# Artificial Intelligence for Text Analytics

國立臺北大學
National Taipei University

# Text Classification and Sentiment Analysis

1102AITA05
MBA, IM, NTPU (M5026) (Spring 2022)
Tue 2, 3, 4 (9:10-12:00) (B8F40)

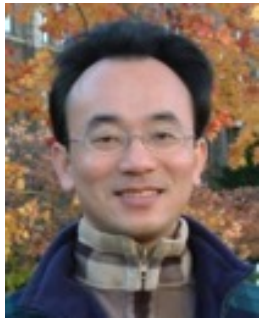
https://meet.google.com/paj-zhhj-mya

aws educate Cloud Ambassador 2020 Cohort

aws academy Accredited Educator

aws certified Cloud Practitioner

aws certified Solutions Architect Associate



**Min-Yuh Day, Ph.D,**
**Associate Professor**

**Institute of Information Management, National Taipei University**

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

2022-03-29

# Syllabus

| Week | Date | Subject/Topics |
|---|---|---|
| 1 | 2022/02/22 | Introduction to Artificial Intelligence for Text Analytics |
| 2 | 2022/03/01 | Foundations of Text Analytics:<br>Natural Language Processing (NLP) |
| 3 | 2022/03/08 | Python for Natural Language Processing |
| 4 | 2022/03/15 | Natural Language Processing with Transformers |
| 5 | 2022/03/22 | Case Study on Artificial Intelligence for Text Analytics I |
| 6 | 2022/03/29 | Text Classification and Sentiment Analysis |

# Syllabus

| Week | Date | Subject/Topics |
|---|---|---|
| 7 | 2022/04/05 | Tomb-Sweeping Day (Holiday, No Classes) |
| 8 | 2022/04/12 | Midterm Project Report |
| 9 | 2022/04/19 | Multilingual Named Entity Recognition (NER),<br>Text Similarity and Clustering |
| 10 | 2022/04/26 | Text Summarization and Topic Models |
| 11 | 2022/05/03 | Text Generation |
| 12 | 2022/05/10 | Case Study on Artificial Intelligence for Text Analytics II |

# Syllabus

| Week | Date | Subject/Topics |
|---|---|---|
| 13 | 2022/05/17 | Question Answering and Dialogue Systems |
| 14 | 2022/05/24 | Deep Learning, Transfer Learning,<br>Zero-Shot, and Few-Shot Learning for Text Analytics |
| 15 | 2022/05/31 | Final Project Report I |
| 16 | 2022/06/07 | Final Project Report II |
| 17 | 2022/06/14 | Self-learning |
| 18 | 2022/06/21 | Self-learning |

# Text Classification and Sentiment Analysis

# Outline

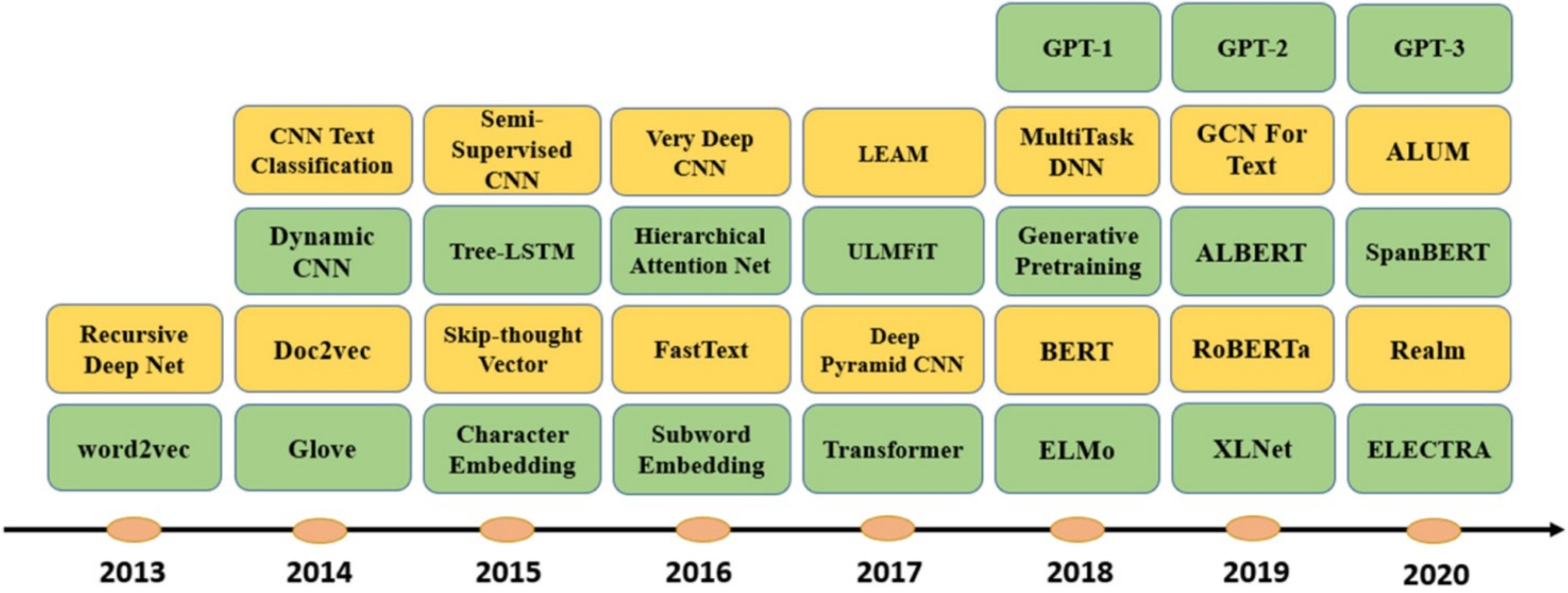
- Text Classification and Sentiment Analysis
  - Dataset
  - Tokenizer
  - Training a Text Classifier
  - Fine-Tuning Transformers

# Text Classification (TC) Tasks

- **Sentiment Analysis**
- **News Categorization**
- **Product Categorization**
- **Topic Analysis**
  - Topic Classification: “customer support” or “ease of use”
- **Natural language inference (NLI)**
  - recognizing textual entailment (RTE)
  - entailment, contradiction, and neutral

Source: Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao (2021), "Deep learning--based text classification: a comprehensive review." ACM Computing Surveys (CSUR) 54, no. 3 (2021): 1-40.

