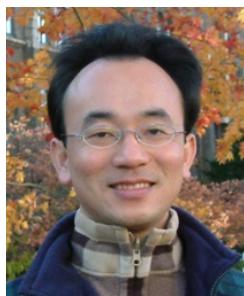


人工智慧文本分析 基礎與應用

(Artificial Intelligence for Text Analytics: Foundations and Applications)



Min-Yuh Day
戴敏育

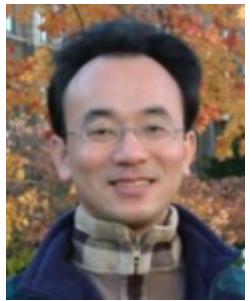
Associate Professor
副教授

Institute of Information Management, National Taipei University
國立臺北大學 資訊管理研究所

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

2020-09-26





戴敏育 博士 (Min-Yuh Day, Ph.D.)

國立台北大學 資訊管理研究所 副教授

中央研究院 資訊科學研究所 訪問學人

國立台灣大學 資訊管理 博士

Publications Co-Chairs, IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013-)

Program Co-Chair, IEEE International Workshop on Empirical Methods for Recognizing Inference in TExt (IEEE EM-RITE 2012-)

Publications Chair, The IEEE International Conference on Information Reuse and Integration (IEEE IRI)



國立臺北大學
National Taipei University



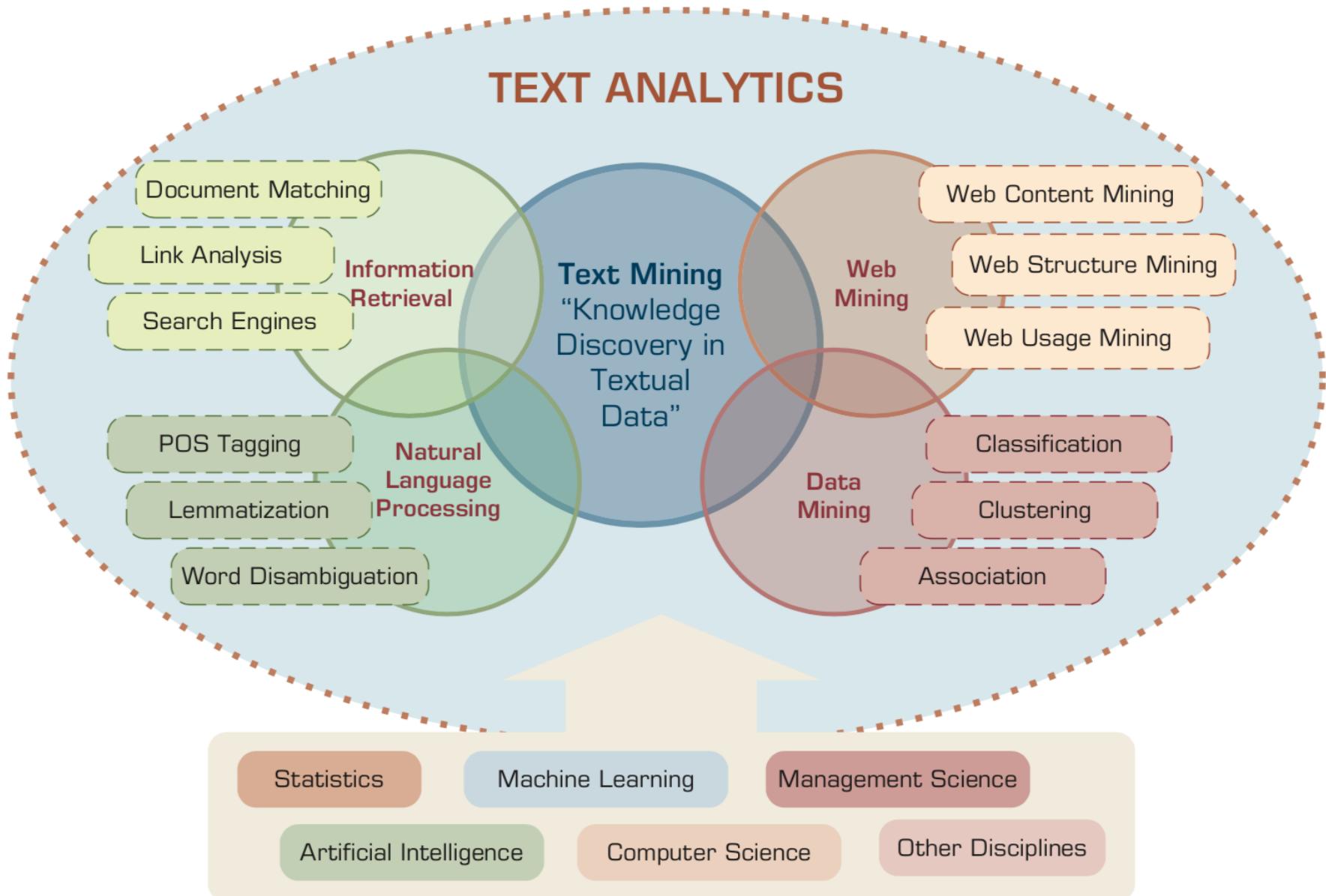
Topics

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. 人工智能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

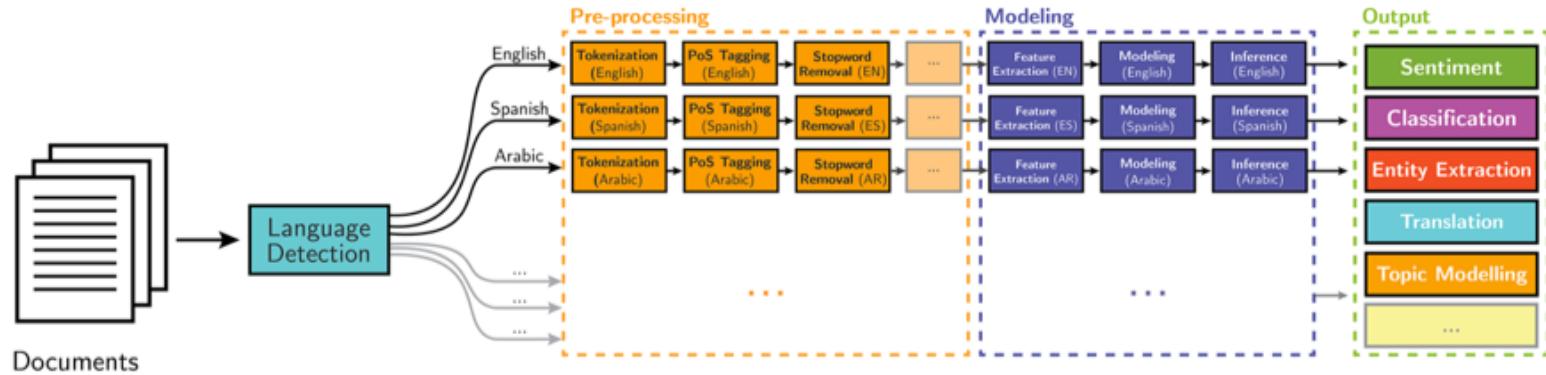
- AI for Text Analytics: Foundations
 - Processing and Understanding Text
- AI for Text Analytics: Application
 - Sentiment Analysis
 - Text classification

Text Analytics and Text Mining

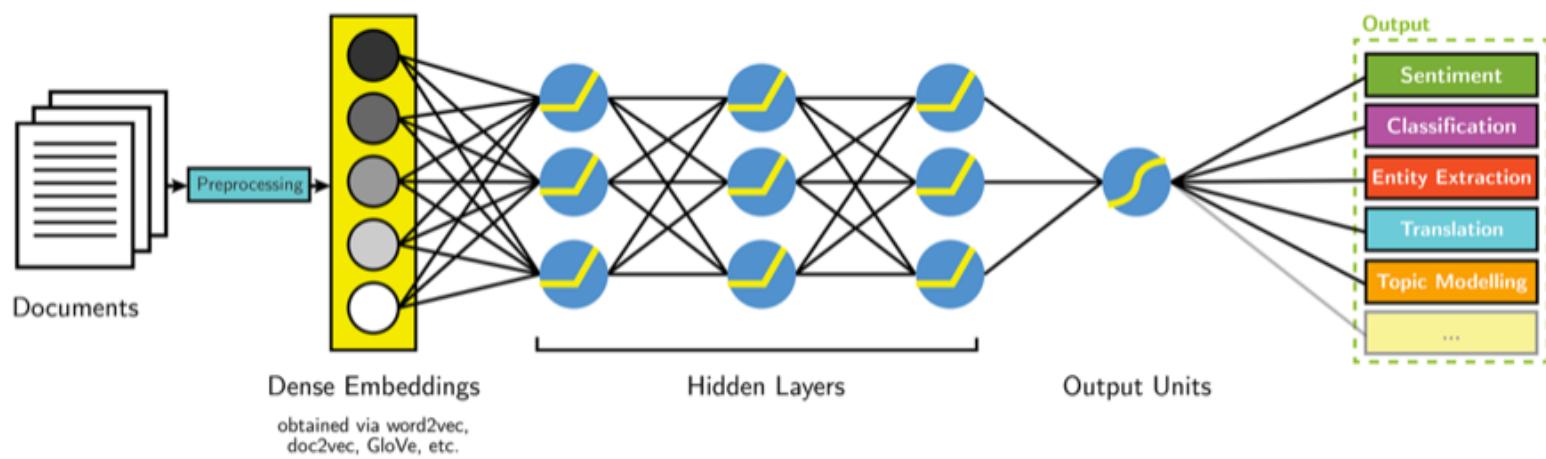


NLP

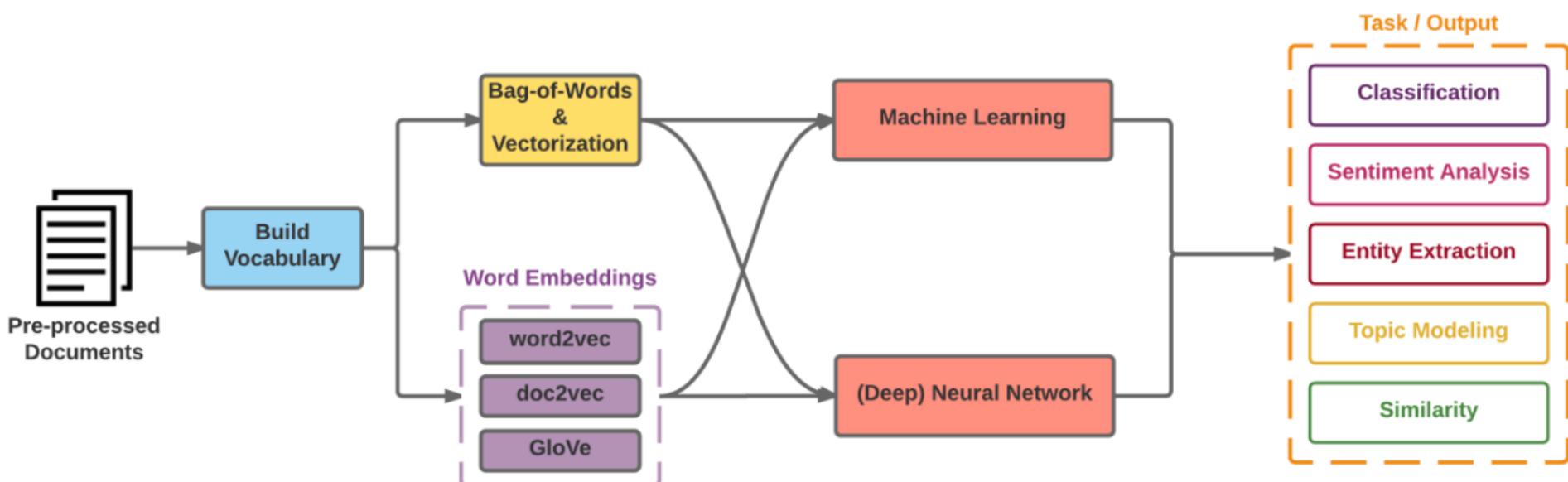
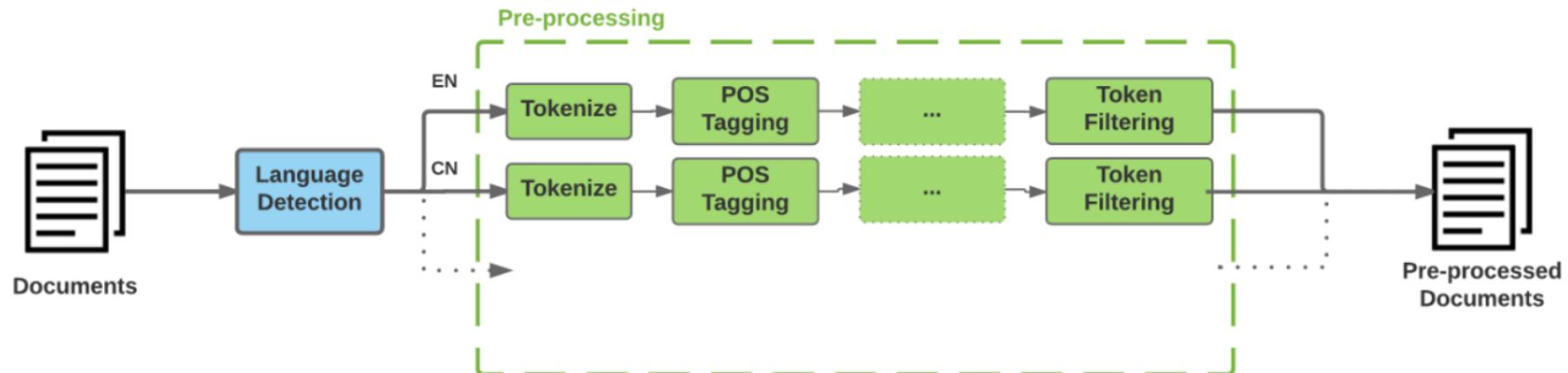
Classical NLP



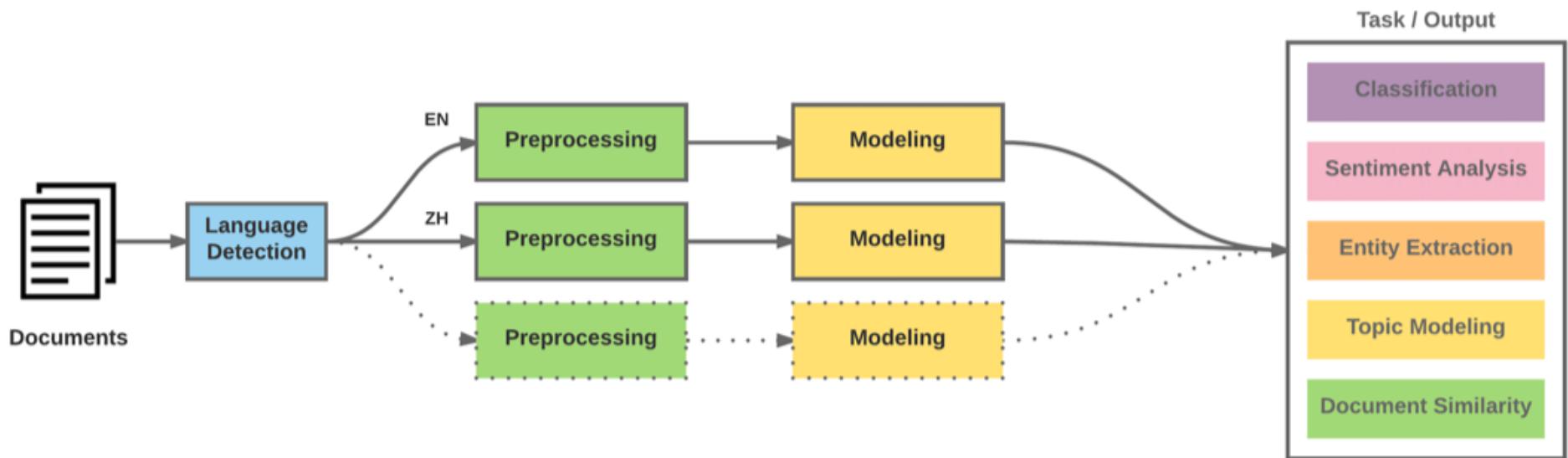
Deep Learning-based NLP



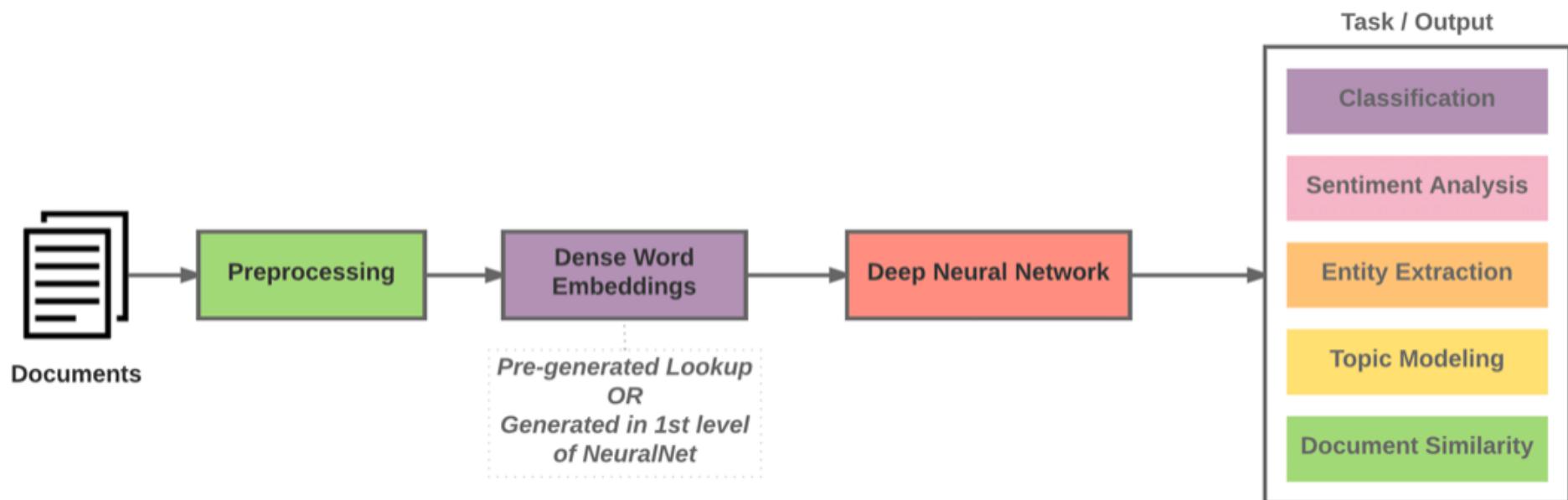
Modern NLP Pipeline



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



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



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://summar.es/ https://www-nplir.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

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.

