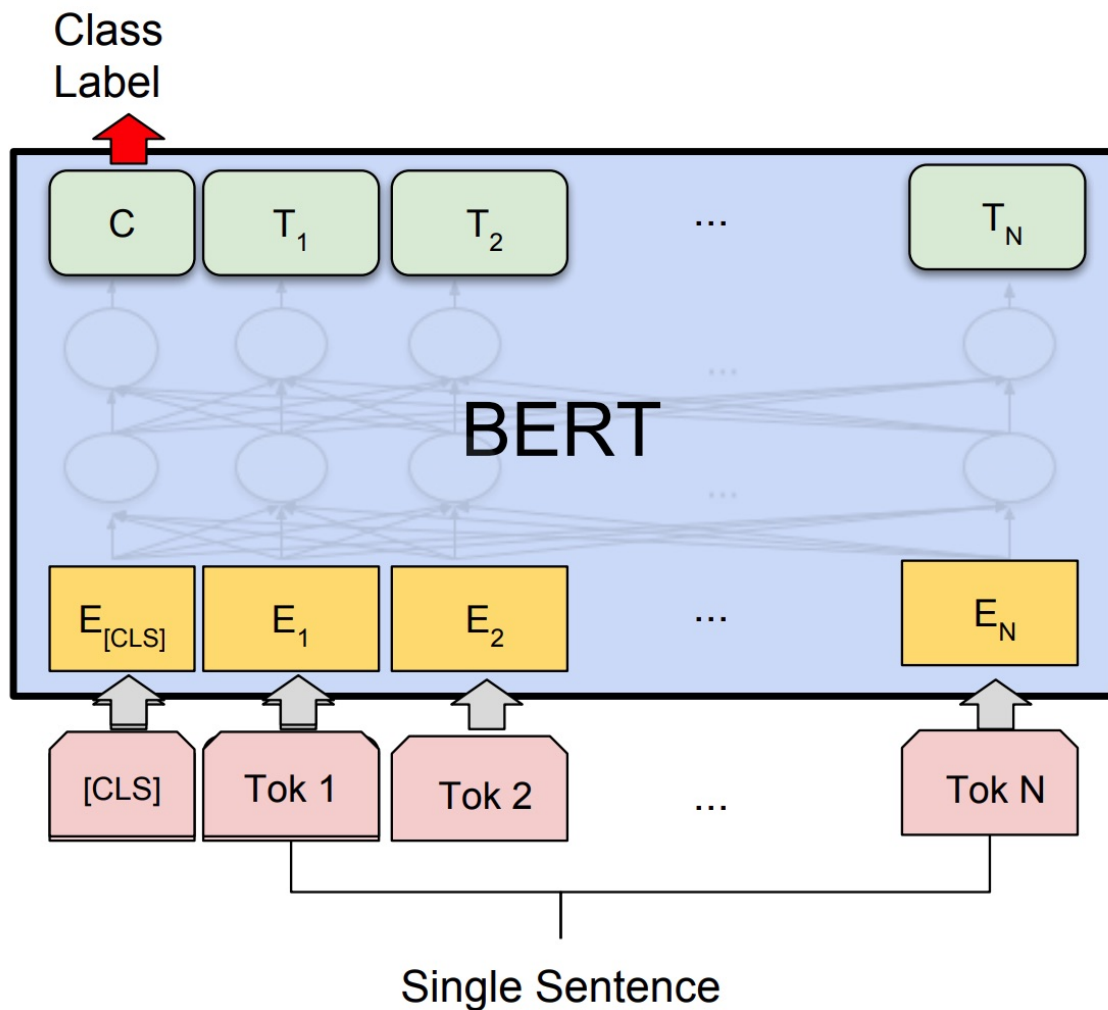


Model #1 Output Class vector as Model #2 Input



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

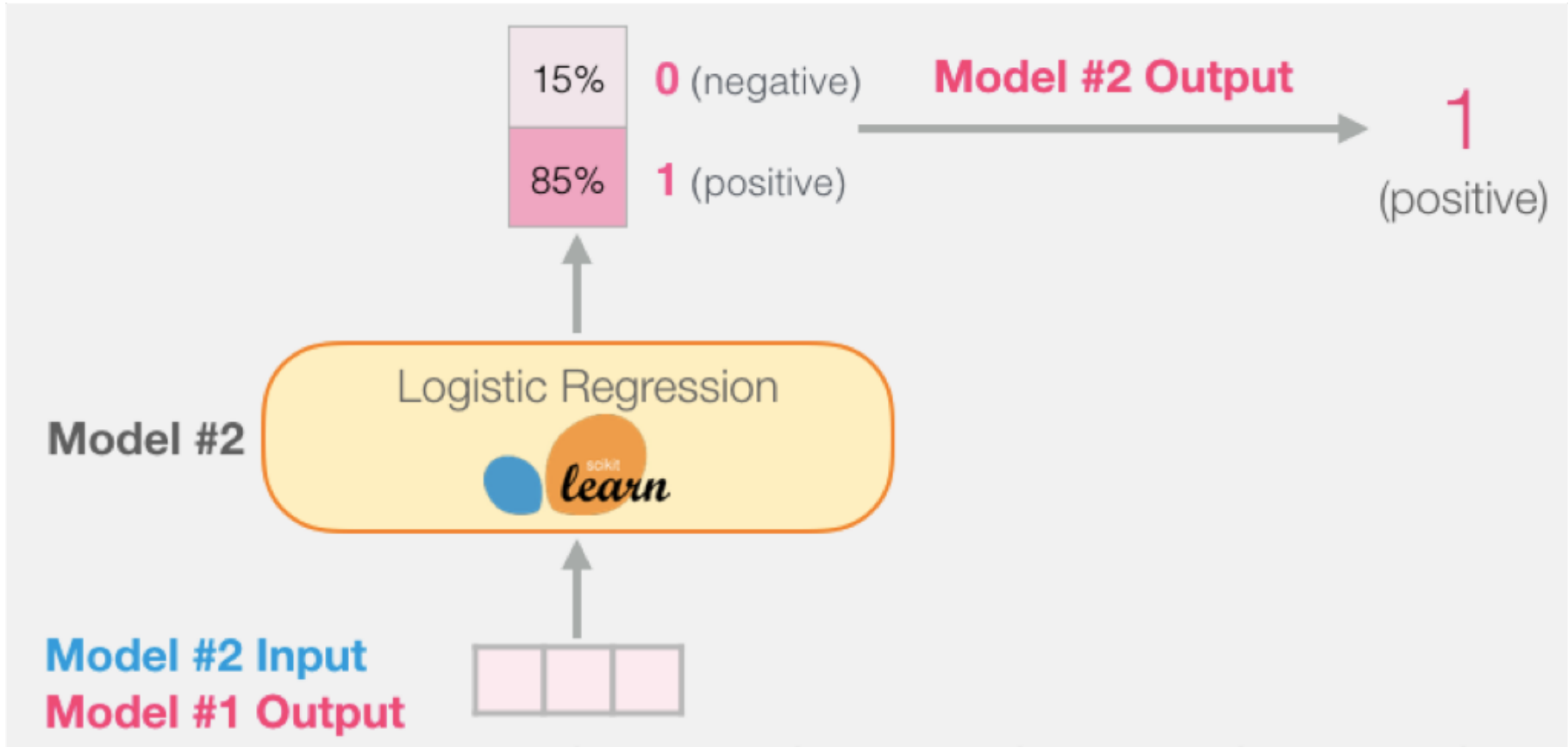