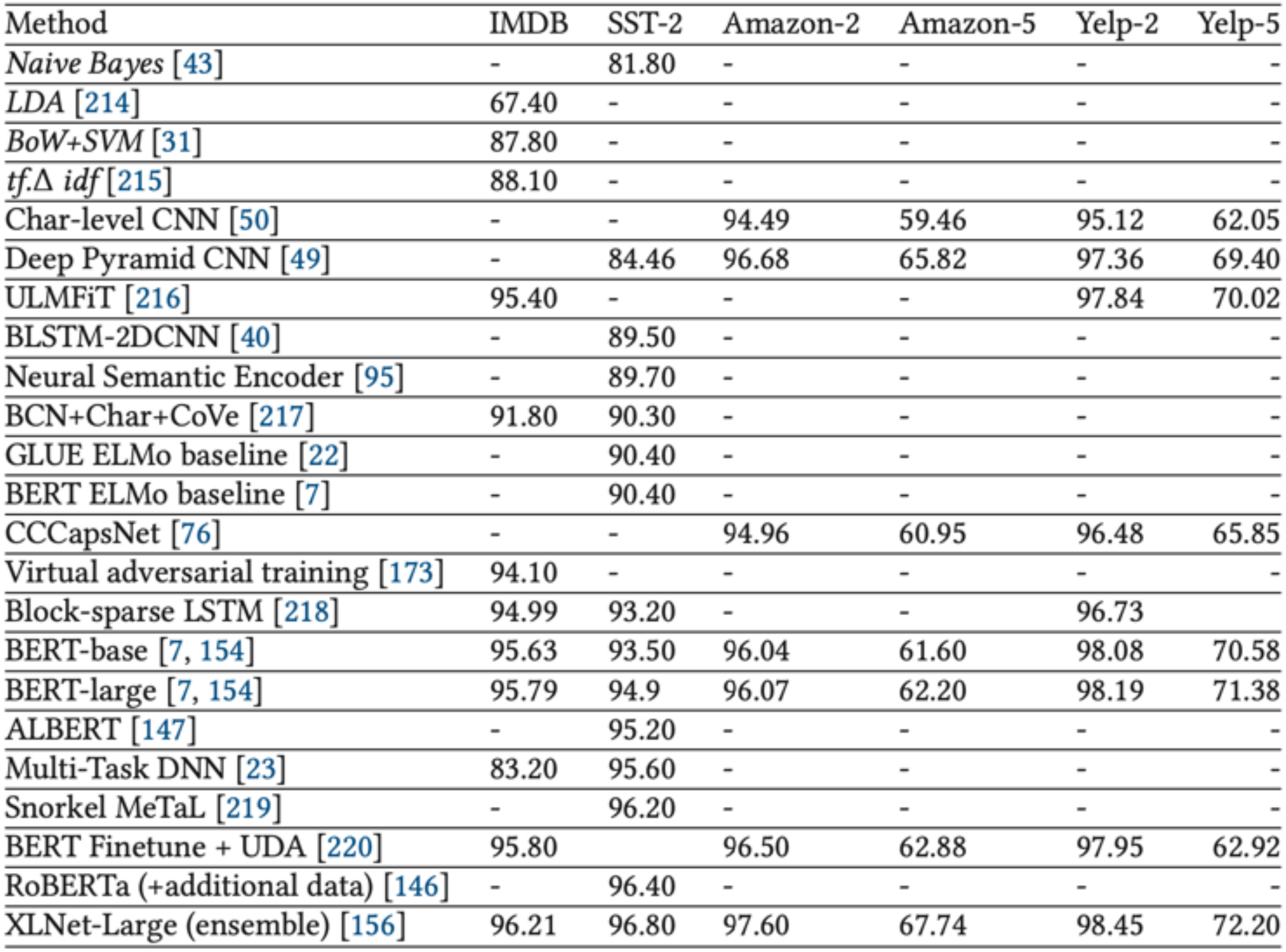
# Deep learning models for text embedding and classification



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | GPT-1 | GPT-2 | GPT-3 |
| | CNN Text Classification | Semi-Supervised CNN | Very Deep CNN | LEAM | MultiTask DNN | GCN For Text | ALUM |
| | Dynamic CNN | Tree-LSTM | Hierarchical Attention Net | ULMFiT | Generative Pretraining | ALBERT | SpanBERT |
| Recursive Deep Net | Doc2vec | Skip-thought Vector | FastText | Deep Pyramid CNN | BERT | RoBERTa | Realm |
| word2vec | Glove | Character Embedding | Subword Embedding | Transformer | ELMo | XLNet | ELECTRA |
| 2013 | 2014 | 2015 | 2016 | 2017 | 2018 | 2019 | 2020 |

Source: Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao (2021), "Deep learning--based text classification: a comprehensive review." ACM Computing Surveys (CSUR) 54, no. 3 (2021): 1-40.