Processing and Understanding Text

Free eBooks - Project Gutenberg

gutenberg.org

Project Gutenberg

search for books

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

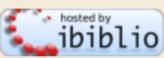
Project Gutenberg appreciates your donation!

[Donate](#)

- Why donate?

in other languages

- Português
- Deutsch
- Français

hosted by


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

Alice's Adventures in Wonderland +
gutenberg.org/files/11/11-h/11-h.htm

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

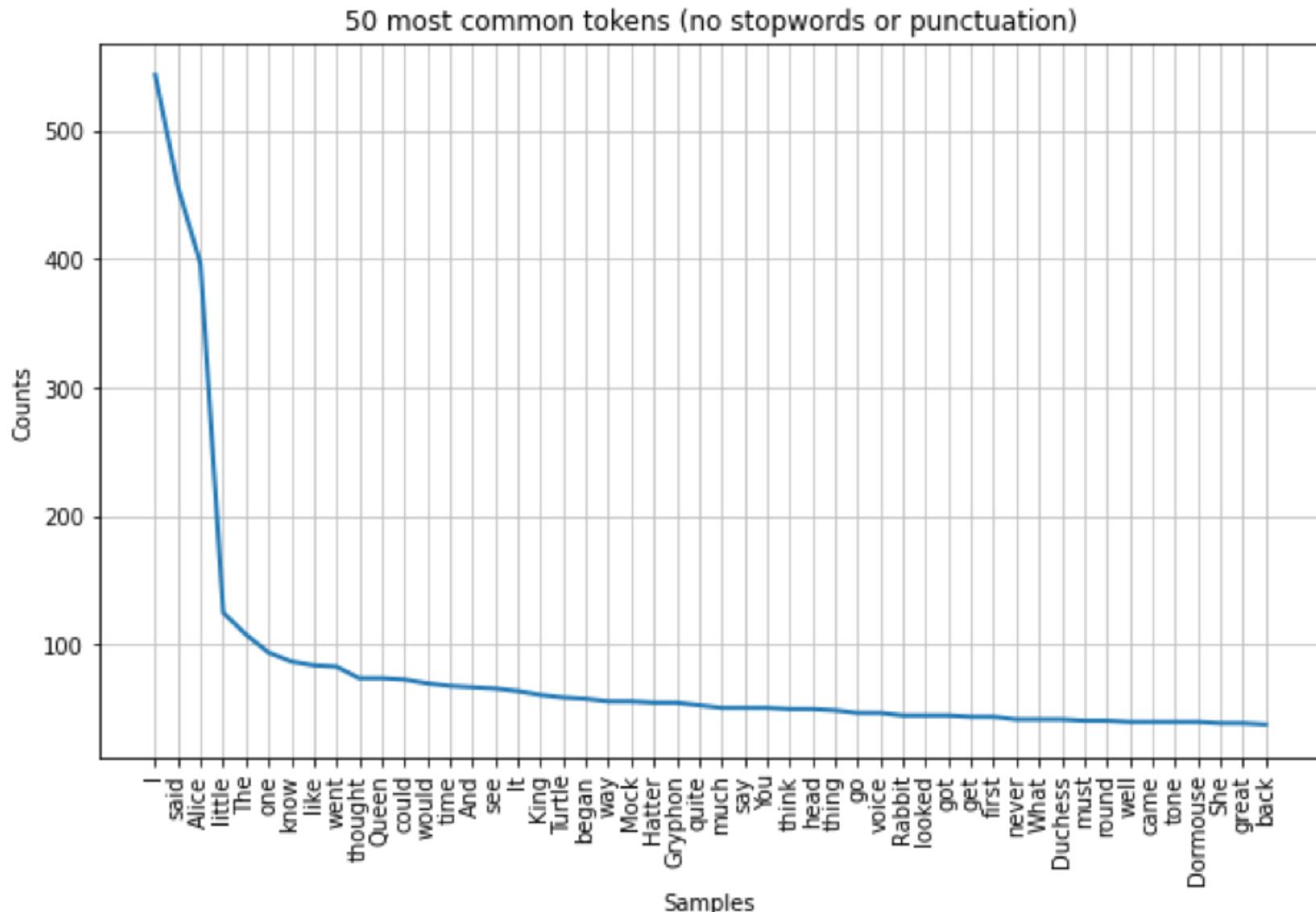
This eBook is for the use of anyone anywhere at no cost and with almost no restrictions whatsoever. You may copy it, give it away or re-use it under the terms of the Project Gutenberg License included with this eBook or online at www.gutenberg.org

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

A small thumbnail image of the book cover for "Alice's Adventures in Wonderland" by Lewis Carroll, showing the same illustration of Alice and the Red Queen.

Alice Top 50 Tokens



Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvWFUbeo4zJ1zTunjMqf2RkCrT>

```
nltk.download('gutenberg')

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

The screenshot shows the Google Colab interface with the following details:

- File Explorer:** On the left, it shows a tree view of files and notebooks.
- Header:** Shows the notebook name "python101.ipynb", a star icon for saving, and various menu options: File, Edit, View, Insert, Runtime, Tools, Help. It also indicates "All changes saved".
- Toolbar:** Includes Comment, Share, and settings icons.
- RAM/Disk:** Shows RAM usage at 100% and Disk usage at 0%.
- Section:** A section titled "Text Processing and Understanding" is expanded.
- List:** Under the section, there is a bulleted list of resources:
 - 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/>
- Code Cell 1:** Contains the code `nltk.download('gutenberg')` and its output showing the download of the 'gutenberg' corpus.
- Code Cell 2:** Contains the code `Text(nltk.corpus.gutenberg.words('carroll-alice.txt'))` and its output, which is the text of Alice's Adventures in Wonderland.
- Code Cell 3:** Contains the code `print(nltk.corpus.gutenberg.fileids())` and its output, which lists several file IDs.

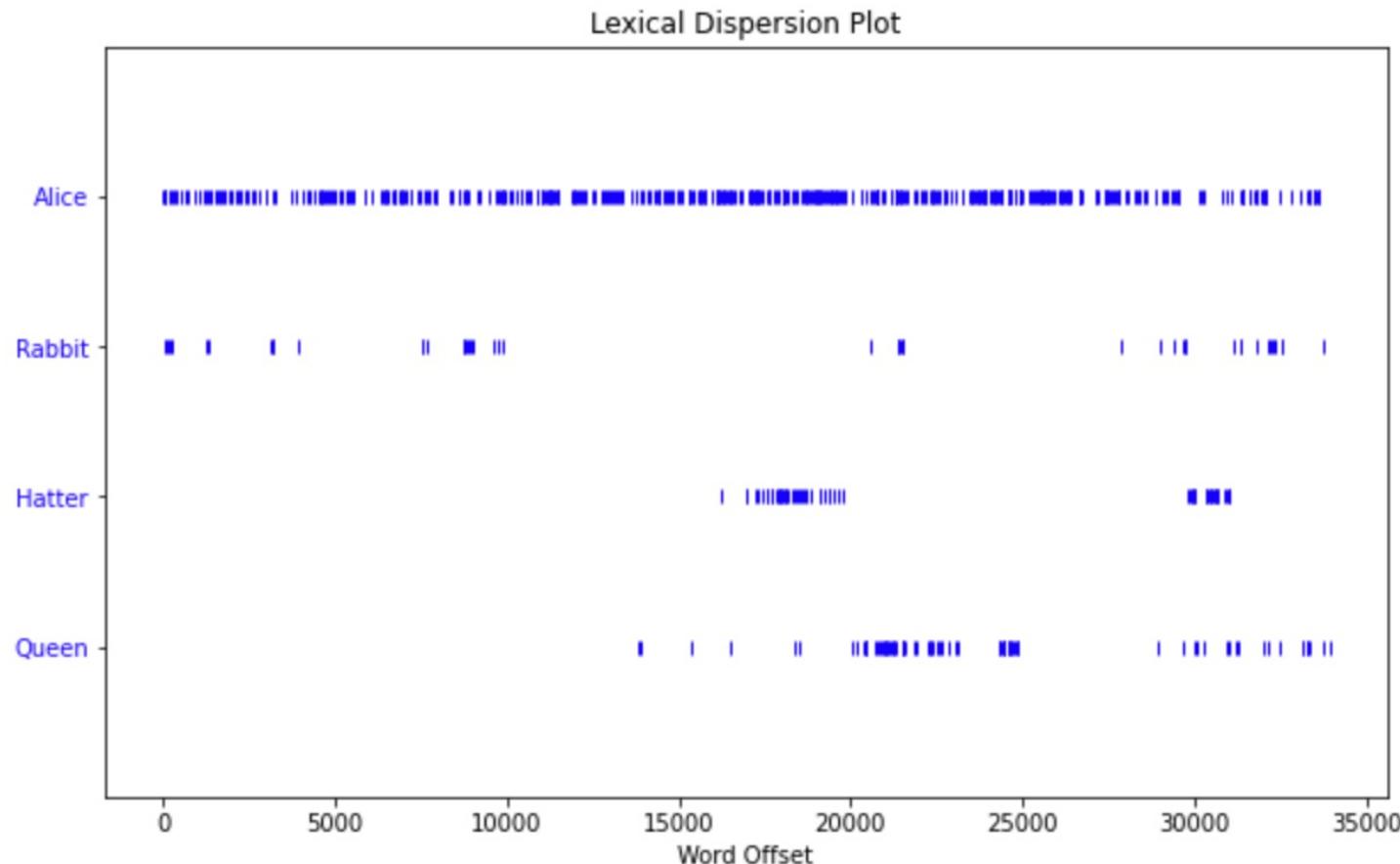
alice.concordance ("Alice")

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

Displaying 25 of 398 matches:

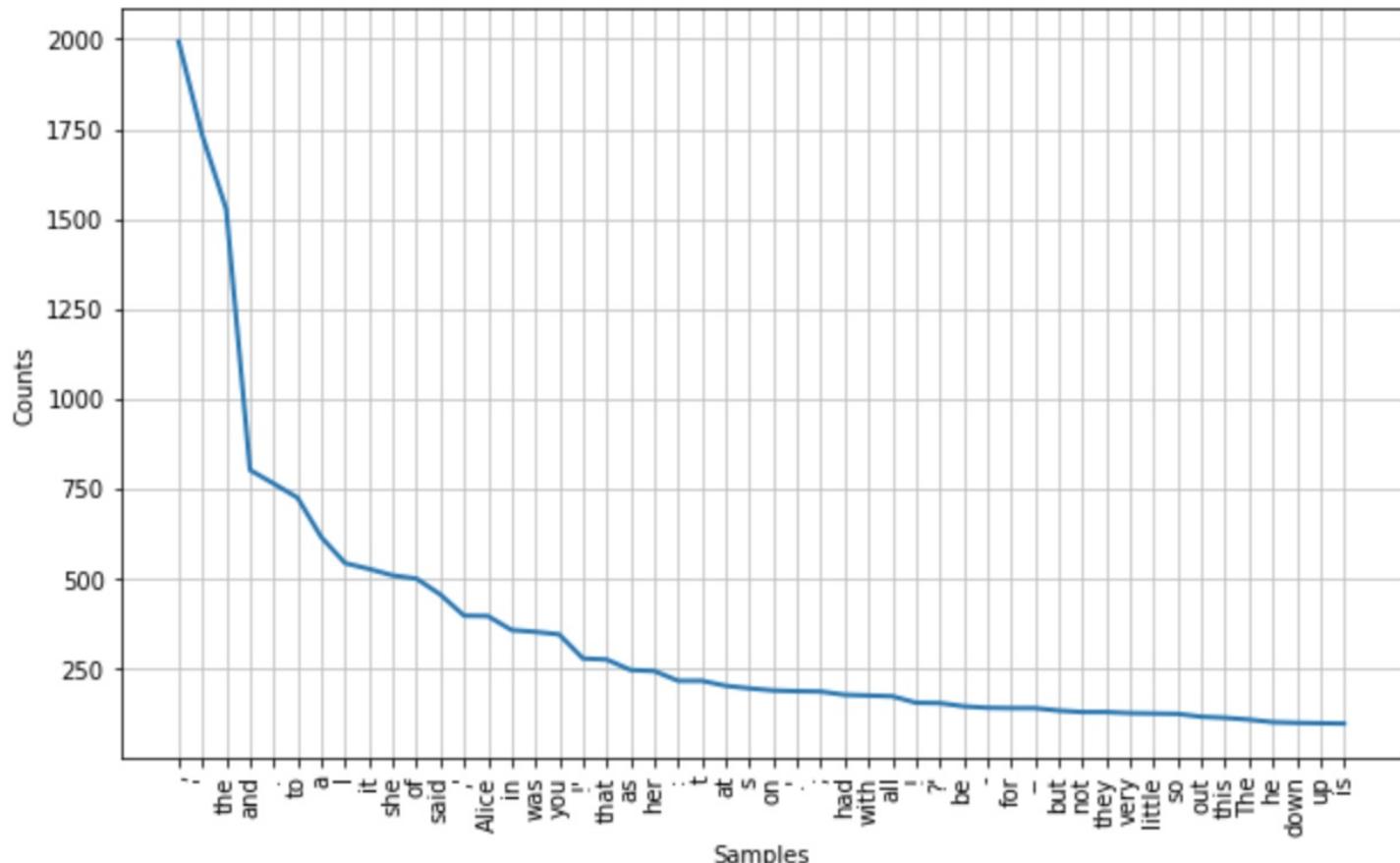
```
alice.dispersion_plot(["Alice", "Rabbit",
                      "Hatter", "Queen"])
```

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



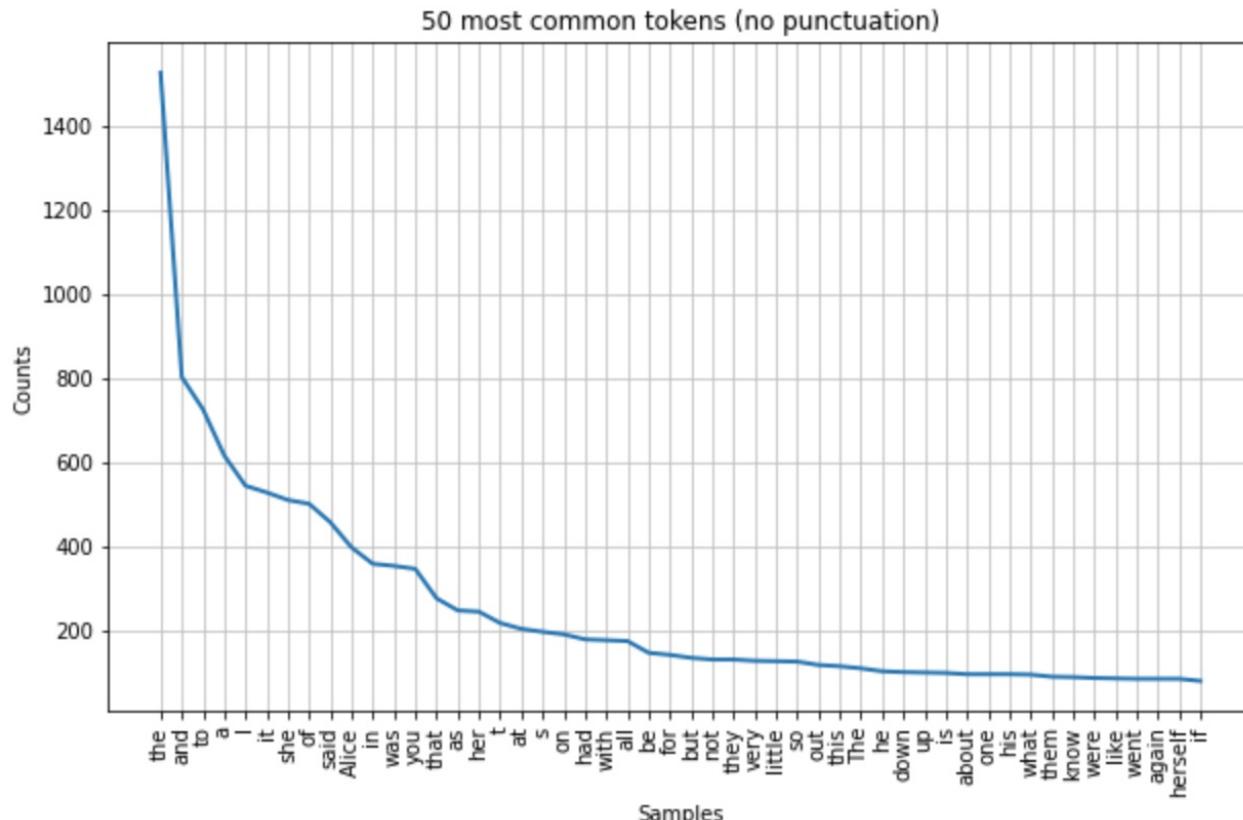
```
fdist = nltk.FreqDist(alice)
fdist.plot(50)
```

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