Fine-tuning BERT on Single Sentence Classification Tasks



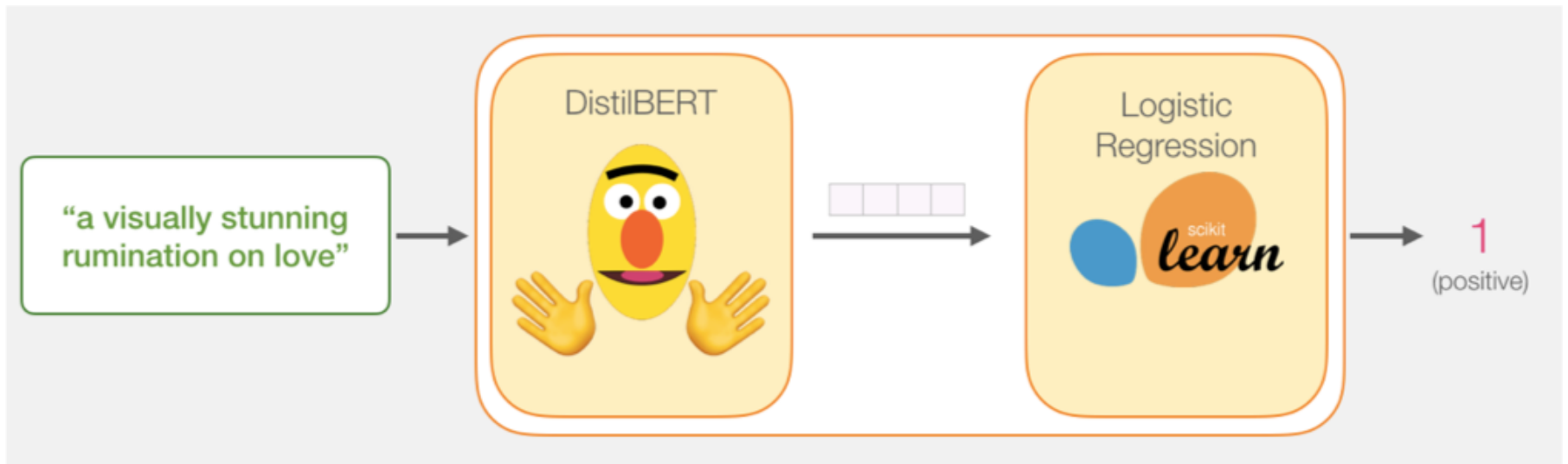
Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).

"Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

Model #1 Output Class vector as Model #2 Input



Logistic Regression Model to classify Class vector



```
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

Tokenization

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

Raw Dataset

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

Tokenize

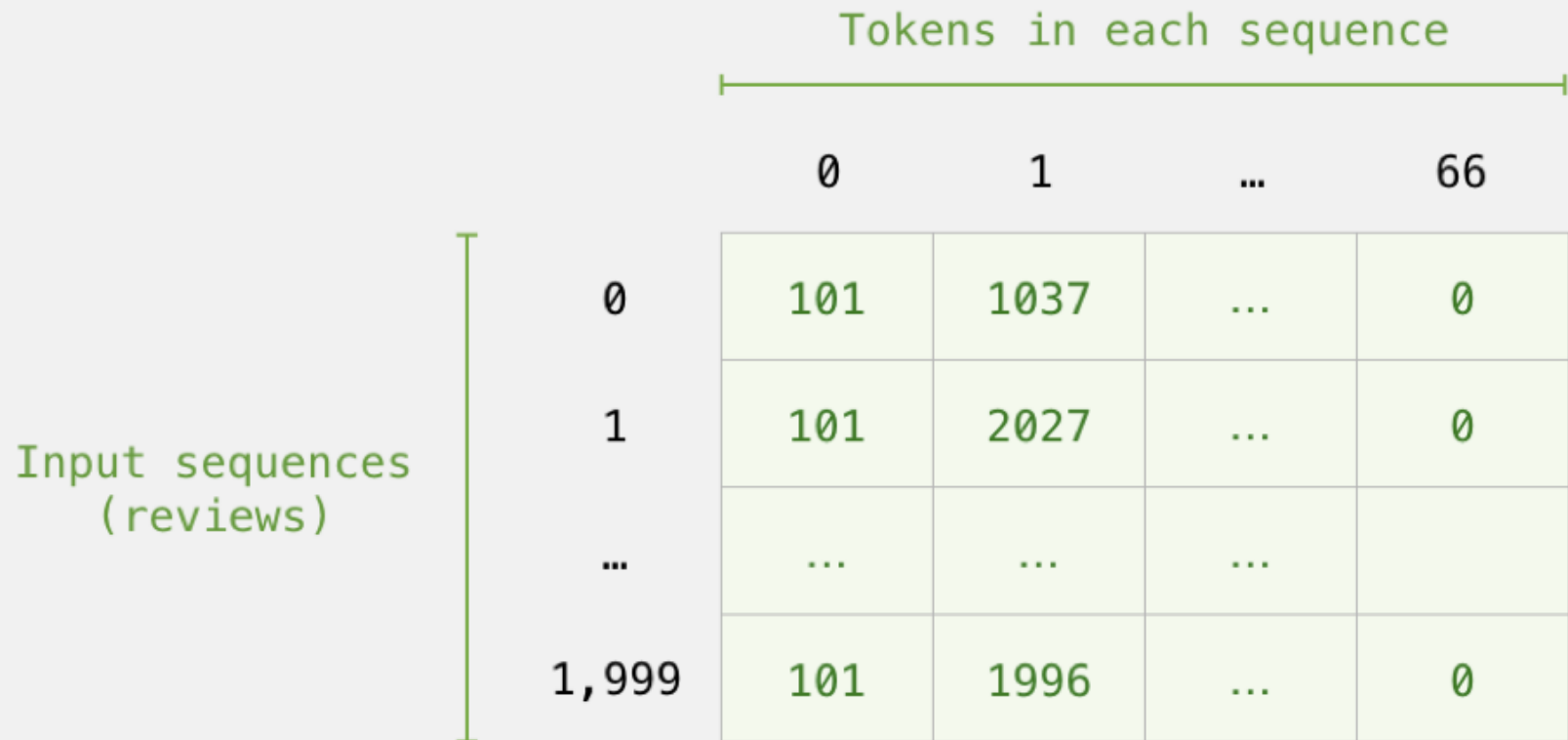


Sequences of Token IDs