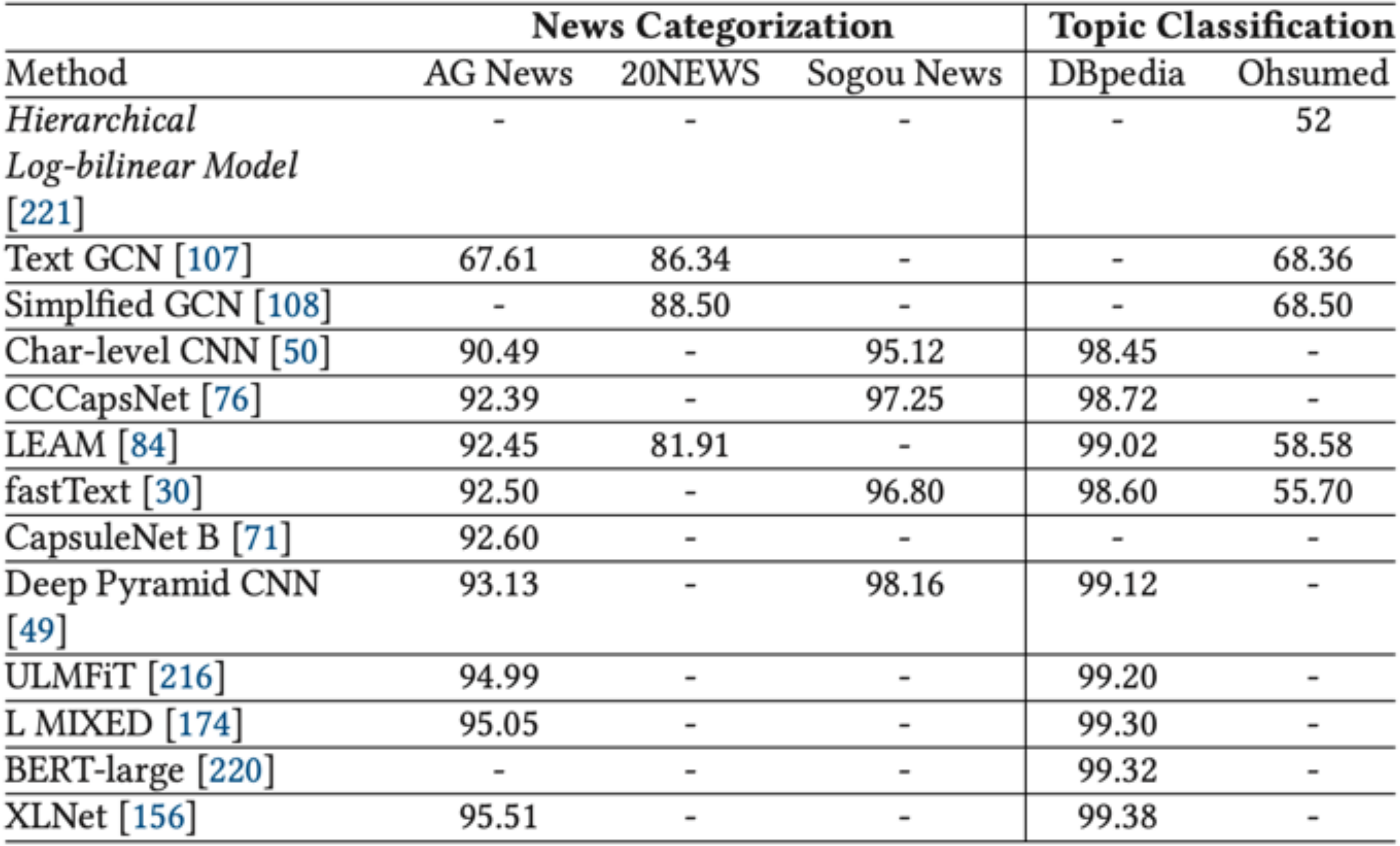
# Text Classification Models on Sentiment Analysis



| Method | IMDB | SST-2 | Amazon-2 | Amazon-5 | Yelp-2 | Yelp-5 |
|---|---|---|---|---|---|---|
| *Naive Bayes* [43] | - | 81.80 | - | - | - | - |
| *LDA* [214] | 67.40 | - | - | - | - | - |
| *BoW+SVM* [31] | 87.80 | - | - | - | - | - |
| *tf.Δ idf* [215] | 88.10 | - | - | - | - | - |
| Char-level CNN [50] | - | - | 94.49 | 59.46 | 95.12 | 62.05 |
| Deep Pyramid CNN [49] | - | 84.46 | 96.68 | 65.82 | 97.36 | 69.40 |
| ULMFiT [216] | 95.40 | - | - | - | 97.84 | 70.02 |
| BLSTM-2DCNN [40] | - | 89.50 | - | - | - | - |
| Neural Semantic Encoder [95] | - | 89.70 | - | - | - | - |
| BCN+Char+CoVe [217] | 91.80 | 90.30 | - | - | - | - |
| GLUE ELMo baseline [22] | - | 90.40 | - | - | - | - |
| BERT ELMo baseline [7] | - | 90.40 | - | - | - | - |
| CCCapsNet [76] | - | - | 94.96 | 60.95 | 96.48 | 65.85 |
| Virtual adversarial training [173] | 94.10 | - | - | - | - | - |
| Block-sparse LSTM [218] | 94.99 | 93.20 | - | - | 96.73 |  |
| BERT-base [7, 154] | 95.63 | 93.50 | 96.04 | 61.60 | 98.08 | 70.58 |
| BERT-large [7, 154] | 95.79 | 94.9 | 96.07 | 62.20 | 98.19 | 71.38 |
| ALBERT [147] | - | 95.20 | - | - | - | - |
| Multi-Task DNN [23] | 83.20 | 95.60 | - | - | - | - |
| Snorkel MeTaL [219] | - | 96.20 | - | - | - | - |
| BERT Finetune + UDA [220] | 95.80 |  | 96.50 | 62.88 | 97.95 | 62.92 |
| RoBERTa (+additional data) [146] | - | 96.40 | - | - | - | - |
| XLNet-Large (ensemble) [156] | 96.21 | 96.80 | 97.60 | 67.74 | 98.45 | 72.20 |

Source: Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao (2021), "Deep learning--based text classification: a comprehensive review." ACM Computing Surveys (CSUR) 54, no. 3 (2021): 1-40.