```
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)")
```



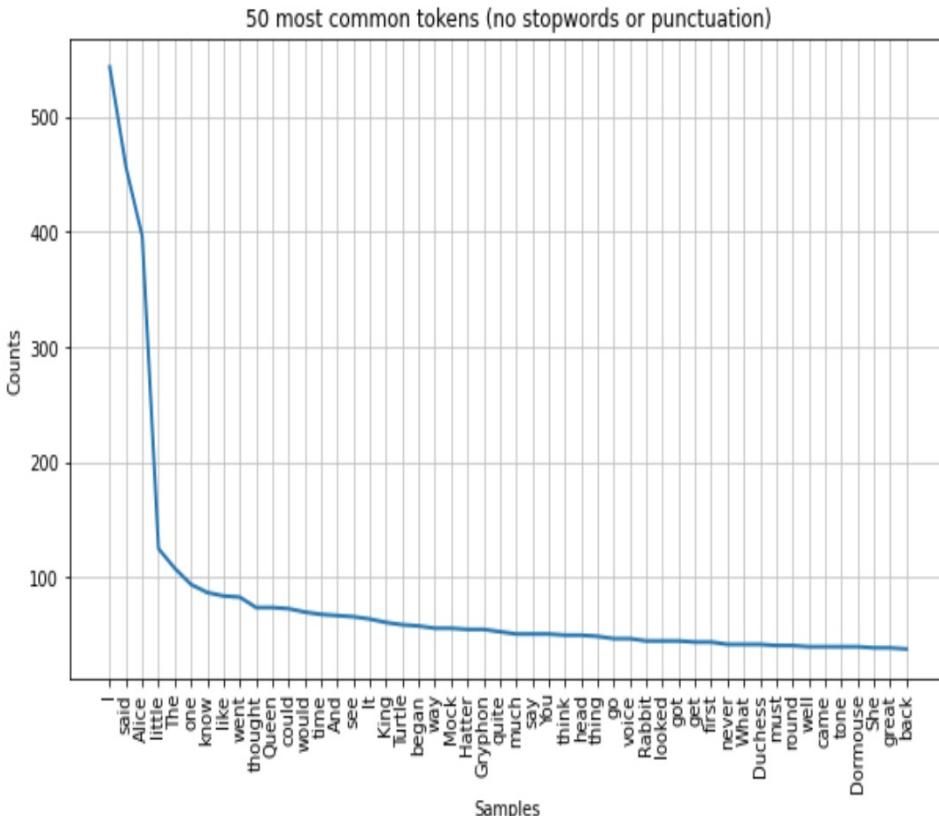
```
nltk.download('stopwords')
stopwords = nltk.corpus.stopwords.words('english')
```

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

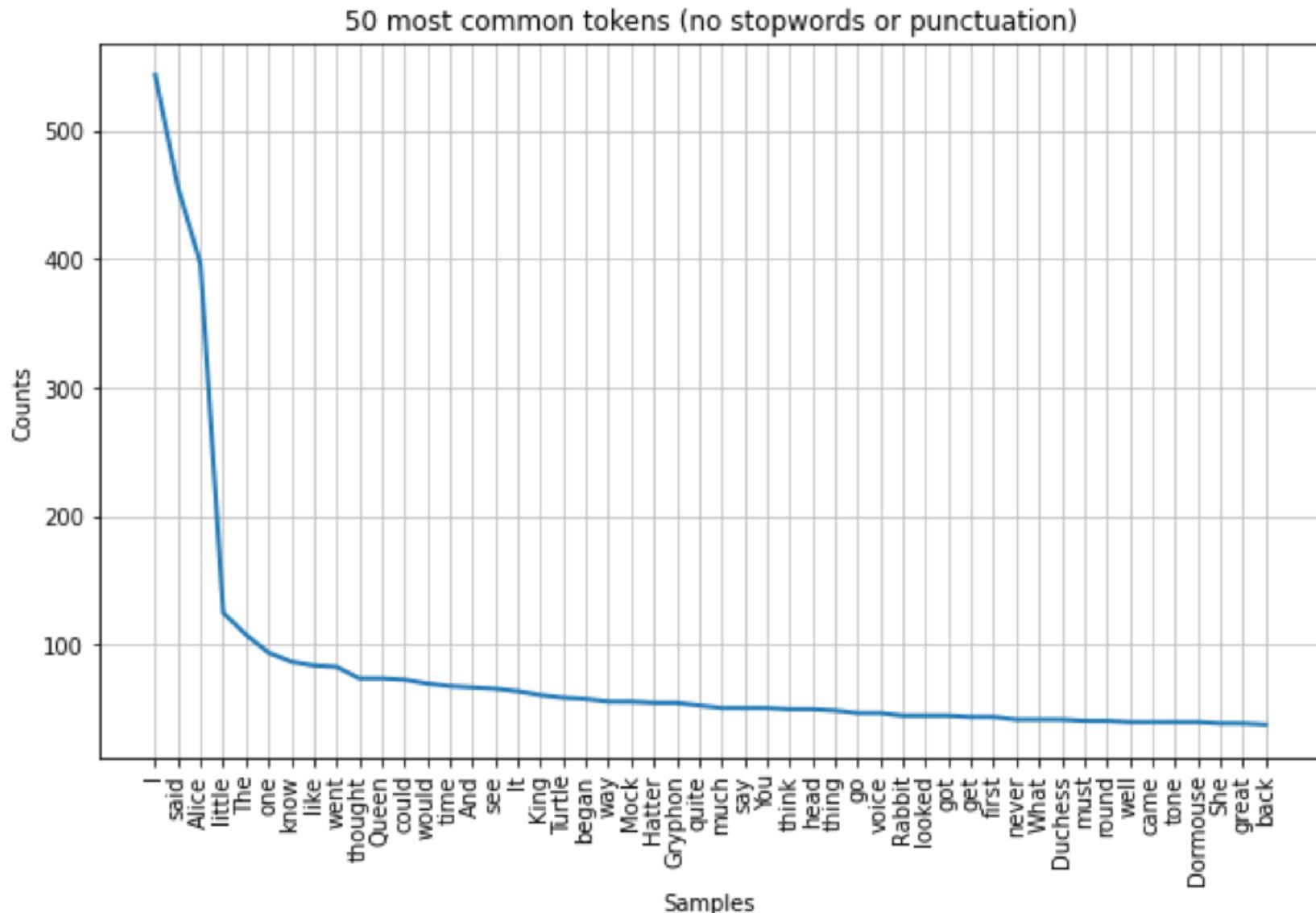
'same',
'so',
'than',
'too',
'very',
's',
't',
'can',
'will',
'just',
'don',
"don't",
'should',
"should've",
'now',
```

```
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)")
```



Alice Top 50 Tokens



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="")
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. The title bar says 'python101.ipynb'. The menu bar includes File, Edit, View, Insert, Runtime, Tools, Help, COMMENT, SHARE, CONNECT, EDITING, CODE, TEXT, CELL UP, CELL DOWN, and A. The main content area has a heading 'Keras preprocessing text'. Below it is a code cell containing Python code for tokenizing text using Keras. The output cell shows the results of running the code, including the documents, word counts, document count, word index, word docs, and the resulting text matrix.

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

Using TensorFlow backend.

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

<https://tinyurl.com/aintpuppython101>

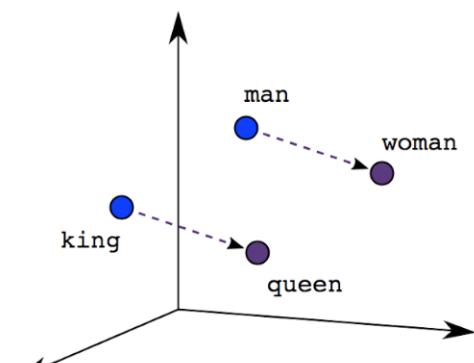
One-hot encoding

'The mouse ran up the clock' =

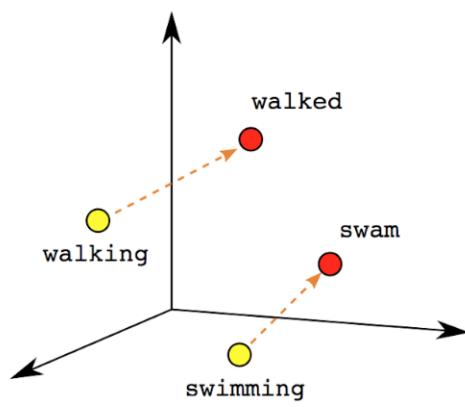
The	1	[[0, 1, 0, 0, 0, 0, 0],
mouse	2	[0, 0, 1, 0, 0, 0, 0],
ran	3	[0, 0, 0, 1, 0, 0, 0],
up	4	[0, 0, 0, 0, 1, 0, 0],
the	1	[0, 1, 0, 0, 0, 0, 0],
clock	5	[0, 0, 0, 0, 0, 1, 0]]

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

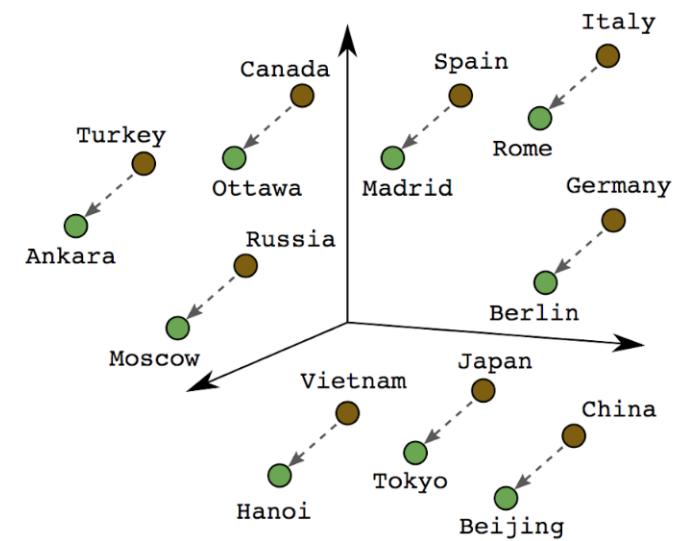
Word embeddings



Male-Female

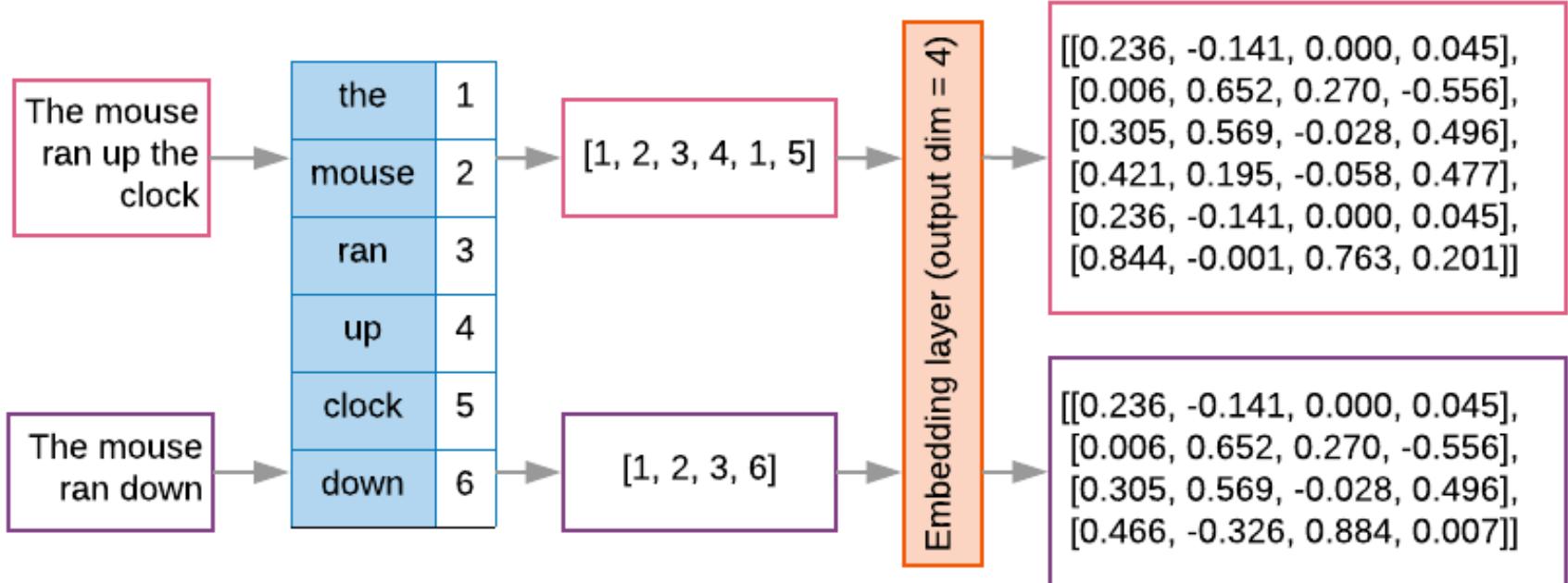


Verb Tense



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)}
```

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

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.]]
```

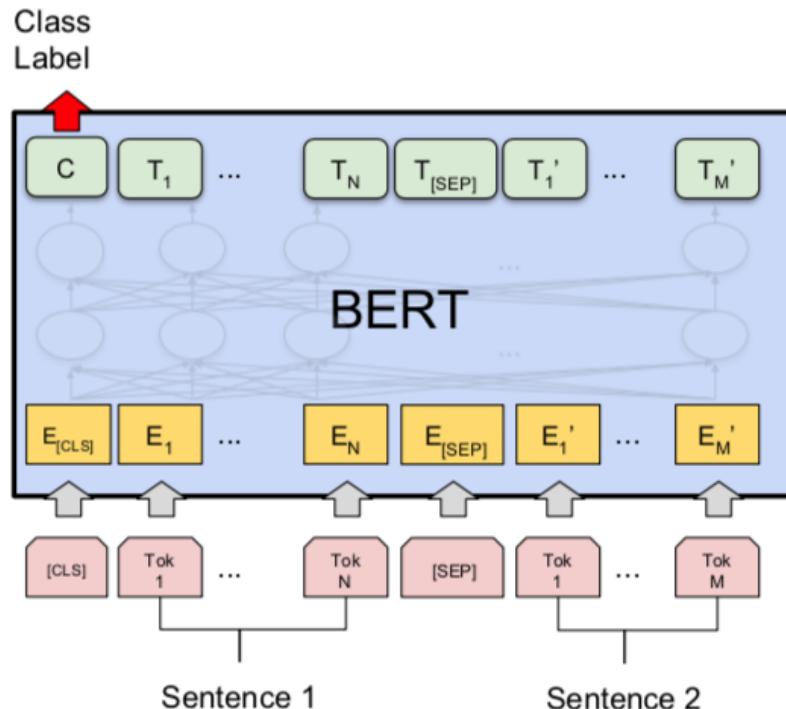
t.texts_to_matrix(docs, mode='tfidf')

```
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)
```