```
[101, 1037, 18385, 1010, 6057, 1998, 2633, 182...  
[101, 4593, 2128, 27241, 23931, 2013, 1996, 62...  
[101, 2027, 3653, 23545, 2037, 4378, 24185, 10...  
[101, 2023, 2003, 1037, 17453, 14726, 19379, 1...  
[101, 5655, 6262, 1005, 1055, 12075, 2571, 376...
```

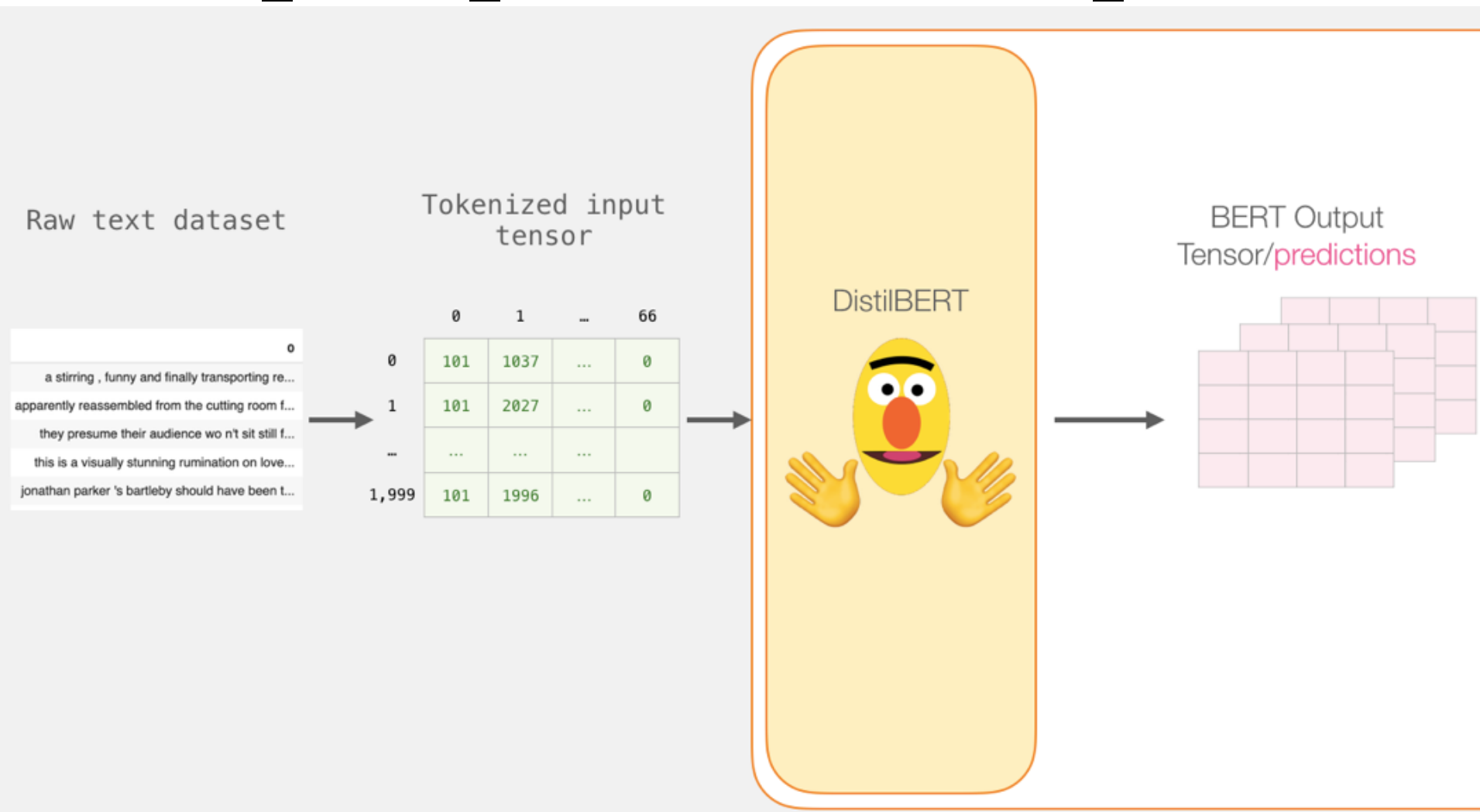
BERT Input Tensor

BERT/DistilBERT Input Tensor

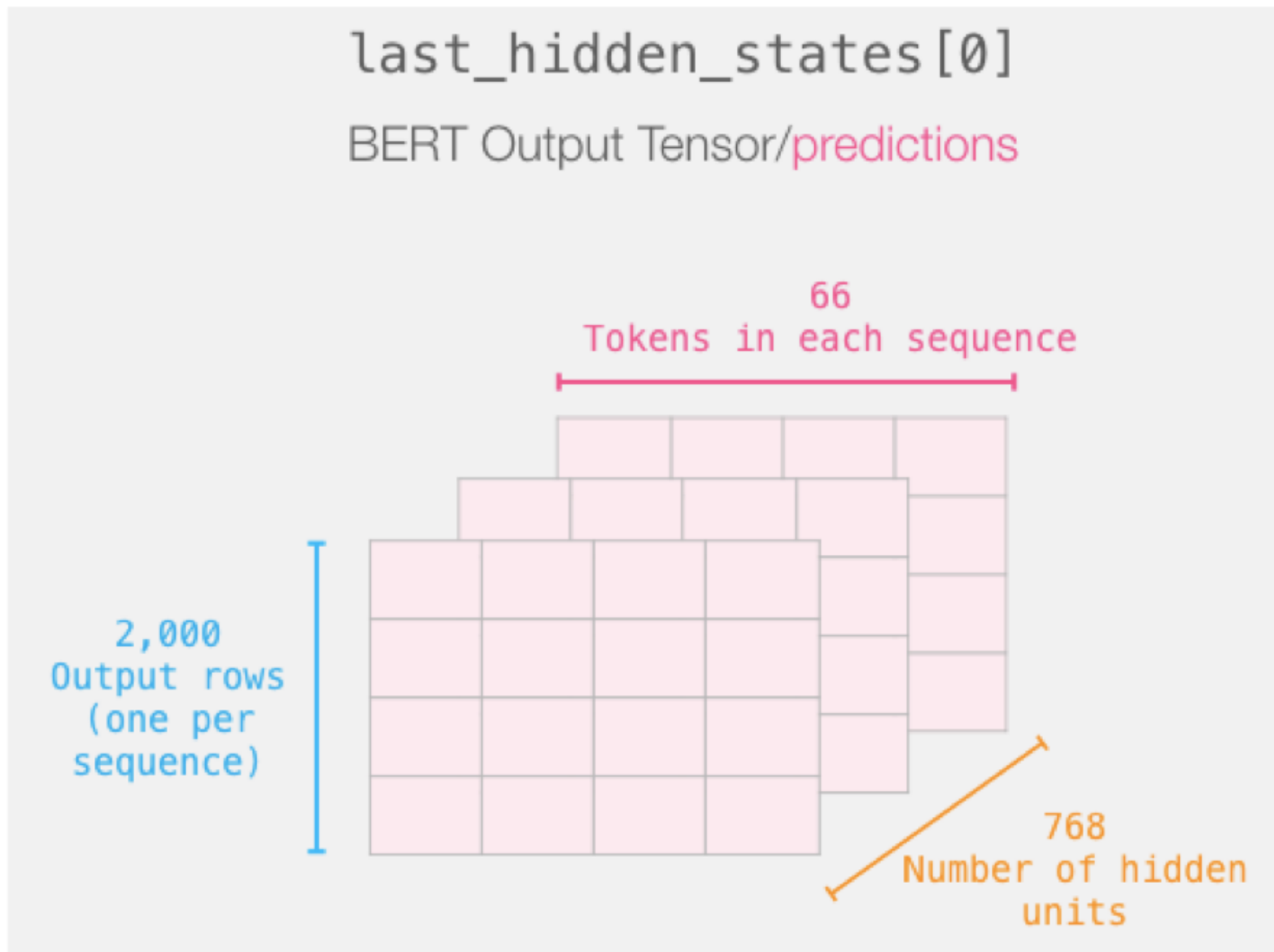