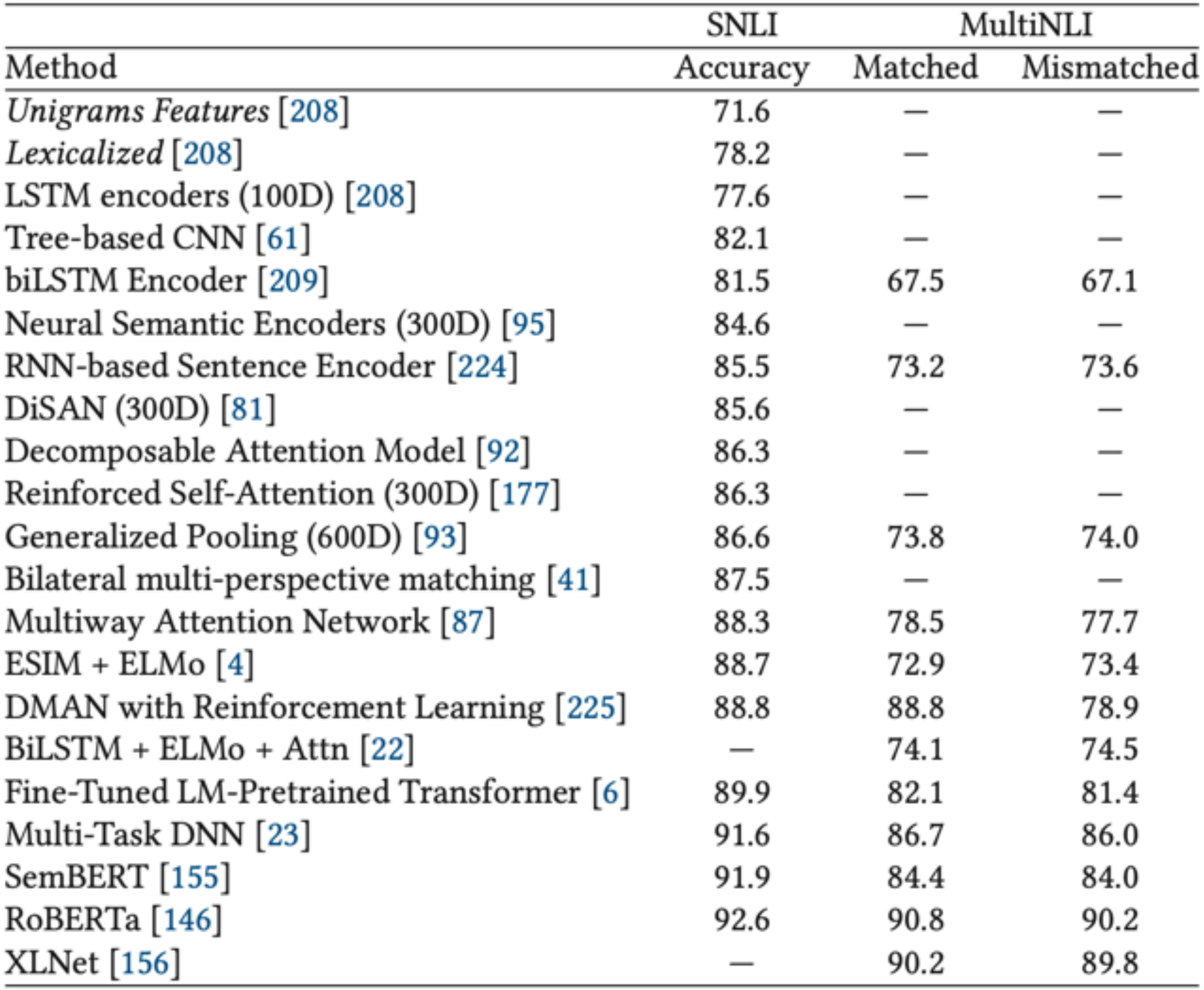
# Classification Models on News Categorization, and Topic Classification



| | News Categorization | | | Topic Classification | |
|---|---|---|---|---|---|
| Method | AG News | 20NEWS | Sogou News | DBpedia | Ohsumed |
| *Hierarchical Log-bilinear Model* [221] | - | - | - | - | 52 |
| Text GCN [107] | 67.61 | 86.34 | - | - | 68.36 |
| Simplfied GCN [108] | - | 88.50 | - | - | 68.50 |
| Char-level CNN [50] | 90.49 | - | 95.12 | 98.45 | - |
| CCCapsNet [76] | 92.39 | - | 97.25 | 98.72 | - |
| LEAM [84] | 92.45 | 81.91 | - | 99.02 | 58.58 |
| fastText [30] | 92.50 | - | 96.80 | 98.60 | 55.70 |
| CapsuleNet B [71] | 92.60 | - | - | - | - |
| Deep Pyramid CNN [49] | 93.13 | - | 98.16 | 99.12 | - |
| ULMFiT [216] | 94.99 | - | - | 99.20 | - |
| L MIXED [174] | 95.05 | - | - | 99.30 | - |
| BERT-large [220] | - | - | - | 99.32 | - |
| XLNet [156] | 95.51 | - | - | 99.38 | - |

Source: Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao (2021), "Deep learning--based text classification: a comprehensive review." ACM Computing Surveys (CSUR) 54, no. 3 (2021): 1-40.