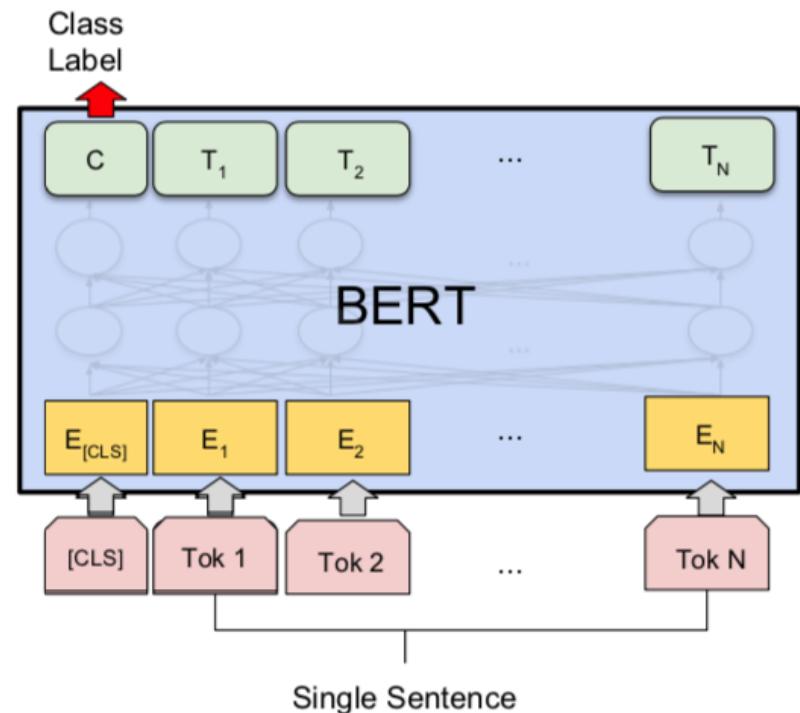
texts_to_matrix:

```
[[0. 0. 1.25276297 1.25276297 0. 0. 0. 0. 0. 0. ]
[0. 0.98082925 0. 0. 1.25276297 0. 0. 0. 0. 0. ]
[0. 0. 0. 0. 1.25276297 1.25276297 0. 0. 0. ]
[0. 0.98082925 0. 0. 0. 0. 1.25276297 0. 0. ]
[0. 0. 0. 0. 0. 0. 0. 1.25276297]]
```

BERT Sequence-level tasks

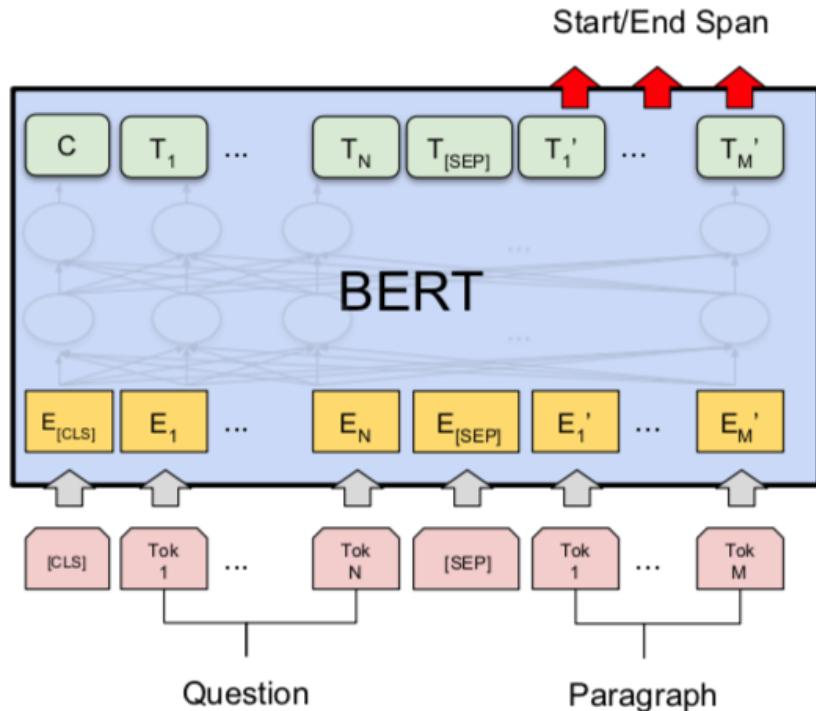


(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG

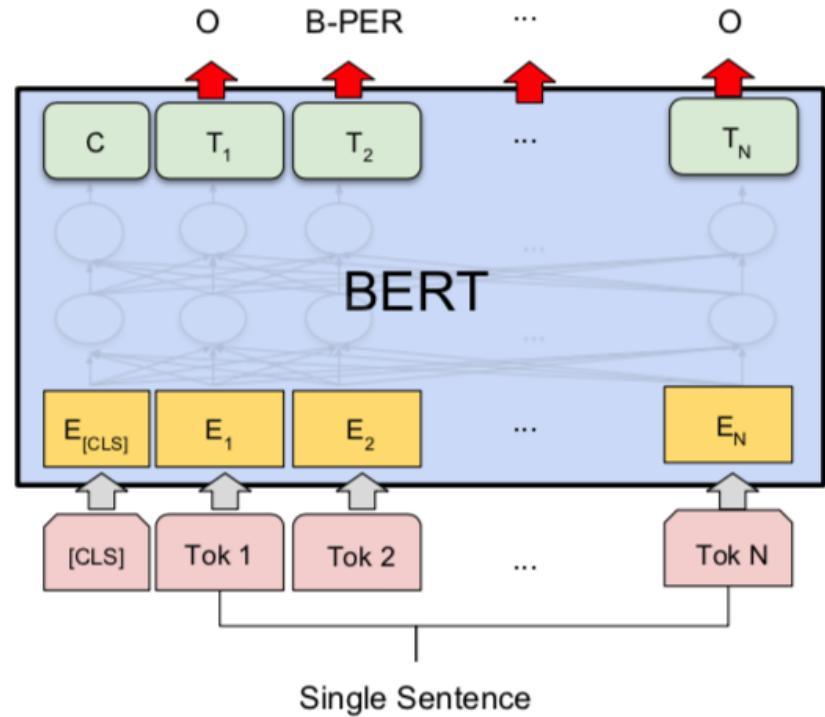


(b) Single Sentence Classification Tasks:
SST-2, CoLA

BERT Token-level tasks



(c) Question Answering Tasks:
SQuAD v1.1

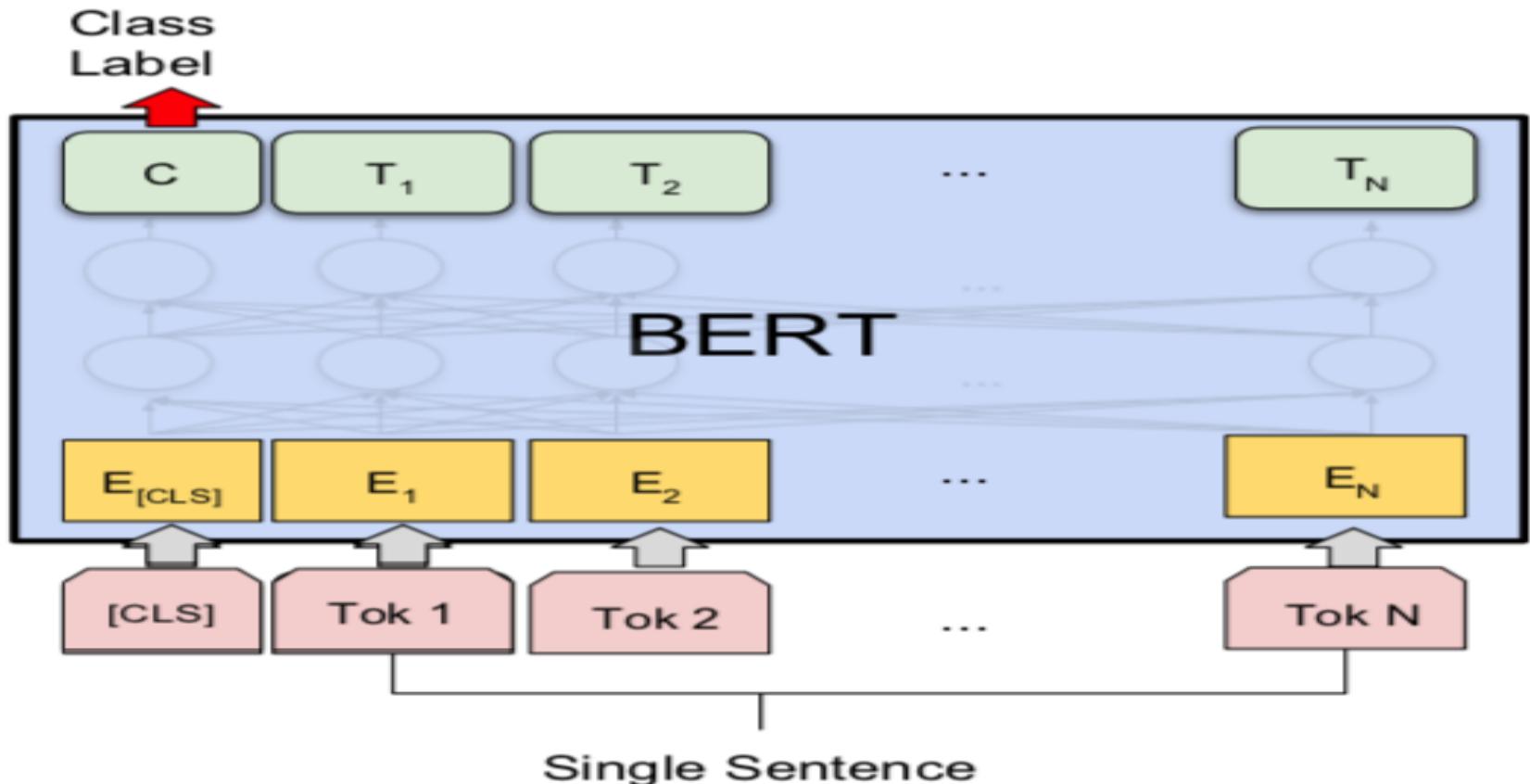


(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

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

A Visual Guide to Using BERT for the First Time

(Jay Alammar, 2019)