Processing with DistilBERT

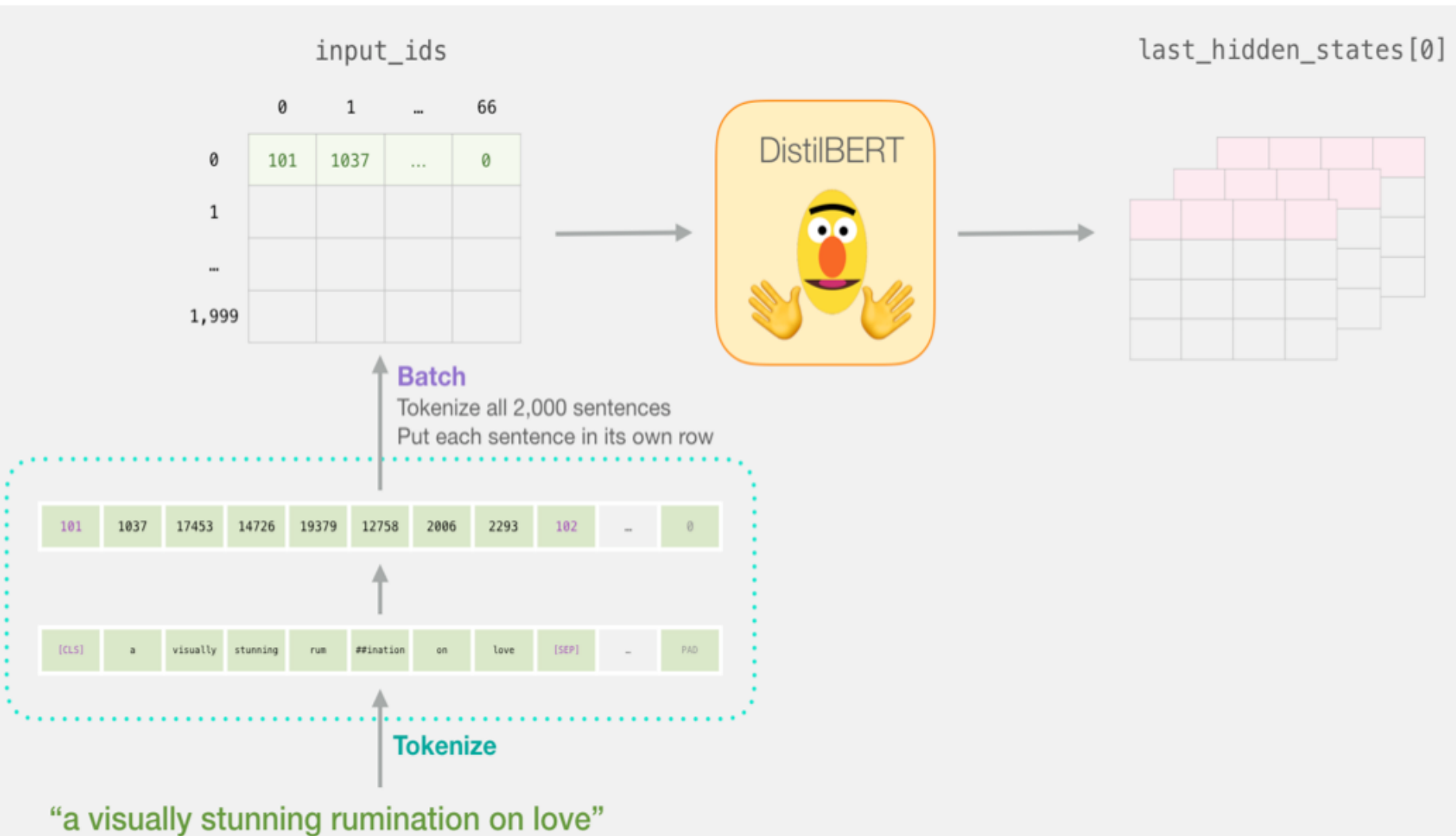
```
input_ids = torch.tensor(np.array(padded))  
last_hidden_states = model(input_ids)
```



Unpacking the BERT output tensor



Sentence to last_hidden_state[0]

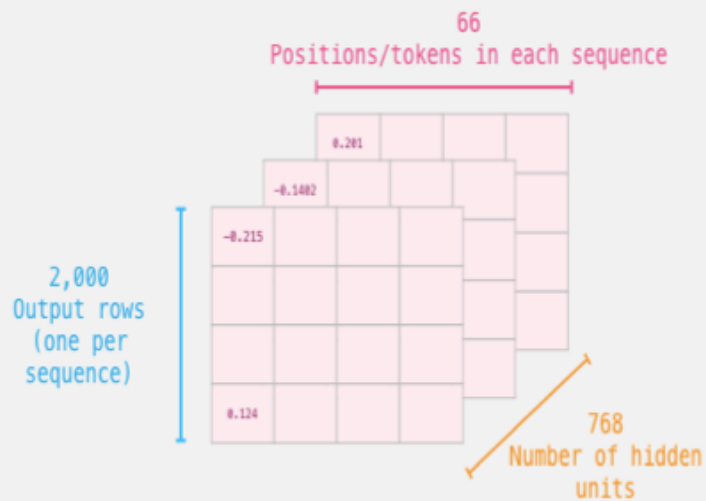


BERT's output for the [CLS] tokens

Slice the output for the first position for all the sequences, take all hidden unit outputs