# Classification Models on Natural Language Inference (NLI)

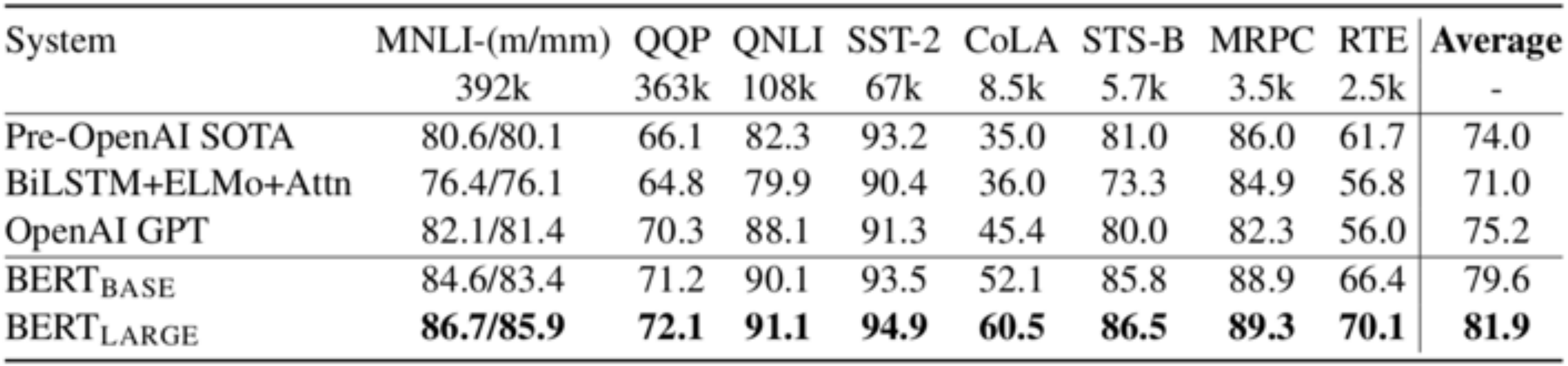


|  | SNLI | MultiNLI |  |
|---|---|---|---|
| Method | Accuracy | Matched | Mismatched |
| *Unigrams Features* [208] | 71.6 | — | — |
| *Lexicalized* [208] | 78.2 | — | — |
| LSTM encoders (100D) [208] | 77.6 | — | — |
| Tree-based CNN [61] | 82.1 | — | — |
| biLSTM Encoder [209] | 81.5 | 67.5 | 67.1 |
| Neural Semantic Encoders (300D) [95] | 84.6 | — | — |
| RNN-based Sentence Encoder [224] | 85.5 | 73.2 | 73.6 |
| DiSAN (300D) [81] | 85.6 | — | — |
| Decomposable Attention Model [92] | 86.3 | — | — |
| Reinforced Self-Attention (300D) [177] | 86.3 | — | — |
| Generalized Pooling (600D) [93] | 86.6 | 73.8 | 74.0 |
| Bilateral multi-perspective matching [41] | 87.5 | — | — |
| Multiway Attention Network [87] | 88.3 | 78.5 | 77.7 |
| ESIM + ELMo [4] | 88.7 | 72.9 | 73.4 |
| DMAN with Reinforcement Learning [225] | 88.8 | 88.8 | 78.9 |
| BiLSTM + ELMo + Attn [22] | — | 74.1 | 74.5 |
| Fine-Tuned LM-Pretrained Transformer [6] | 89.9 | 82.1 | 81.4 |
| Multi-Task DNN [23] | 91.6 | 86.7 | 86.0 |
| SemBERT [155] | 91.9 | 84.4 | 84.0 |
| RoBERTa [146] | 92.6 | 90.8 | 90.2 |
| XLNet [156] | — | 90.2 | 89.8 |

Source: Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao (2021), "Deep learning--based text classification: a comprehensive review." ACM Computing Surveys (CSUR) 54, no. 3 (2021): 1-40.

# General Language Understanding Evaluation (GLUE) benchmark

# GLUE Test results



| System | MNLI-(m/mm) | QQP | QNLI | SST-2 | CoLA | STS-B | MRPC | RTE | **Average** |
|---|---|---|---|---|---|---|---|---|---|
| | 392k | 363k | 108k | 67k | 8.5k | 5.7k | 3.5k | 2.5k | - |
| Pre-OpenAI SOTA | 80.6/80.1 | 66.1 | 82.3 | 93.2 | 35.0 | 81.0 | 86.0 | 61.7 | 74.0 |
| BiLSTM+ELMo+Attn | 76.4/76.1 | 64.8 | 79.9 | 90.4 | 36.0 | 73.3 | 84.9 | 56.8 | 71.0 |
| OpenAI GPT | 82.1/81.4 | 70.3 | 88.1 | 91.3 | 45.4 | 80.0 | 82.3 | 56.0 | 75.2 |
| BERT$_{\text{BASE}}$ | 84.6/83.4 | 71.2 | 90.1 | 93.5 | 52.1 | 85.8 | 88.9 | 66.4 | 79.6 |
| BERT$_{\text{LARGE}}$ | **86.7/85.9** | **72.1** | **91.1** | **94.9** | **60.5** | **86.5** | **89.3** | **70.1** | **81.9** |

**MNLI**: Multi-Genre Natural Language Inference
**QQP**: Quora Question Pairs
**QNLI**: Question Natural Language Inference
**SST-2**: The Stanford Sentiment Treebank
**CoLA**: The Corpus of Linguistic Acceptability
**STS-B**:The Semantic Textual Similarity Benchmark
**MRPC**: Microsoft Research Paraphrase Corpus
**RTE**: Recognizing Textual Entailment

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

# Emotions

| ☺ | ☹ |
|---|---|
| Love | Anger |
| Joy | Sadness |
| Surprise | Fear |

Source: Bing Liu (2011) , “Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data,” Springer, 2nd Edition,

# Example of Opinion: review segment on iPhone

"I bought an iPhone a few days ago.

It was such a nice phone.

The touch screen was really cool.

The voice quality was clear too.

However, my mother was mad with me as I did not tell her before I bought it.