“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**

Sentiment Classification: SST2

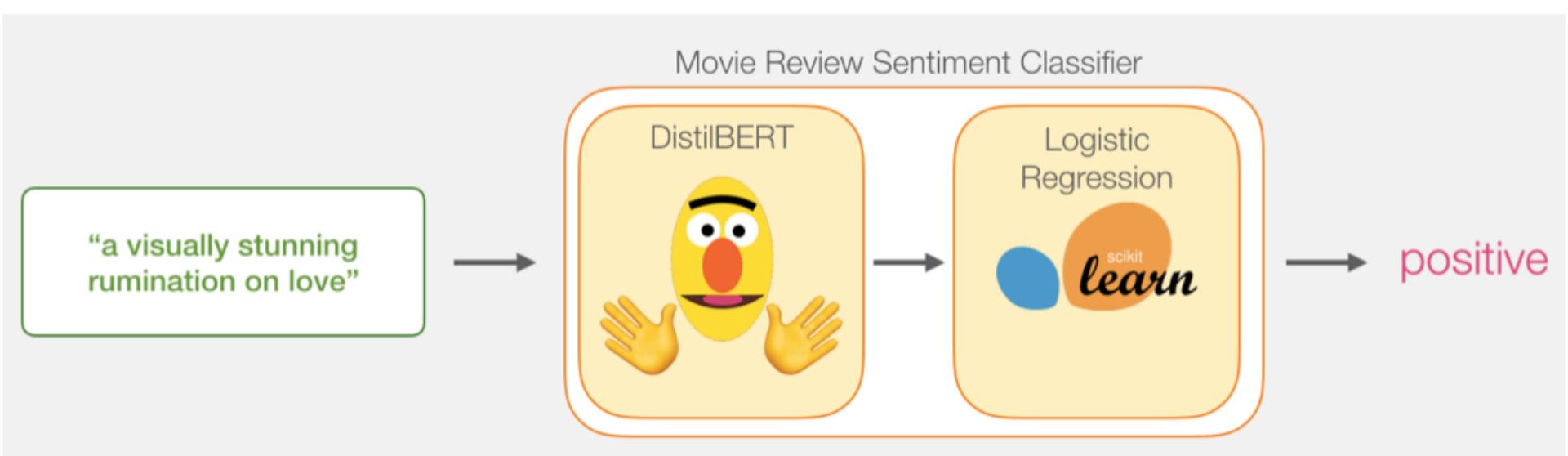
Sentences from movie reviews

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

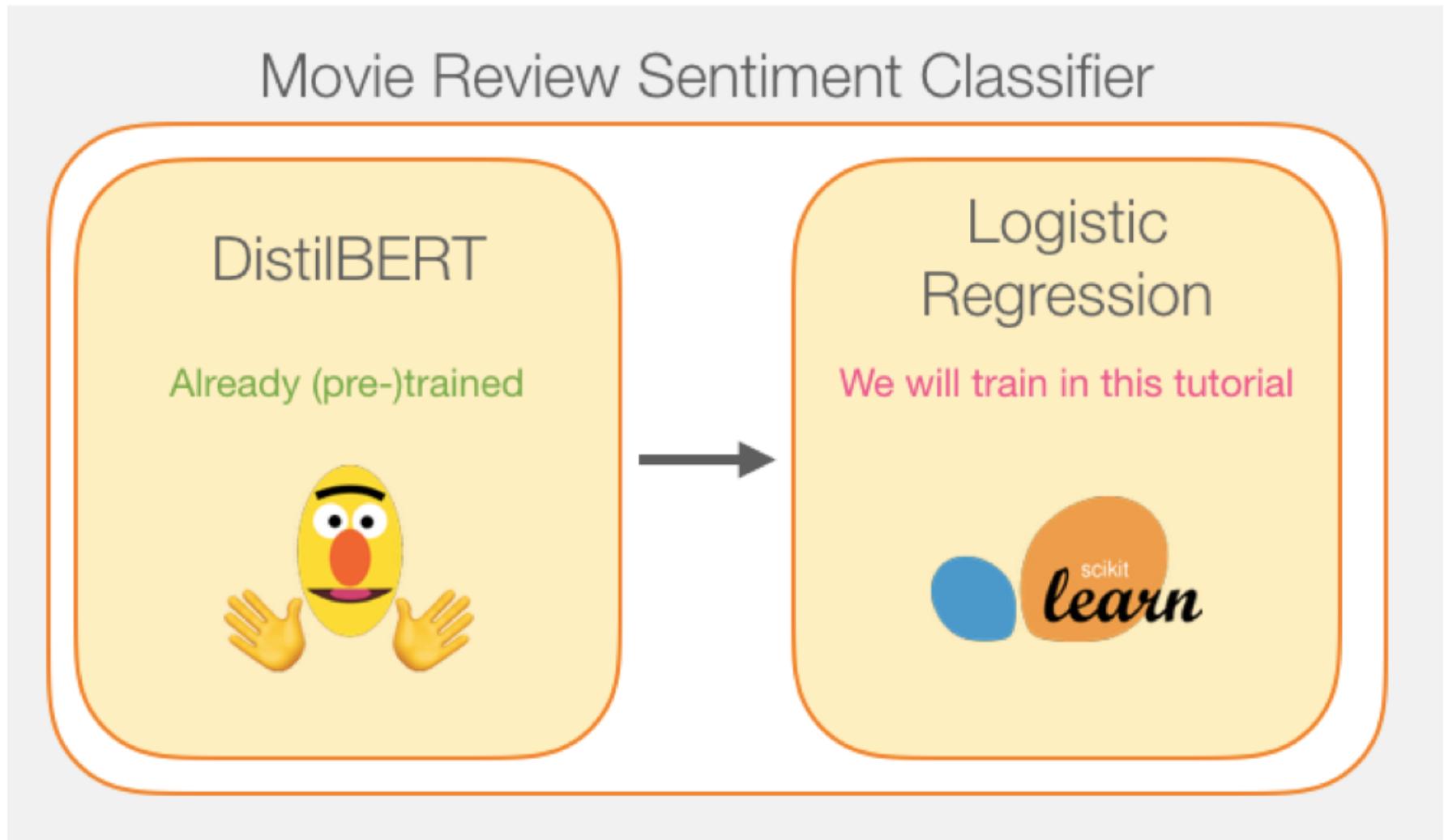


Movie Review Sentiment Classifier



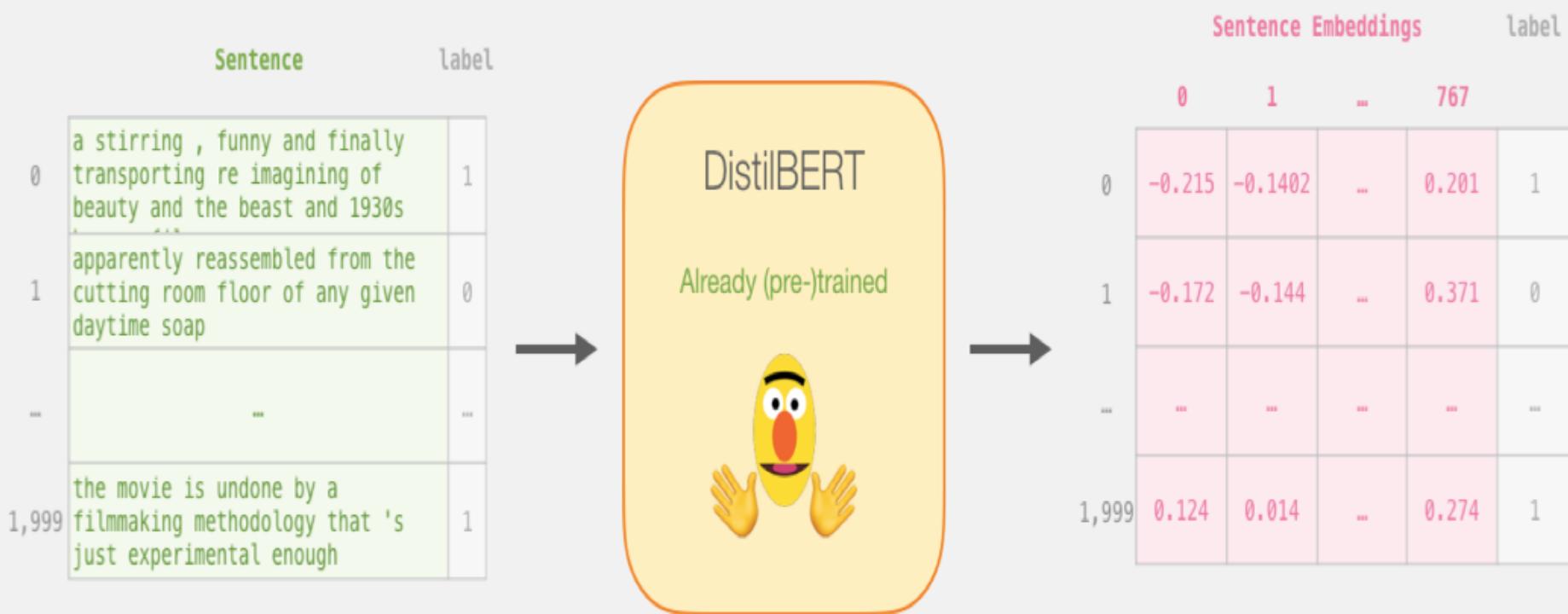
Movie Review Sentiment Classifier

Model Training



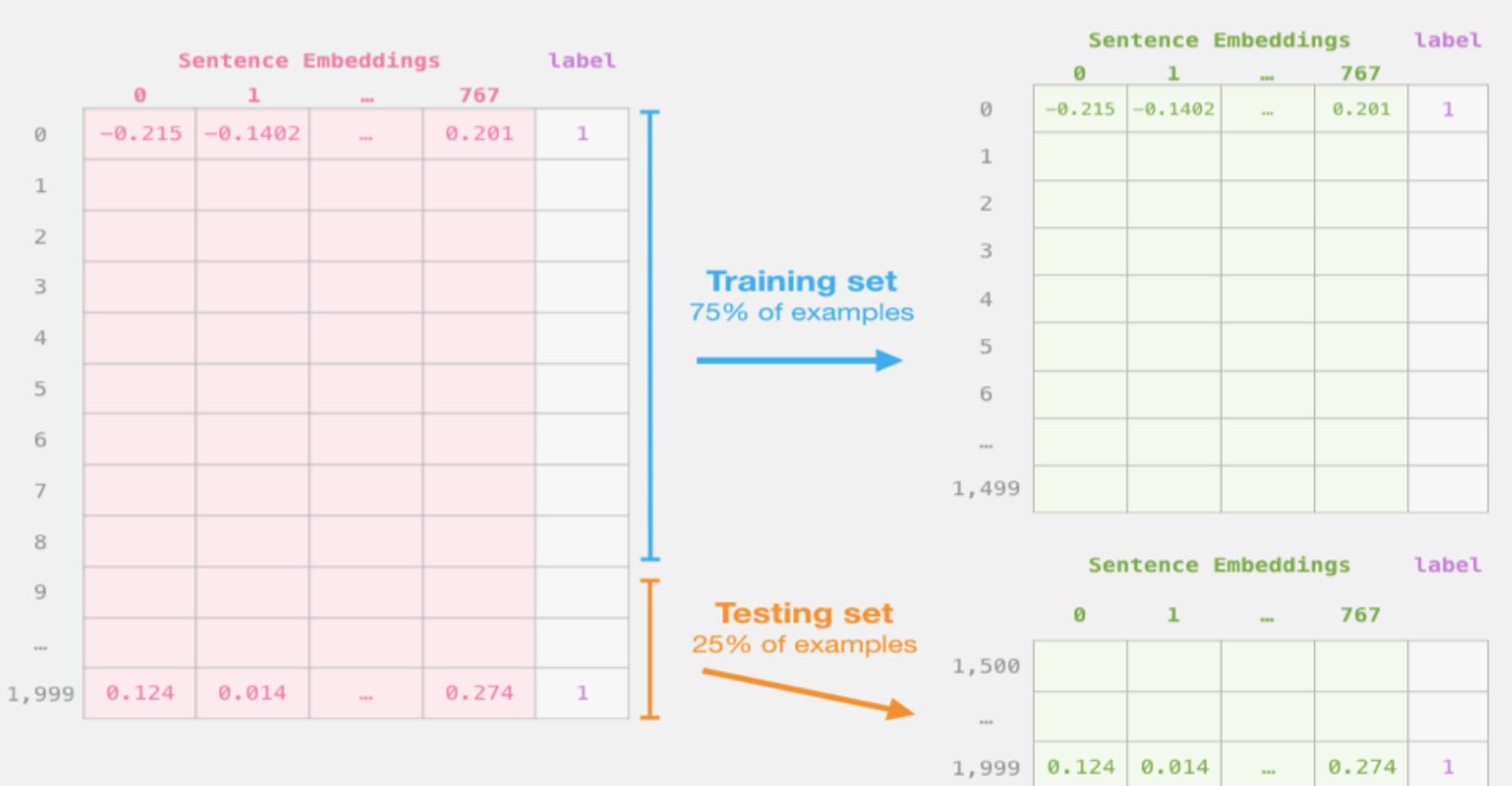
Step # 1 Use distilBERT to Generate Sentence Embeddings

Step #1: Use DistilBERT to embed all the sentences



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

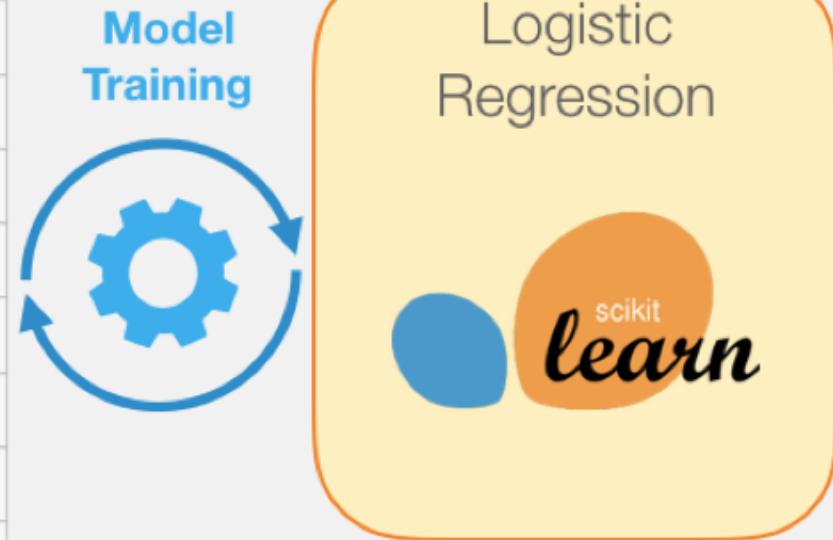
Step #2: Test/Train Split for model #2, logistic regression



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

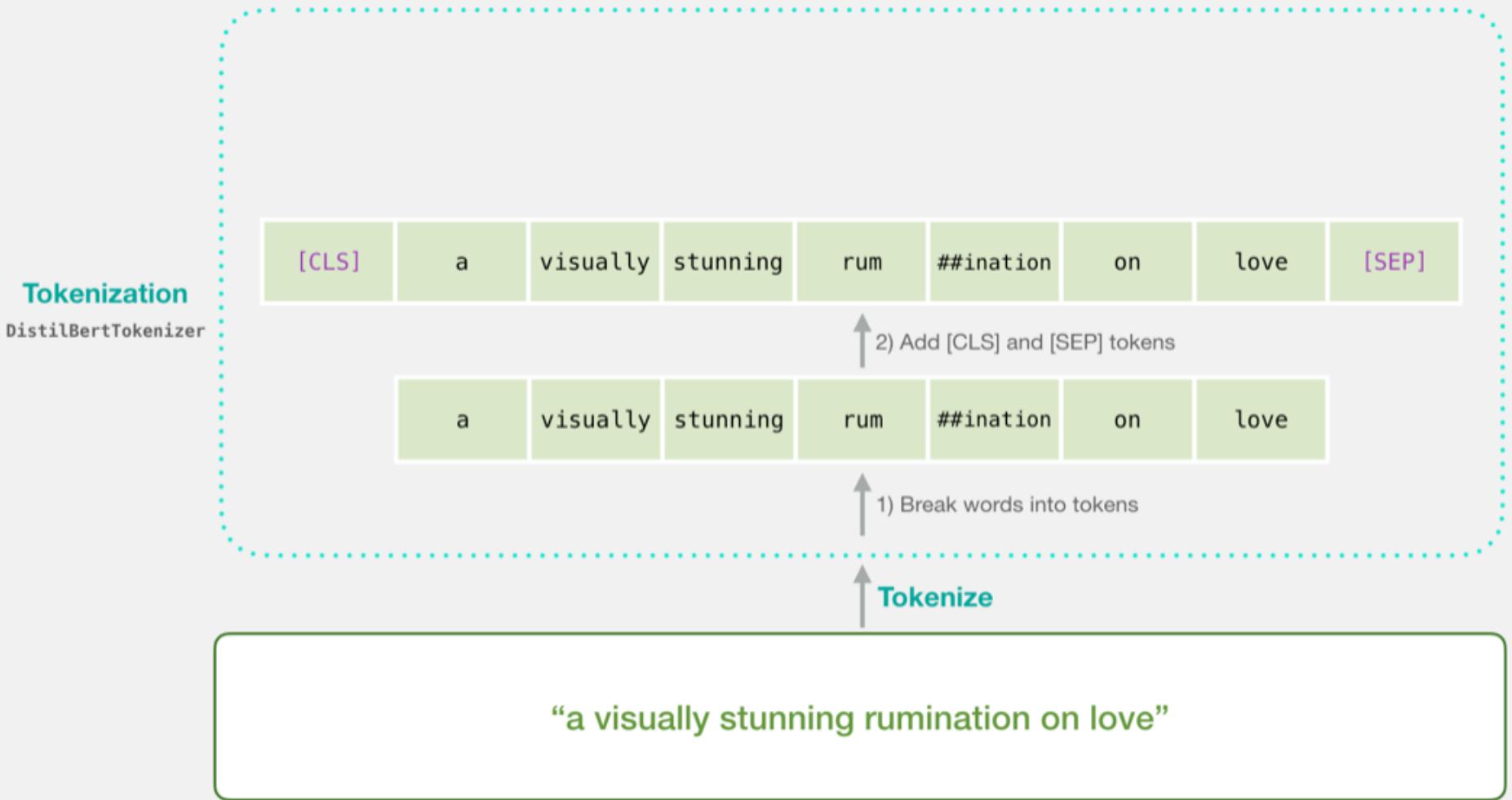
Step #3: Train the logistic regression model using the training set