```
features = last_hidden_states[0][:, 0, :].numpy()
```

`last_hidden_states[0]`
BERT Output Tensor/predictions

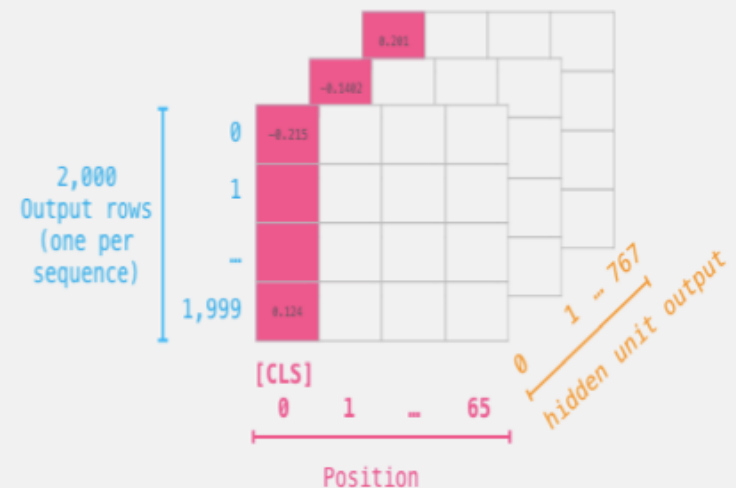


only the first position: [CLS]

`last_hidden_states[0][:, 0, :]`

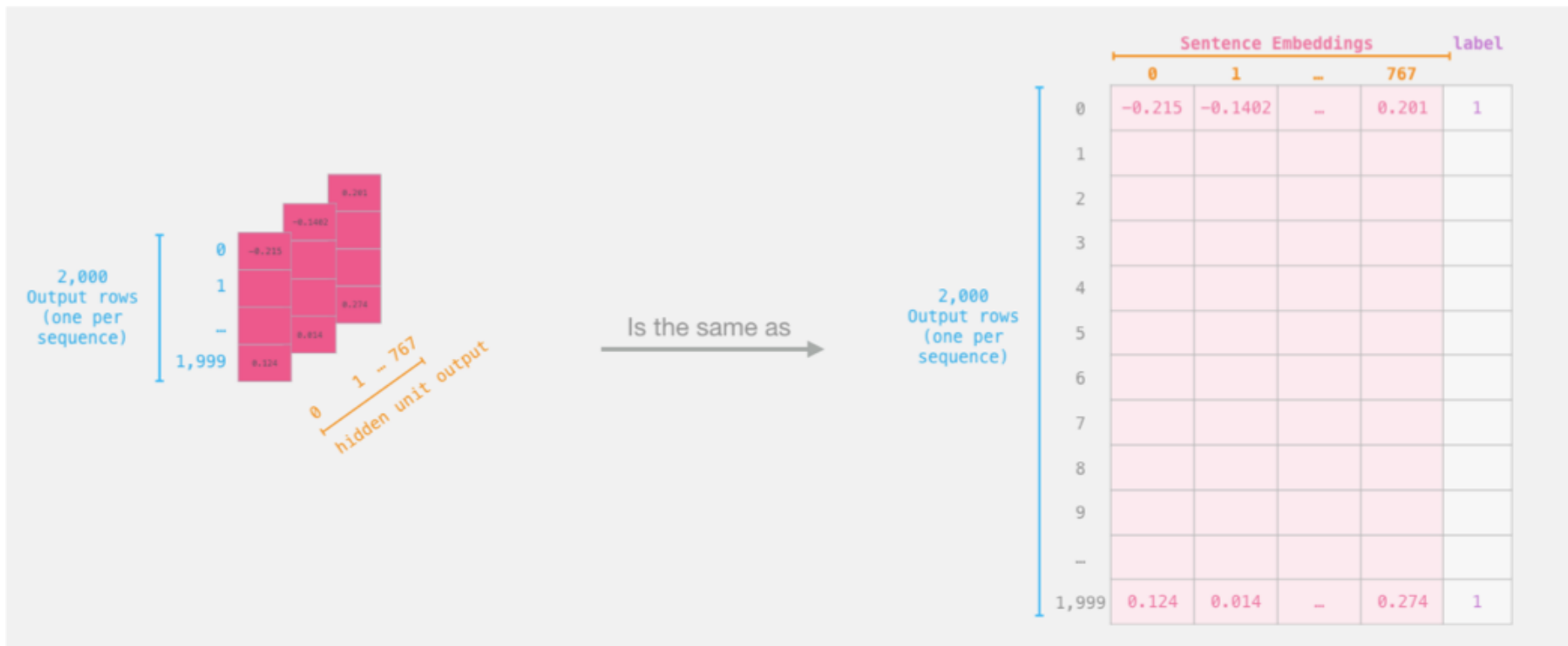
all sentences