She also thought the phone was too expensive, and wanted me to return it to the shop. … "

Source: Bing Liu (2011) , "Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data," Springer, 2nd Edition,

# Example of Opinion: review segment on iPhone

"(1) I bought an iPhone a few days ago.

(2) It was such a nice phone.

(3) The touch screen was really cool.

(4) The voice quality was clear too.

(5) However, my mother was mad with me as I did not tell her before I bought it.

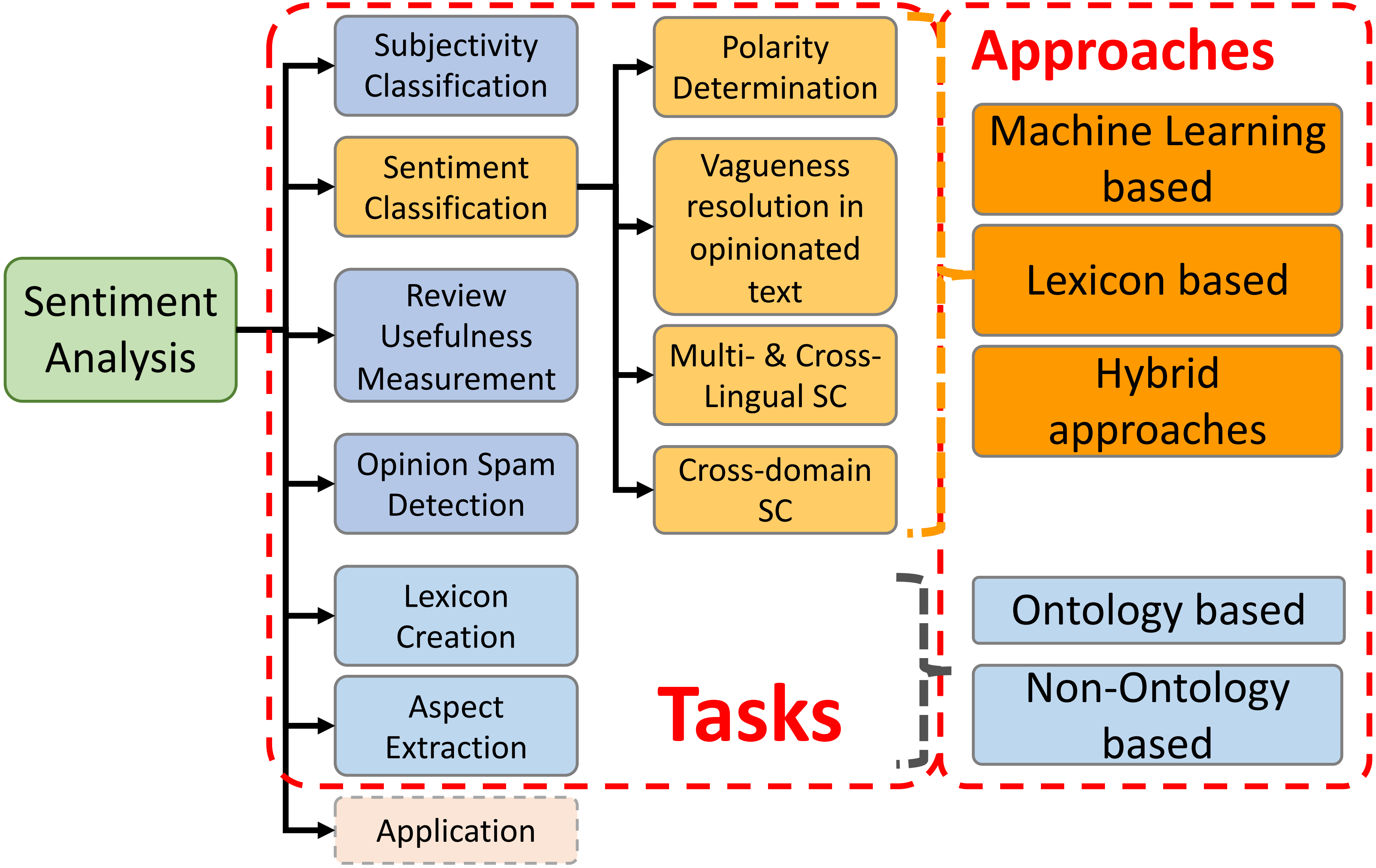
(6) She also thought the phone was too expensive, and wanted me to return it to the shop. … "