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



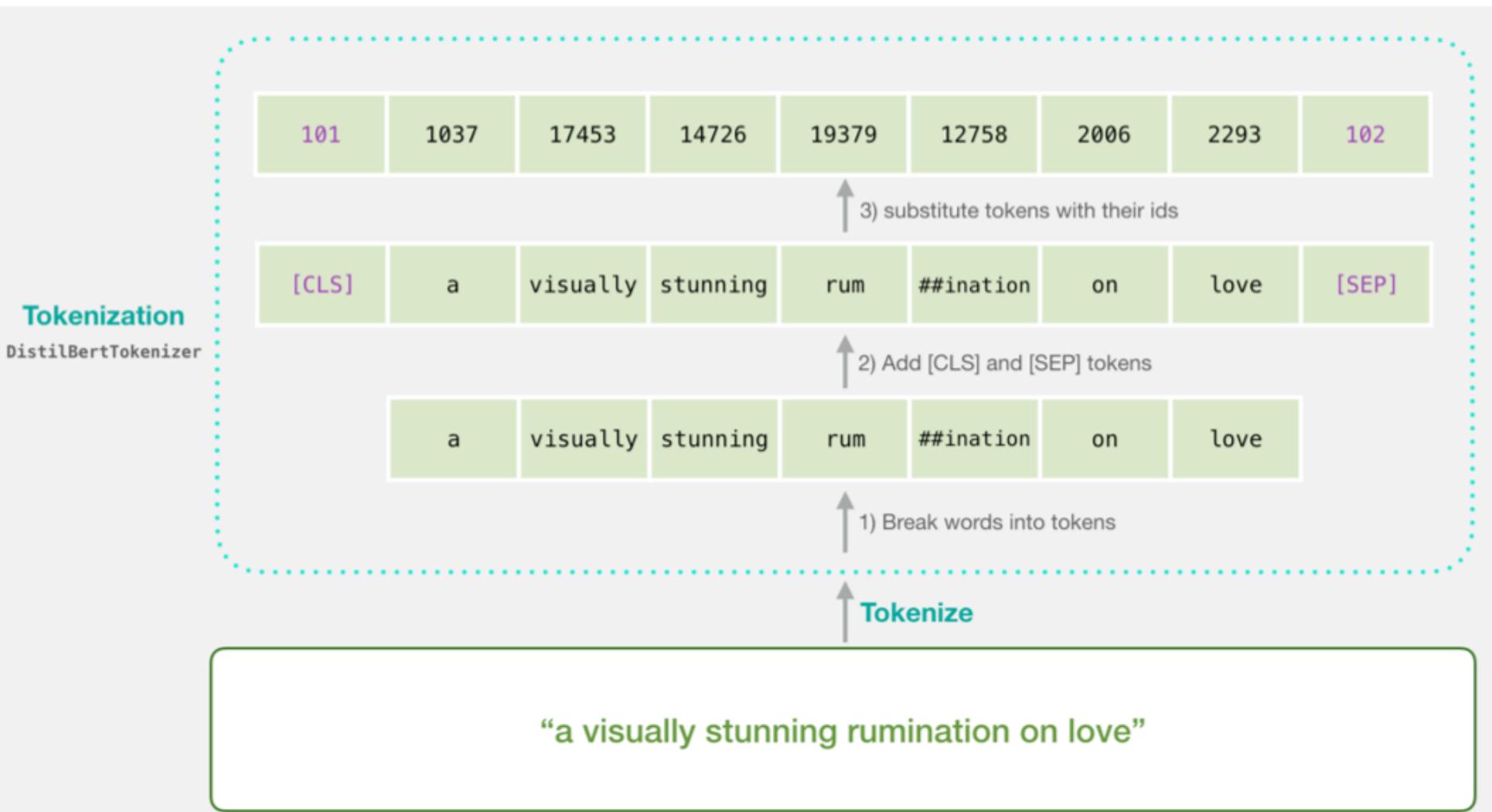
Tokenization

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

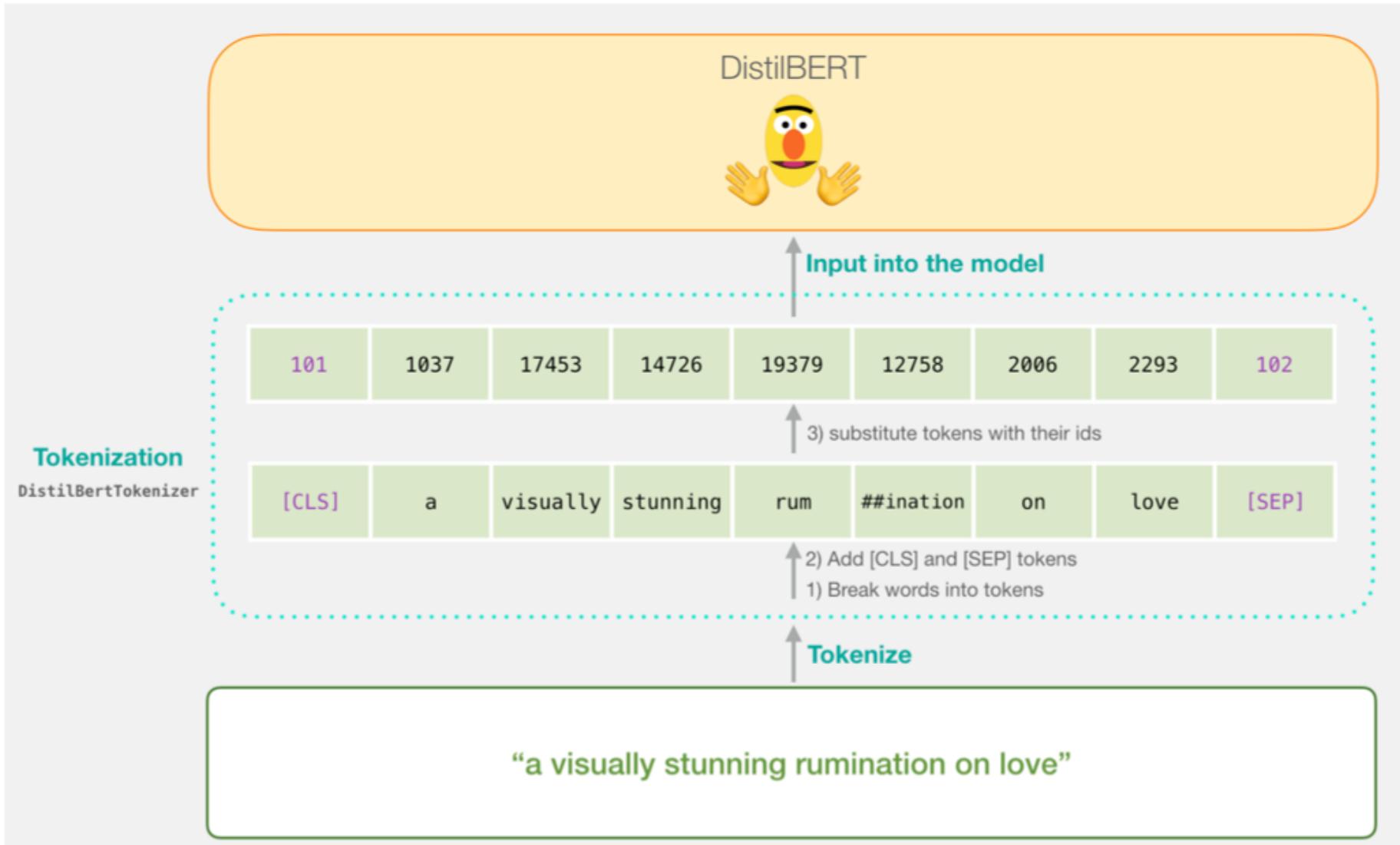


Tokenization

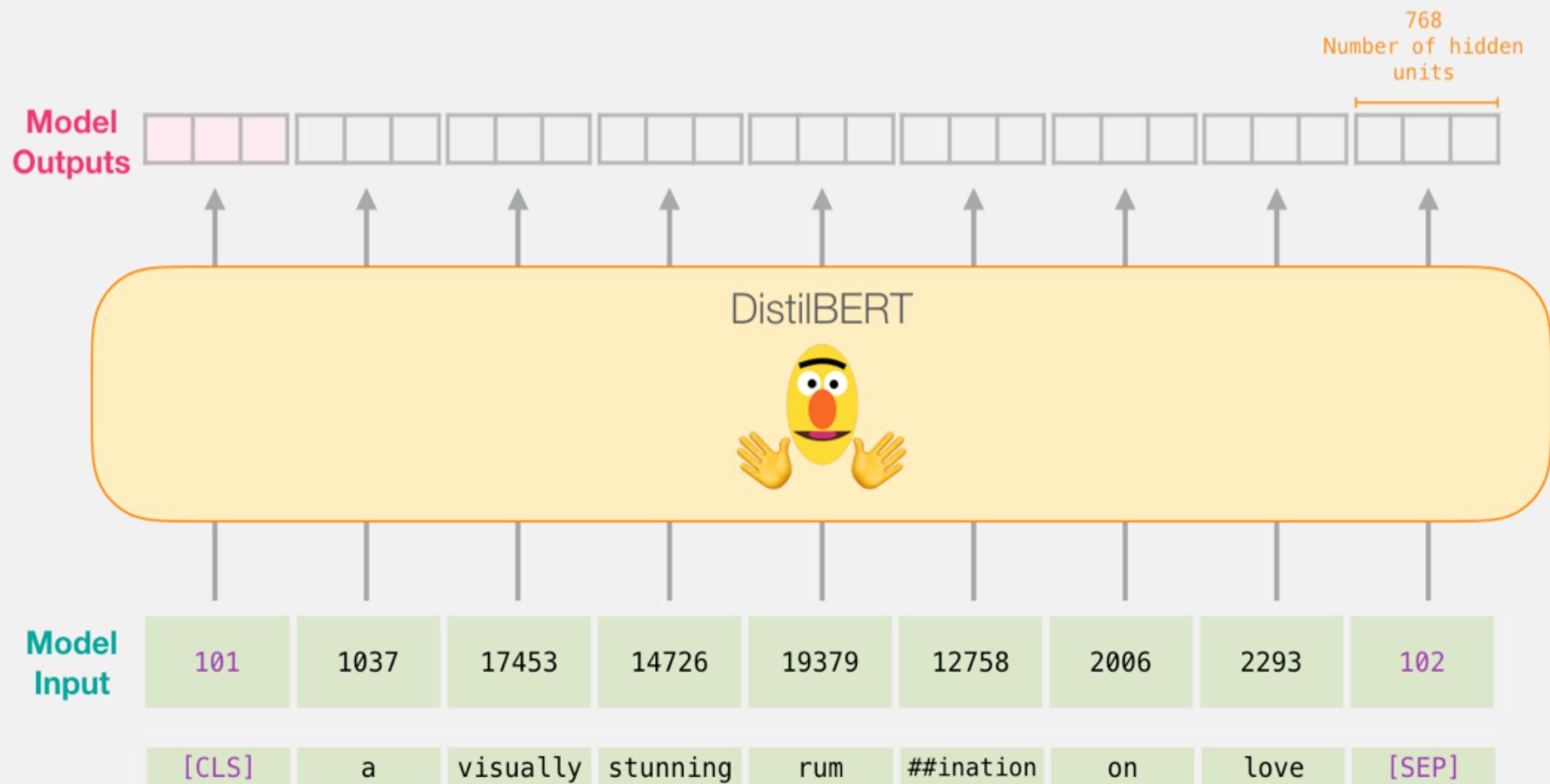
```
tokenizer.encode("a visually stunning rumption on love",  
    add_special_tokens=True)
```



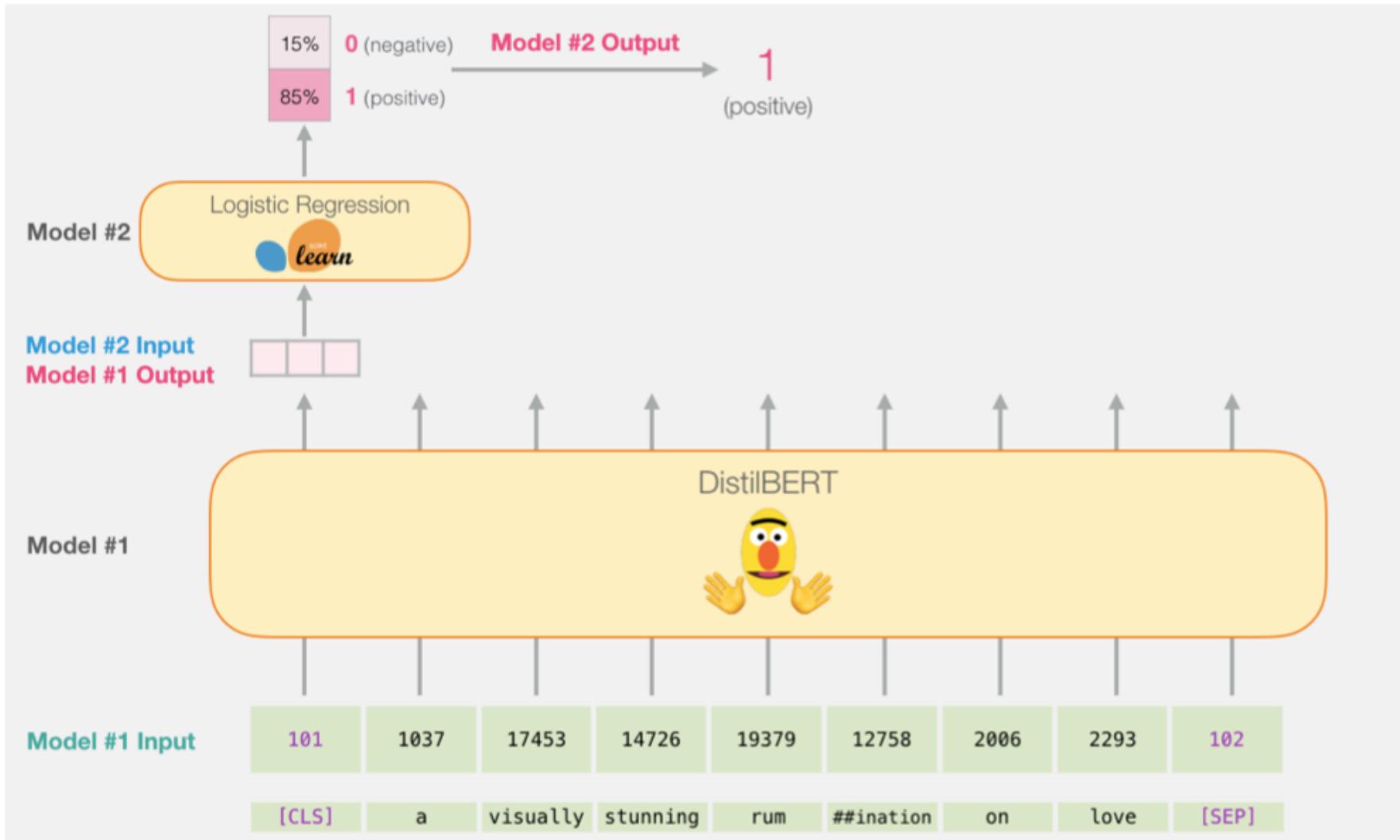
Tokenization for BERT Model



Flowing Through DistilBERT (768 features)

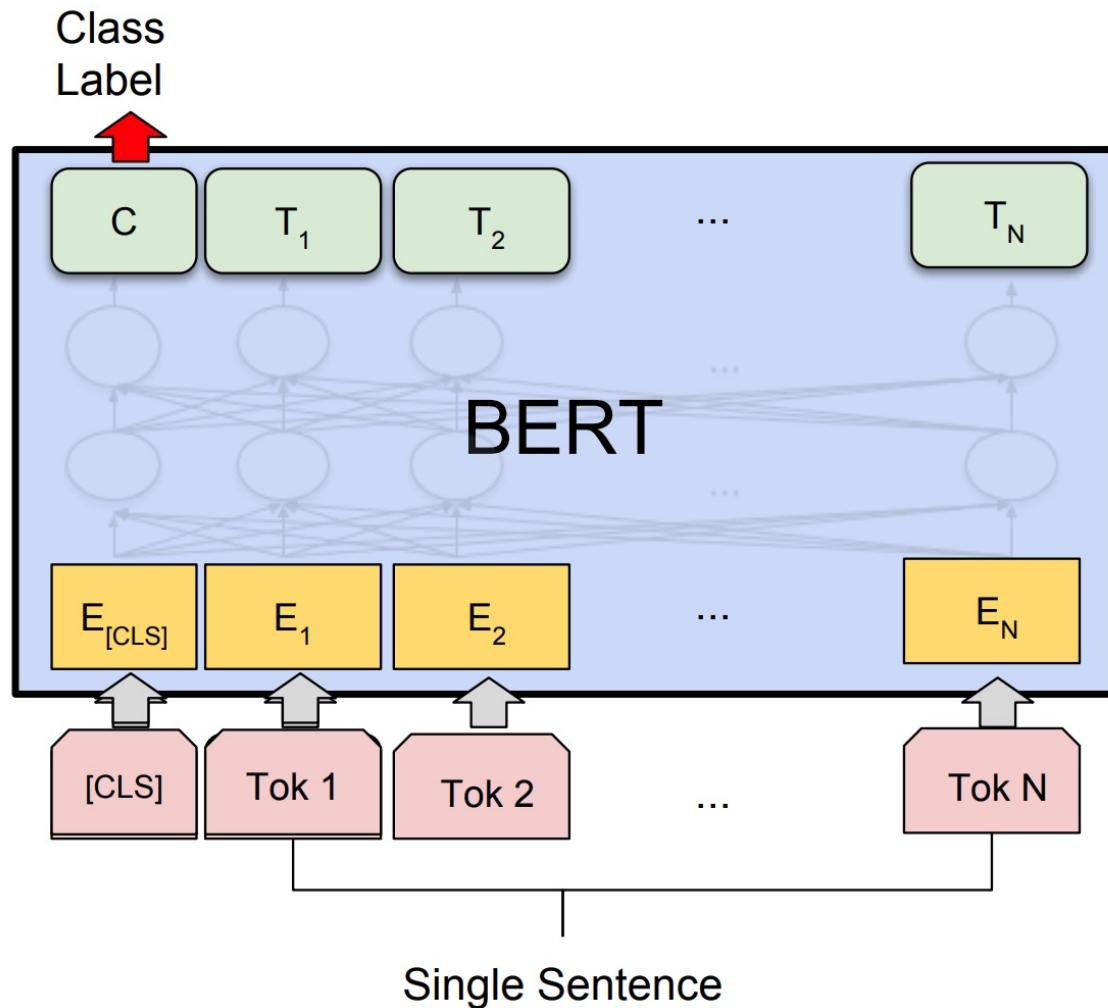


Model #1 Output Class vector as Model #2 Input



Source: Jay 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/>

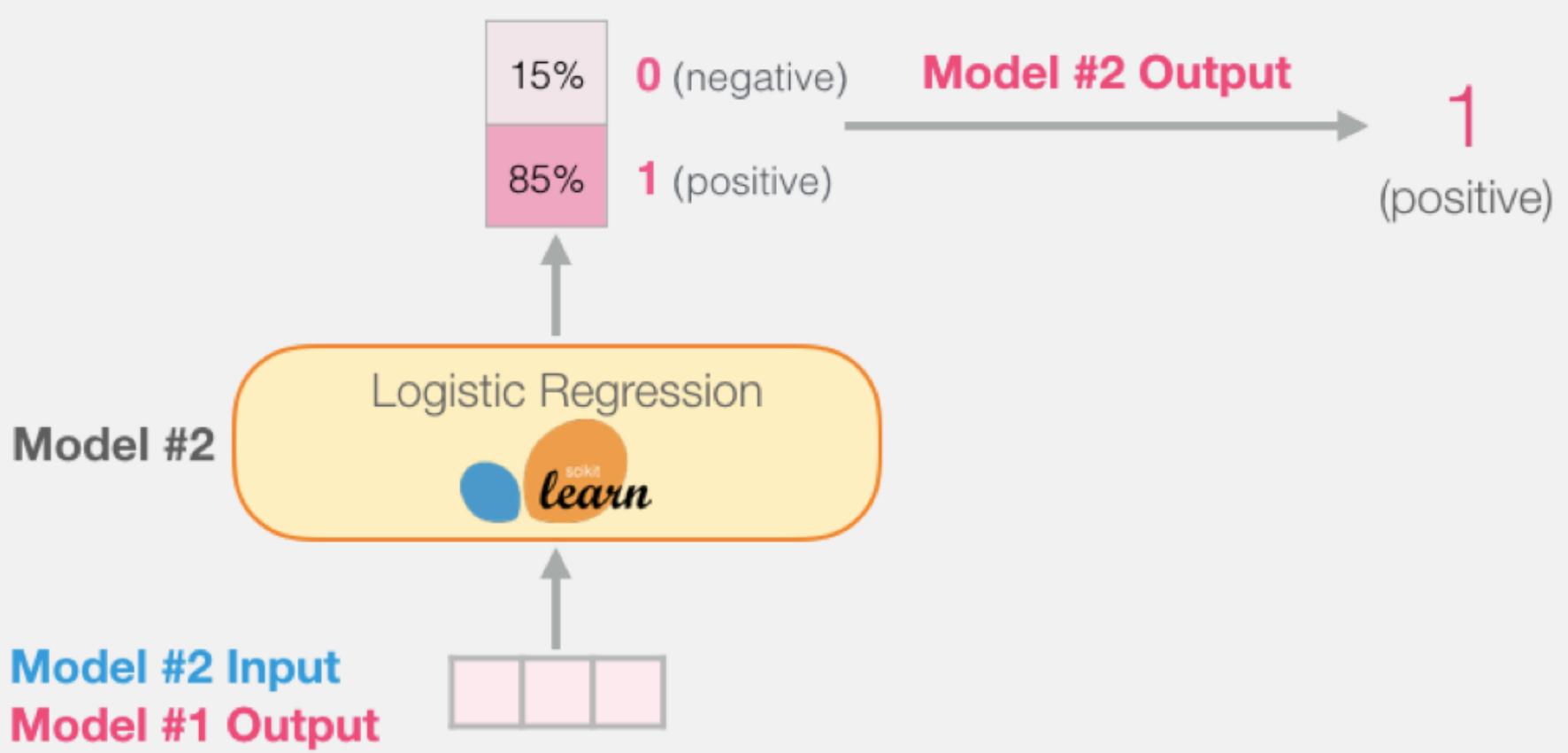
Fine-tuning BERT on Single Sentence Classification Tasks