all hidden unit outputs



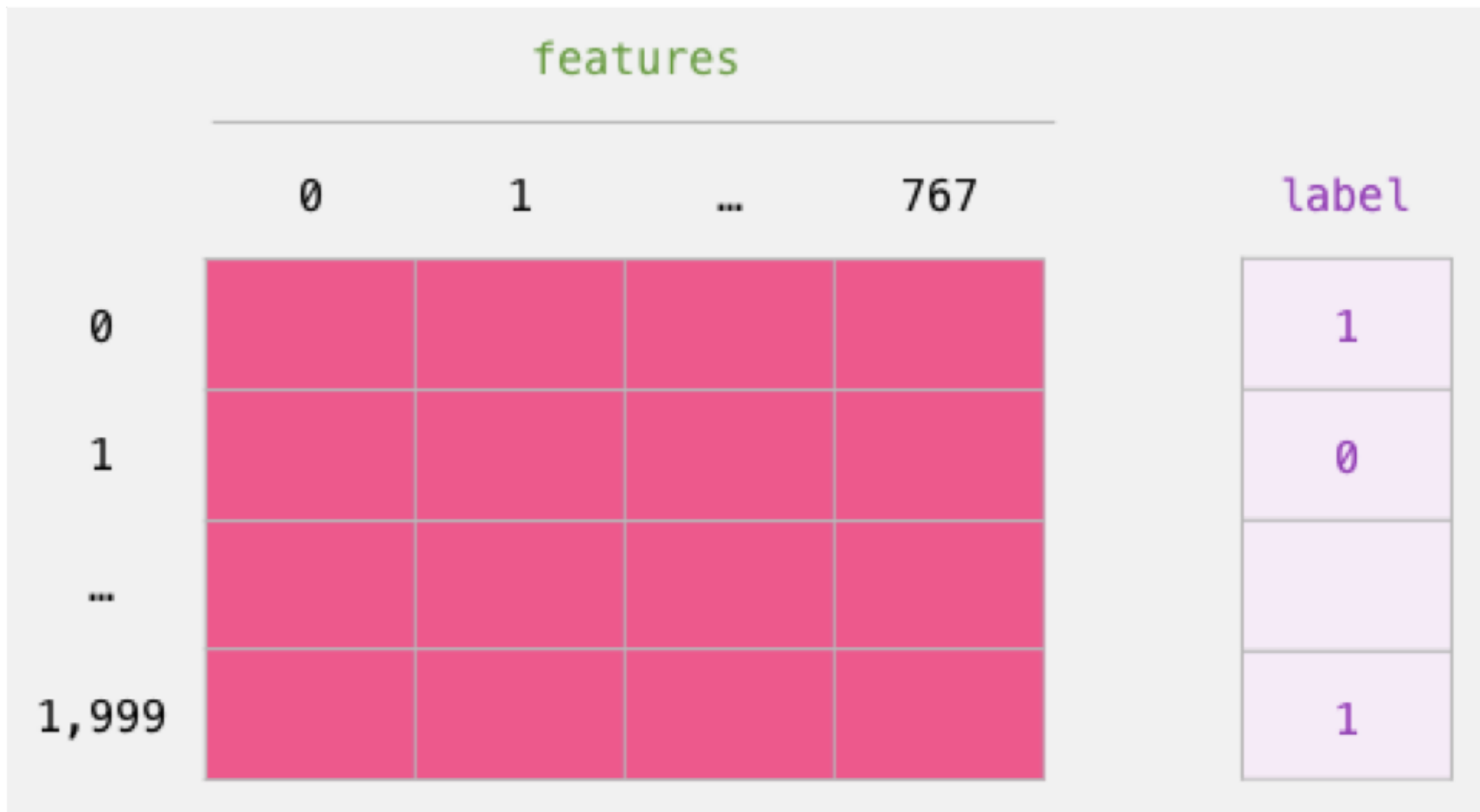
The tensor sliced from BERT's output

Sentence Embeddings



Dataset for Logistic Regression (768 Features)

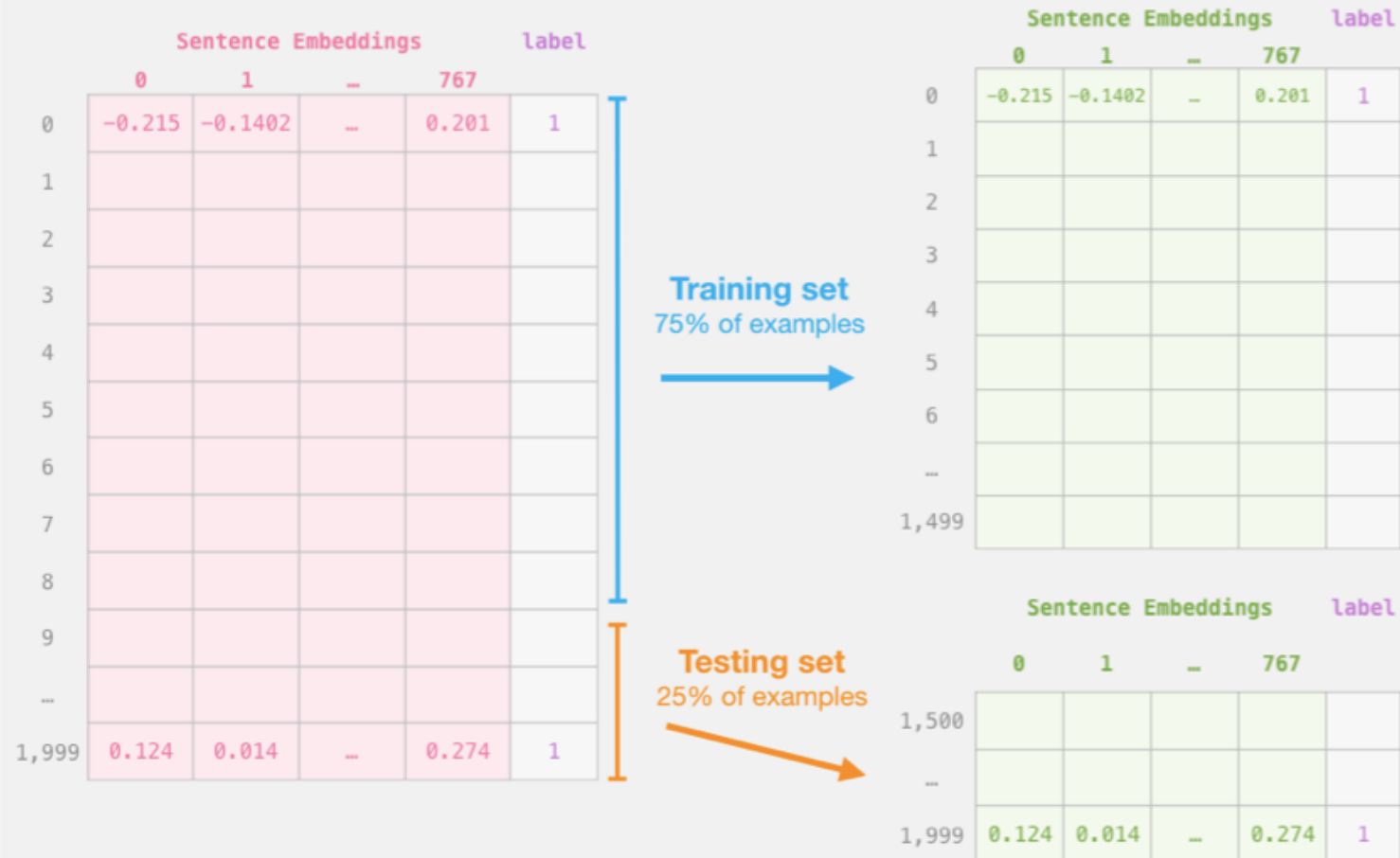
The features are the output vectors of BERT for the [CLS] token (position #0)