☺ +Positive Opinion

☹ -Negative Opinion

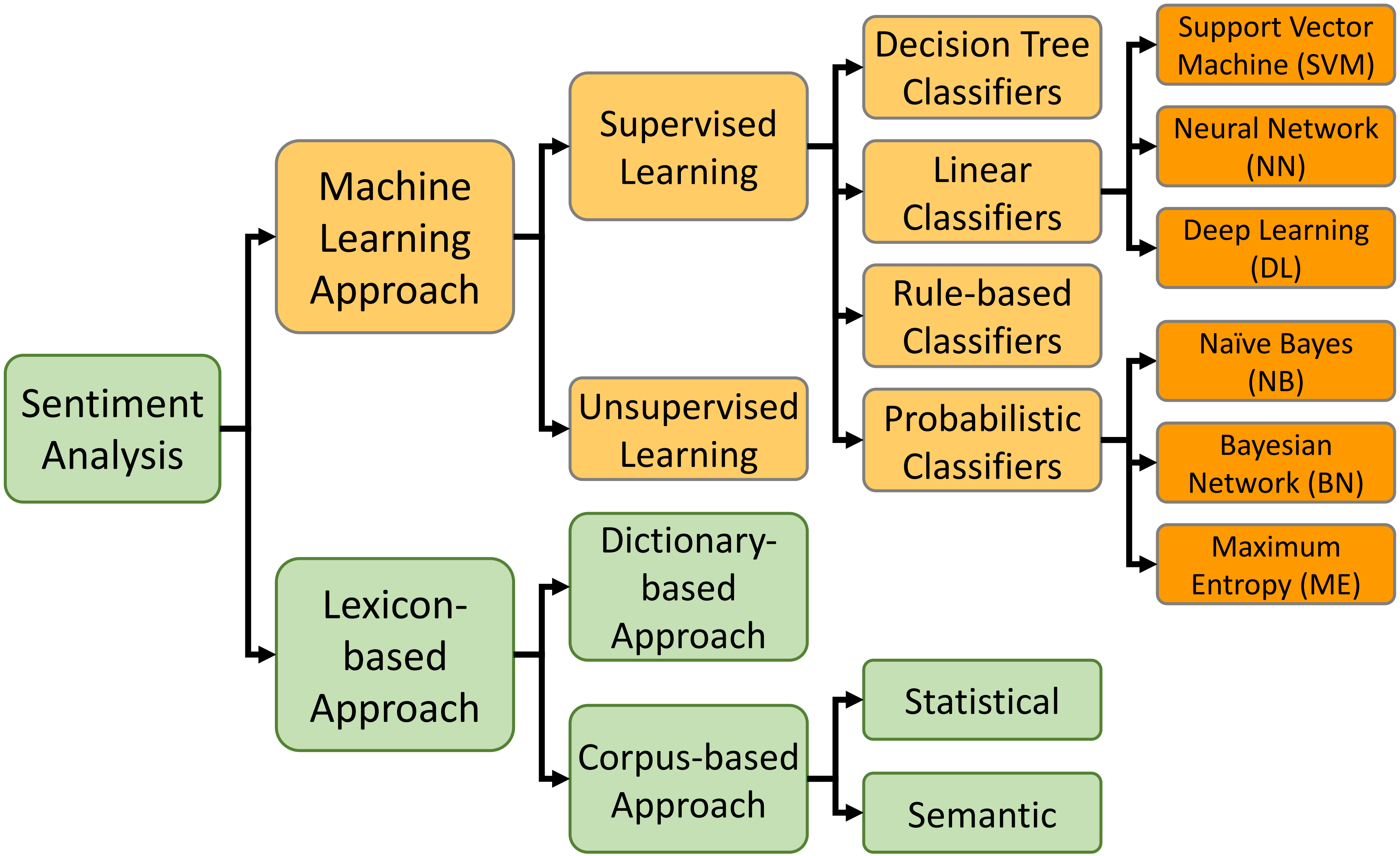
Source: Bing Liu (2011) , "Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data," Springer, 2nd Edition,

# Sentiment Analysis



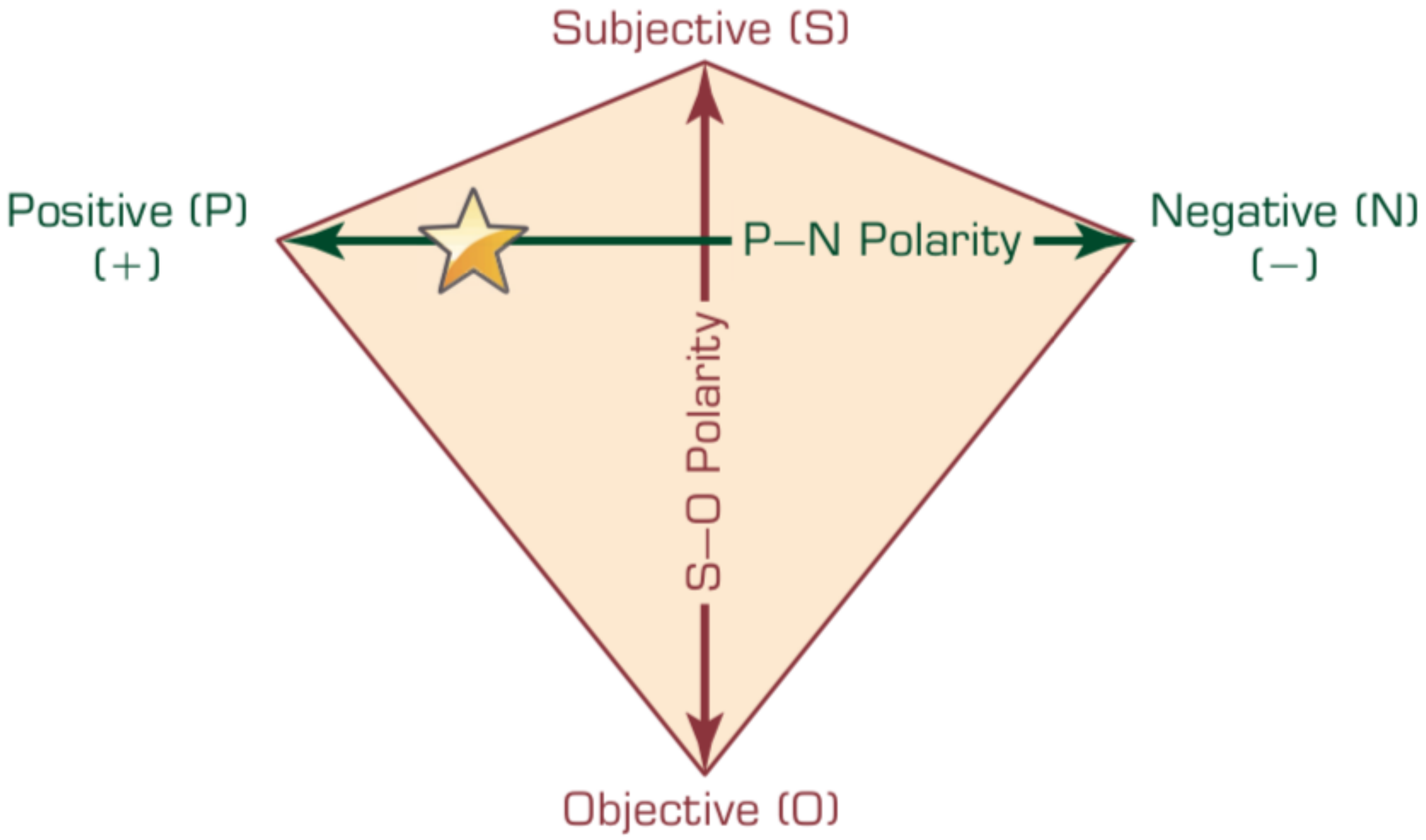
Source: Kumar Ravi and Vadlamani Ravi (2015), "A survey on opinion mining and sentiment analysis: tasks, approaches and applications." Knowledge-Based Systems, 89, pp.14-46.