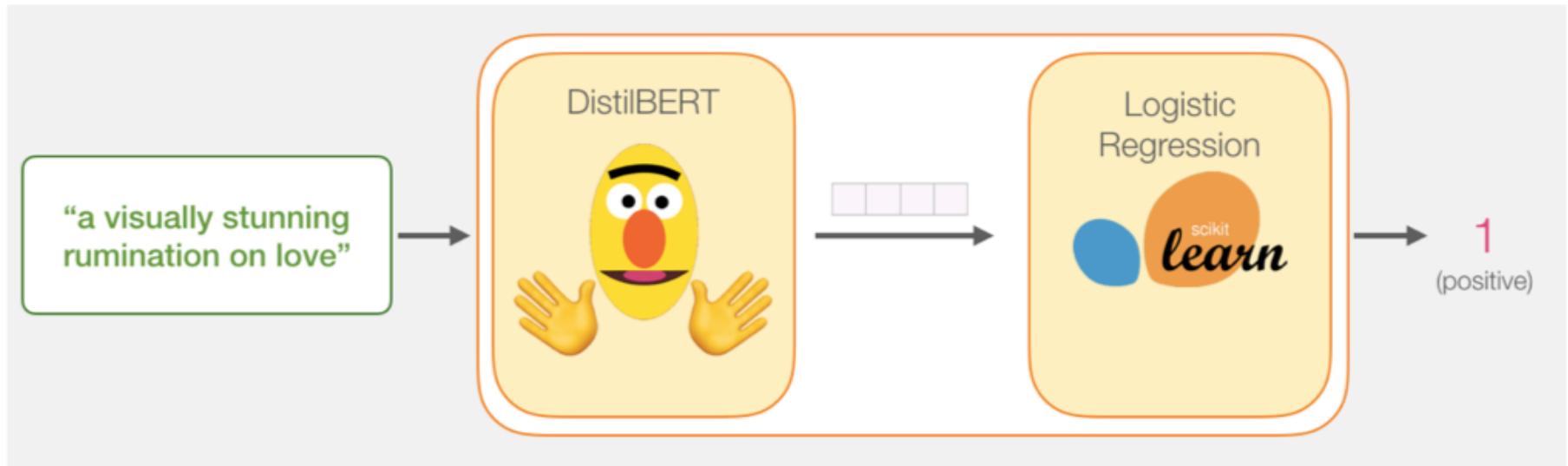
Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).