```
labels = df[1]
```

```
train_features, test_features, train_labels, test_labels =  
train_test_split(features, labels)
```

Step #2: Test/Train Split for model #2, logistic regression



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%
```

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

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

“a visually stunning
rumination on love”

Reviewer #1

That’s a **positive** thing to say

“reassembled from the cutting room
floor of any given daytime soap”

Reviewer #2

That’s **negative**

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

The screenshot shows the TensorFlow website interface. At the top, there is a navigation bar with the TensorFlow logo, links for 'Install', 'Learn', 'API', 'Resources', and 'More', a search bar, and language selection ('English'). Below this is a breadcrumb trail: 'TensorFlow Core > Overview > Tutorials > Guide > TF 1'. The left sidebar contains a list of tutorials, with 'Text classification with preprocessed text' highlighted under the 'BEGINNER' section. The main content area features the title 'Text classification with preprocessed text: Movie reviews' with a five-star rating. Below the title are three buttons: 'Run in Google Colab', 'View source on GitHub', and 'Download notebook'. The text describes the notebook's purpose: classifying movie reviews as positive or negative using the IMDB dataset. It mentions that the dataset is split into 25,000 training and 25,000 testing reviews, which are balanced. The notebook uses the `tf.keras` API. A right-hand 'Contents' sidebar lists the steps: Setup, Download the IMDB dataset, Try the encoder, Explore the data, Prepare the data for training, Build the model, Hidden units, Loss function and optimizer, Train the model, Evaluate the model, and Create a graph of accuracy and loss over time.

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

Table of contents

- Leveraging gensim for building a FastText model
- Text Classification**
 - Text Classification: IMDB Movie Reviews
 - Download the IMDB dataset
 - Explore the data
 - Prepare the data for training
 - Build the model
 - Train the model
 - Evaluate the model
 - Create a graph of accuracy and loss over time
 - Text Classification: BBC News Articles
- Python Programming
- OS, IO, files, and Google Drive
- Python Numpy
- Python Pandas
- Section

+ Code + Text RAM Disk Editing

Text Classification

- 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
- 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>

Text Classification: IMDB Movie Reviews

Source: François Chollet (2017), Text classification with preprocessed text: Movie reviews, https://www.tensorflow.org/tutorials/keras/text_classification

```
[25] 1 !pip install tf-nightly
      2 import tensorflow as tf
      3 print(tf.__version__)
```

```
Collecting tf-nightly
  Downloading https://files.pythonhosted.org/packages/2a/a0/7381cd278a8e1a9235f032ea811af07bbe31ed45ac9781f2...
  517.6MB 24kB/s
Collecting tf-estimator-nightly
  Downloading https://files.pythonhosted.org/packages/0f/fb/984408ab3aee0bddfc02e1136a4fd76c4e58fd128c458e20...
  460kB 40.2MB/s
Requirement already satisfied: google-pasta>=0.1.8 in /usr/local/lib/python3.6/dist-packages (from tf-nightly)
```

<https://tinyurl.com/imtkupython101>

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 navigation bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help" menus. A "Table of contents" sidebar on the left lists various topics, with "Sentiment Analysis" highlighted. The main workspace shows two code cells. The first cell, labeled [2], contains a shell command to download a dataset. The second cell, labeled [3], contains Python code to load the dataset into a pandas DataFrame. The output of the second cell shows the DataFrame's structure, including the number of entries and the data types of the columns.

python101.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

RAM Disk Editing

Table of contents

- Topic Modeling with Gensim LDA model
- Topic Modeling with Scikit-learn LDA and NMF
- Topic Modeling Visualization
- Text Similarity and Clustering
 - Text Similarity
 - Text Clustering
- Semantic Analysis and Named Entity Recognition (NER)
 - Semantic Analysis
 - Named Entity Recognition (NER)
- Sentiment Analysis**
 - Sentiment Analysis - Unsupervised Lexical
 - Sentiment Analysis - Supervised Machine Learning
 - Sentiment Analysis - Supervised Deep Learning Models
 - Sentiment Analysis - Advanced Deep Learning
- Data Visualization

+ Code + Text

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

```
[3] 1 import numpy as np
    2 import pandas as pd
    3 import tensorflow as tf
    4 import tensorflow_hub as hub
    5
    6 df = pd.read_csv('http://mail.tku.edu.tw/myday/data/example/movie_reviews.csv')
    7 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

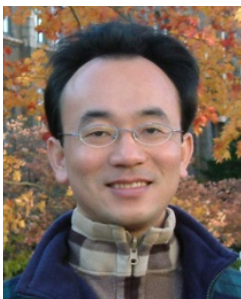
- Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson.
- Dipanjan Sarkar (2019), Text Analytics with Python: A Practitioner's Guide to Natural Language Processing, Second Edition. APress.
- Benjamin 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 Alamar (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>

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