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

Model #1 Output Class vector as Model #2 Input



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 ruminations 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...
```

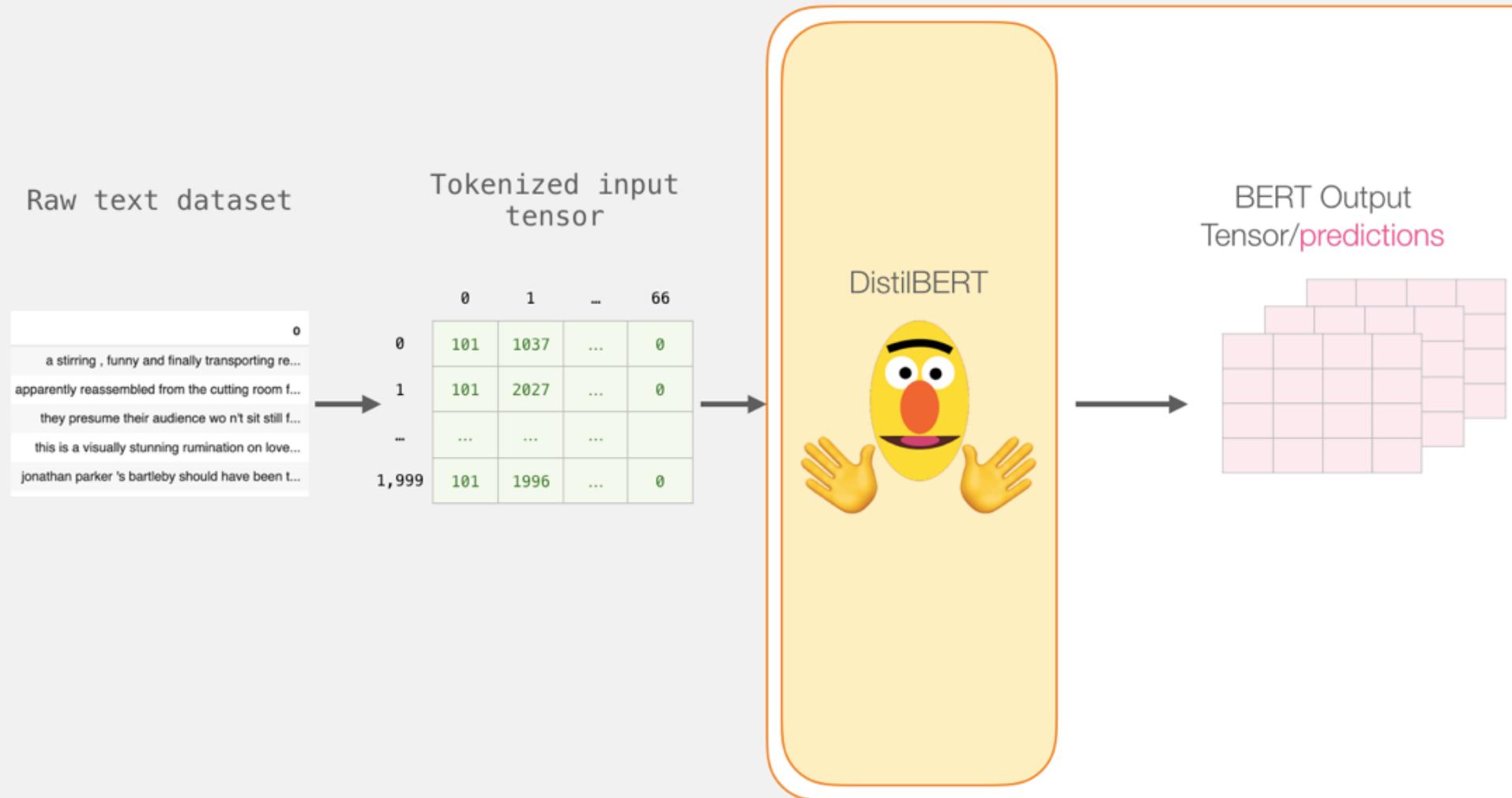
BERT Input Tensor

BERT/DistilBERT Input Tensor

Tokens in each sequence				
Input sequences (reviews)	0	1	...	66
0	101	1037	...	0
1	101	2027	...	0
...	
1,999	101	1996	...	0

Processing with DistilBERT

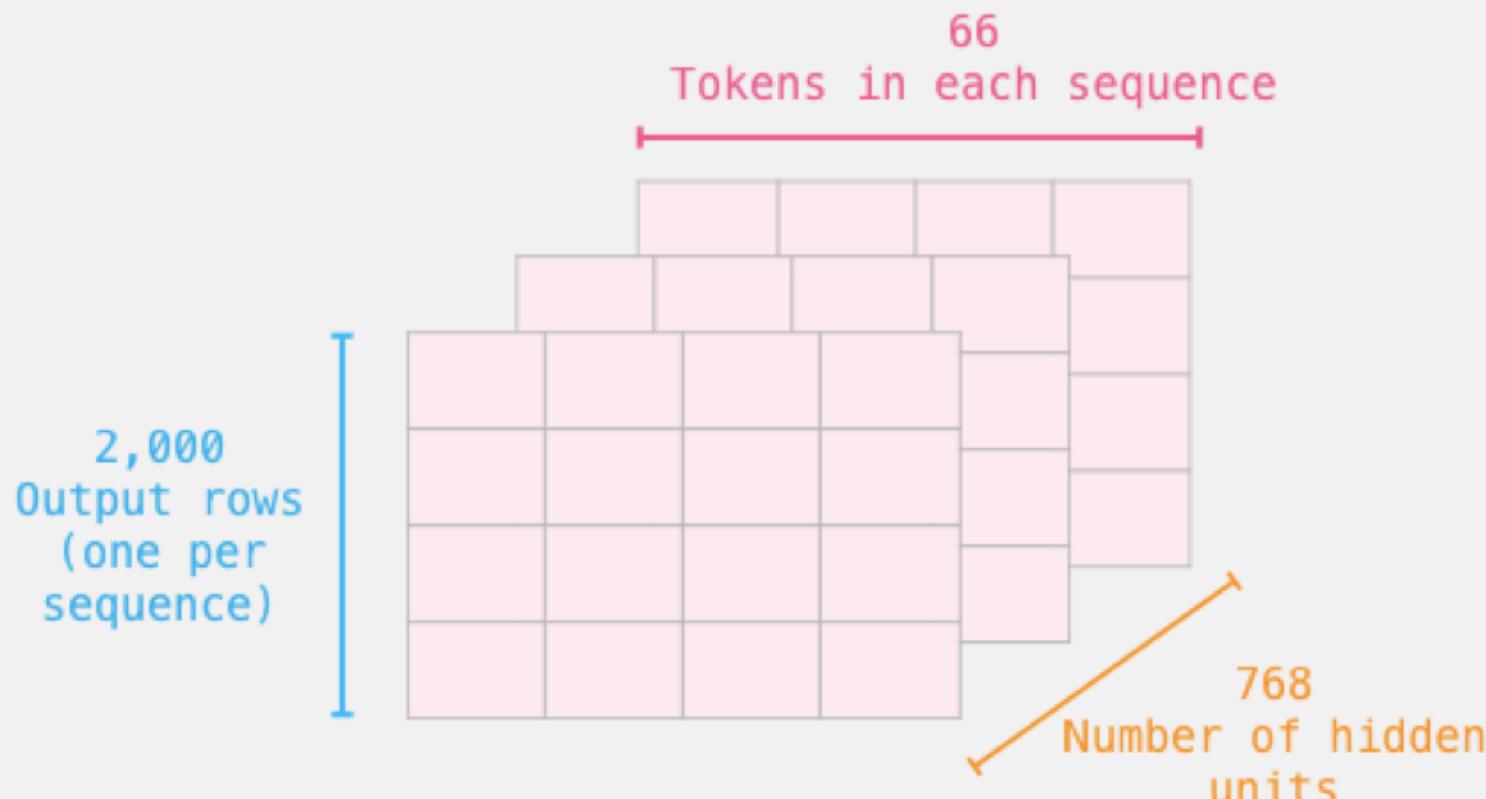
```
input_ids = torch.tensor(np.array(padded))  
last_hidden_states = model(input_ids)
```



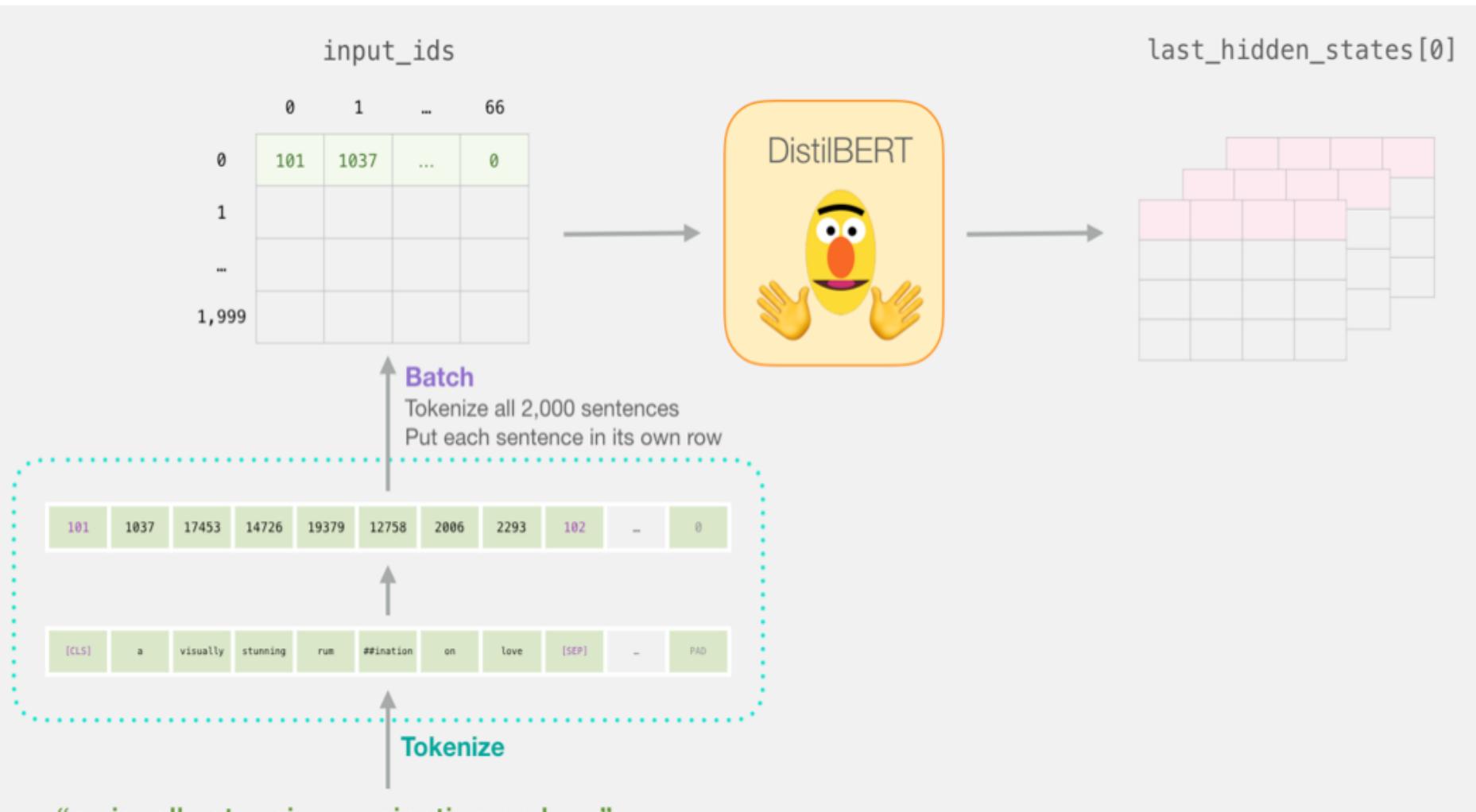
Unpacking the BERT output tensor

`last_hidden_states[0]`

BERT Output Tensor/**predictions**

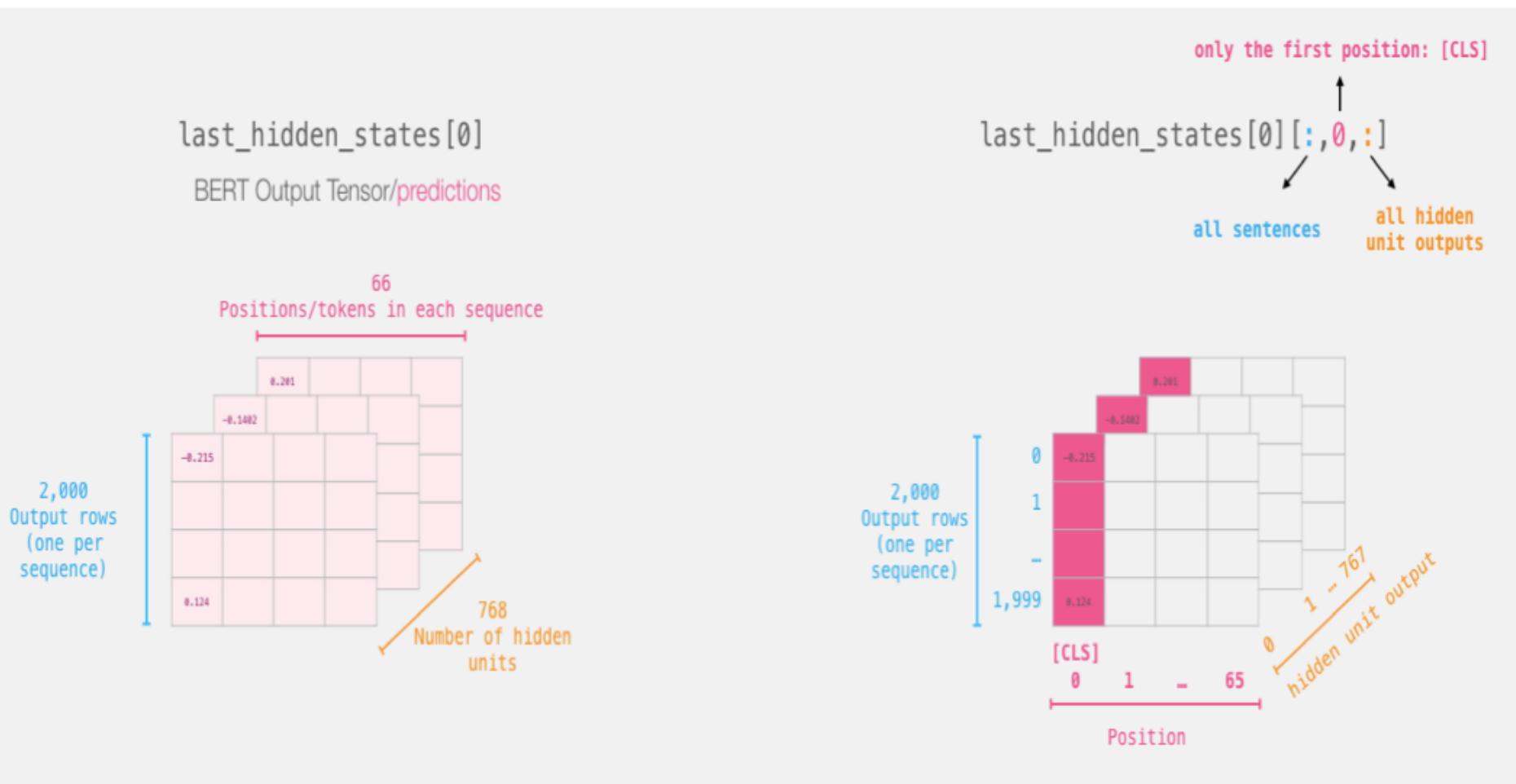


Sentence to last_hidden_state[0]



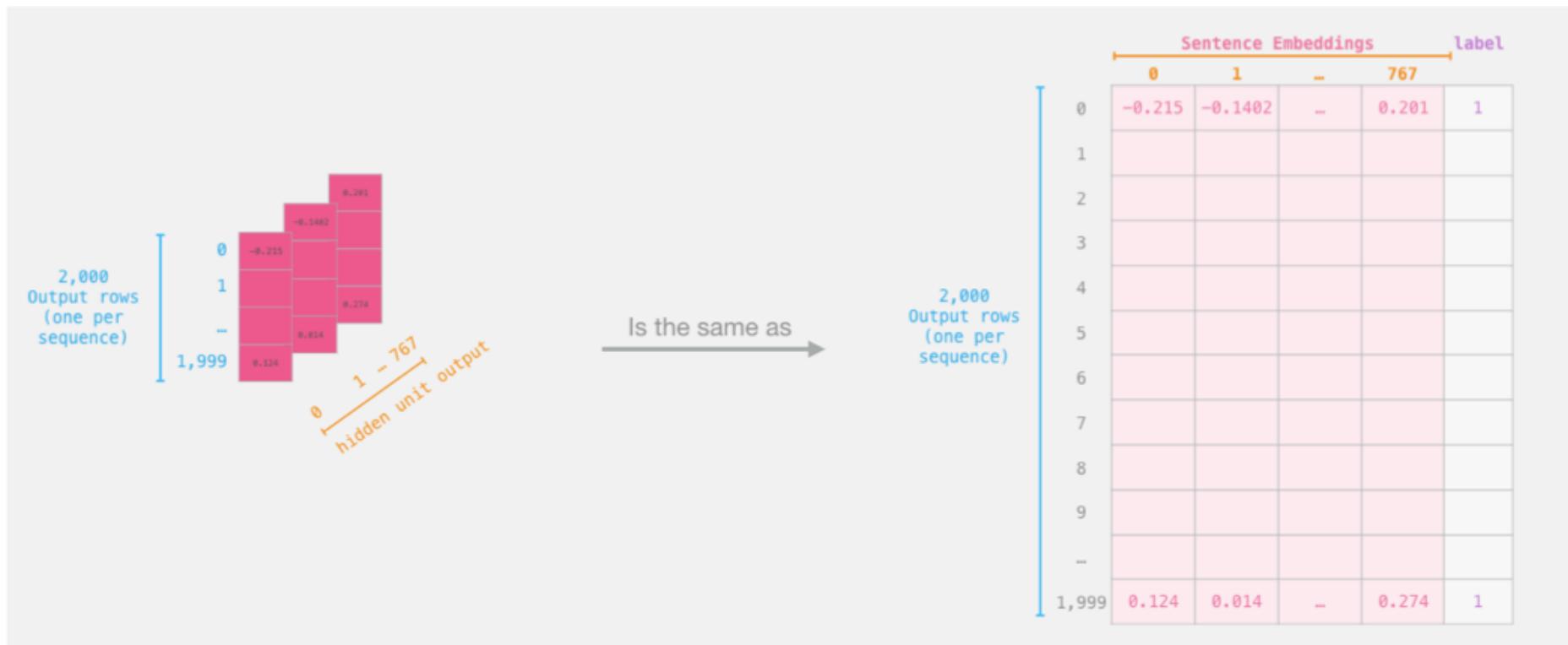
BERT's output for the [CLS] tokens

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



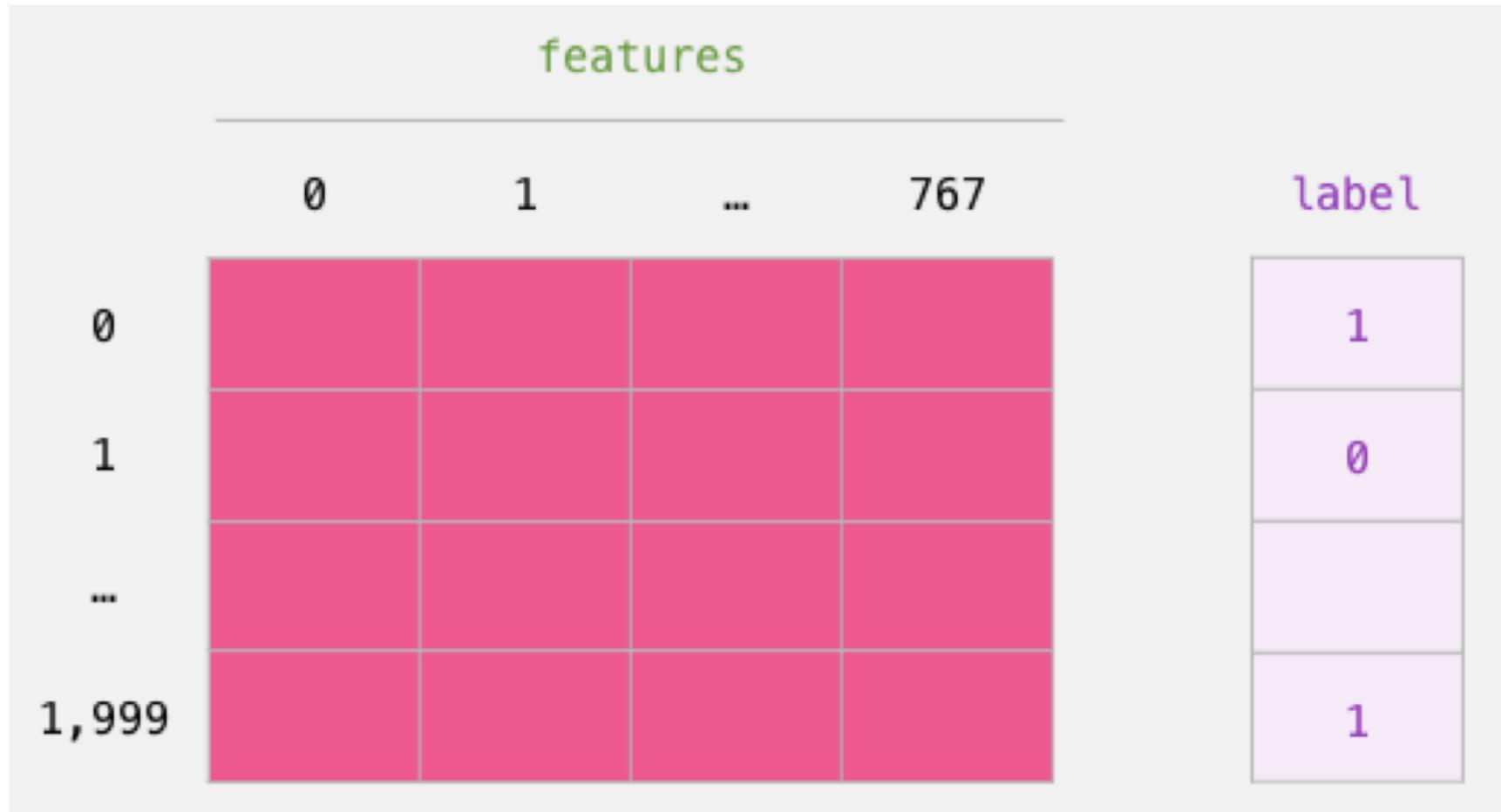
The tensor sliced from BERT's output

Sentence Embeddings



Dataset for Logistic Regression (768 Features)

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

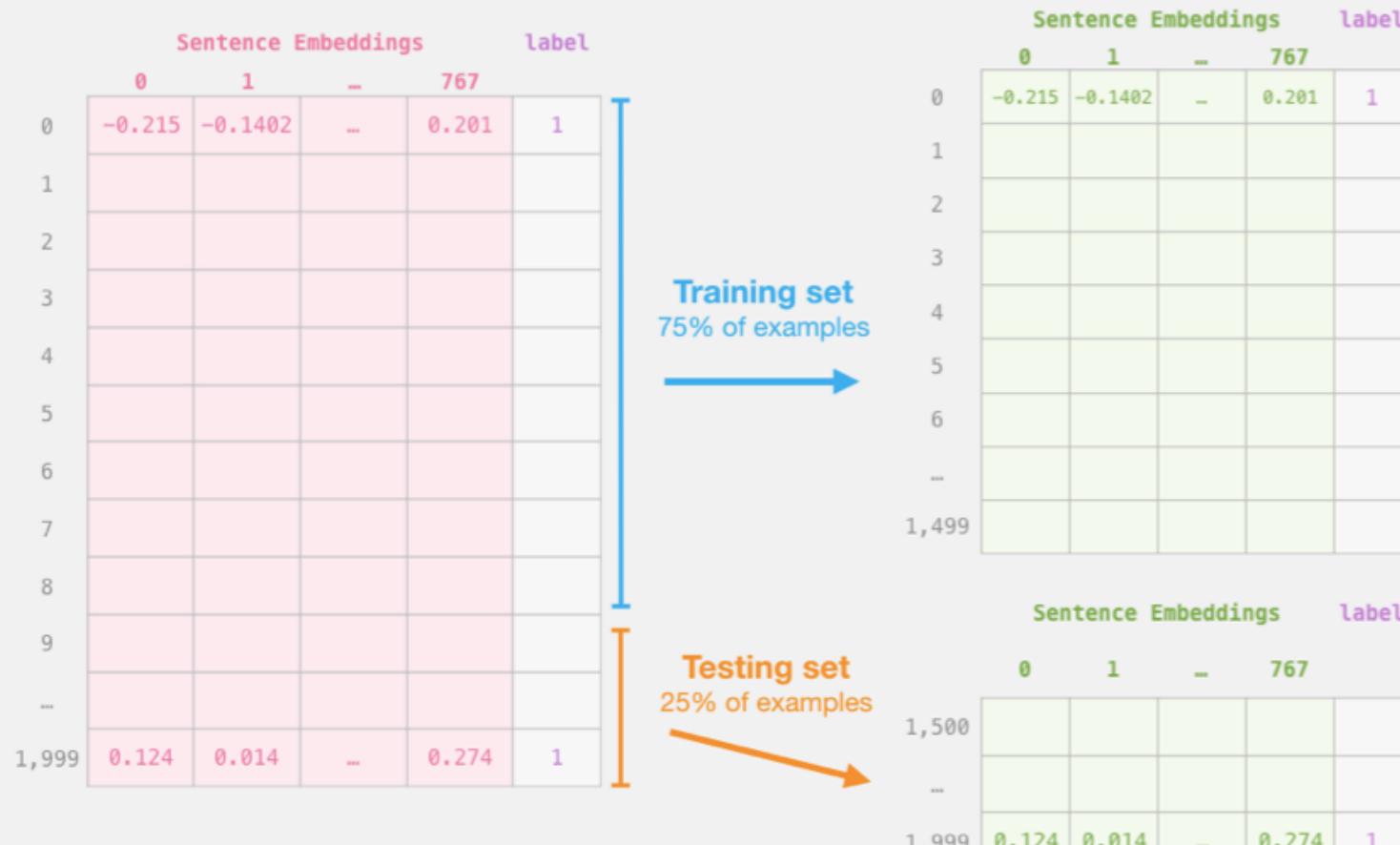


```

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

CO 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 ruminations 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

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

Contents

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
Create a graph of accuracy and loss over time

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](#).

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

Python in Google Colab (Python101)

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

The screenshot shows a Google Colab notebook titled "python101.ipynb". The left sidebar contains a "Table of contents" with various sections like "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", and a "+ Section" button. The main area has a "Text Classification" section expanded, containing a list of references. Below it is a "Text Classification: IMDB Movie Reviews" section with a source link and a code cell for installing TensorFlow.

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

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...
```

Requirement already satisfied: google-pasta>=0.1.8 in /usr/local/lib/python3.6/dist-packages (from tf-nightly)

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

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

The screenshot shows a Google Colab interface with a Jupyter notebook titled "python101.ipynb". The notebook has a "Table of contents" sidebar on the left. The main area displays a section titled "Sentiment Analysis" which includes a bulleted list of sources and a subsection titled "Sentiment Analysis - Unsupervised Lexical". Below this are two code cells:

```
[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/aintpuppython101>

Summary

- AI for Text Analytics: Foundations
 - Processing and Understanding Text
- AI for Text Analytics: Application
 - Sentiment Analysis
 - Text classification

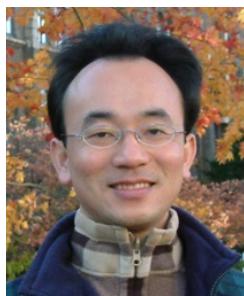
References

- 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>
- The Super Duper NLP Repo, <https://notebooks.quantumstat.com/>
- Min-Yuh Day (2020), Python 101, <https://tinyurl.com/aintpuppython101>

Q & A

人工智慧文本分析 基礎與應用

(Artificial Intelligence for Text Analytics:
Foundations and Applications)



Min-Yuh Day

戴敏育

Associate Professor

副教授

Institute of Information Management, National Taipei University

國立臺北大學 資訊管理研究所

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

2020-09-26