# Sentiment Classification Techniques



Source: Jesus Serrano-Guerrero, Jose A. Olivas, Francisco P. Romero, and Enrique Herrera-Viedma (2015), "Sentiment analysis: A review and comparative analysis of web services," Information Sciences, 311, pp. 18-38.

# P–N Polarity and S–O Polarity Relationship



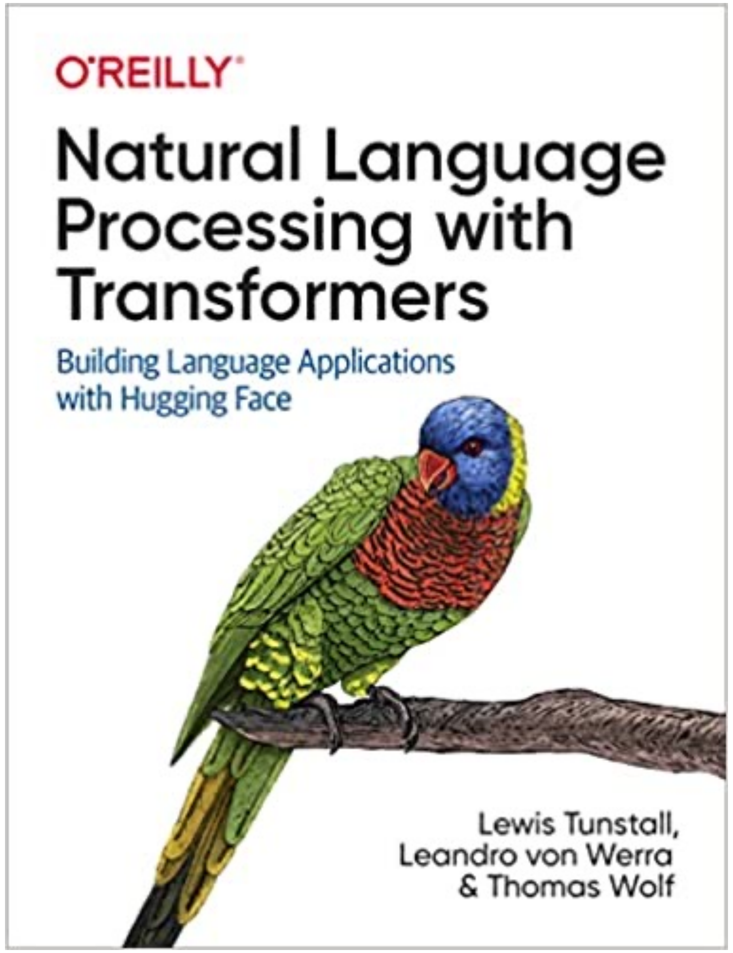
Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022),

# Natural Language Processing with Transformers:

Building Language Applications with Hugging Face,

O'Reilly Media.



O'REILLY®

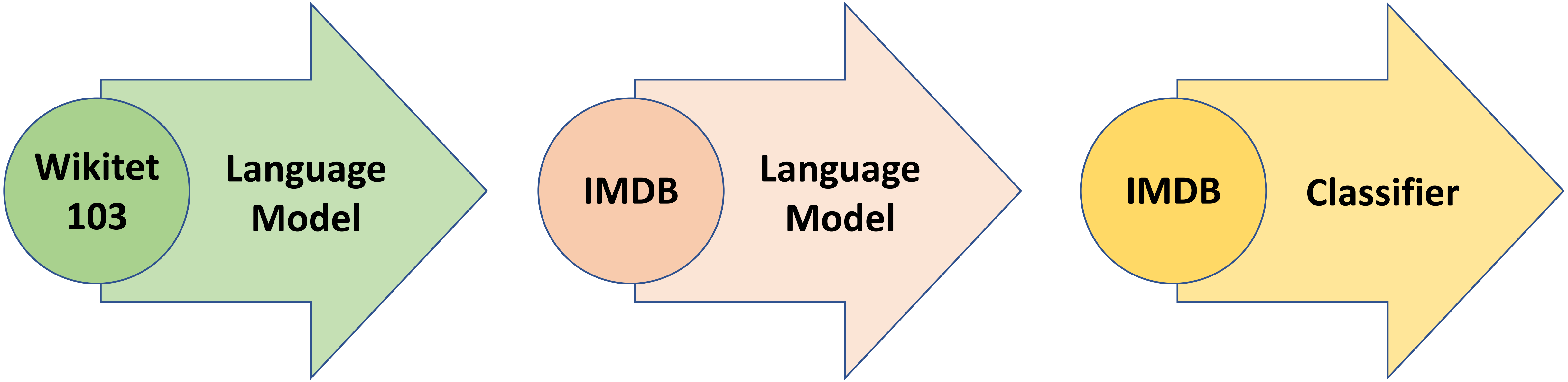
Natural Language Processing with Transformers

Building Language Applications with Hugging Face

Lewis Tunstall,
Leandro von Werra
& Thomas Wolf

Source: https://www.amazon.com/Natural-Language-Processing-Transformers-Applications/dp/1098103246

# ULMFiT: 3 Steps
# Transfer Learning in NLP



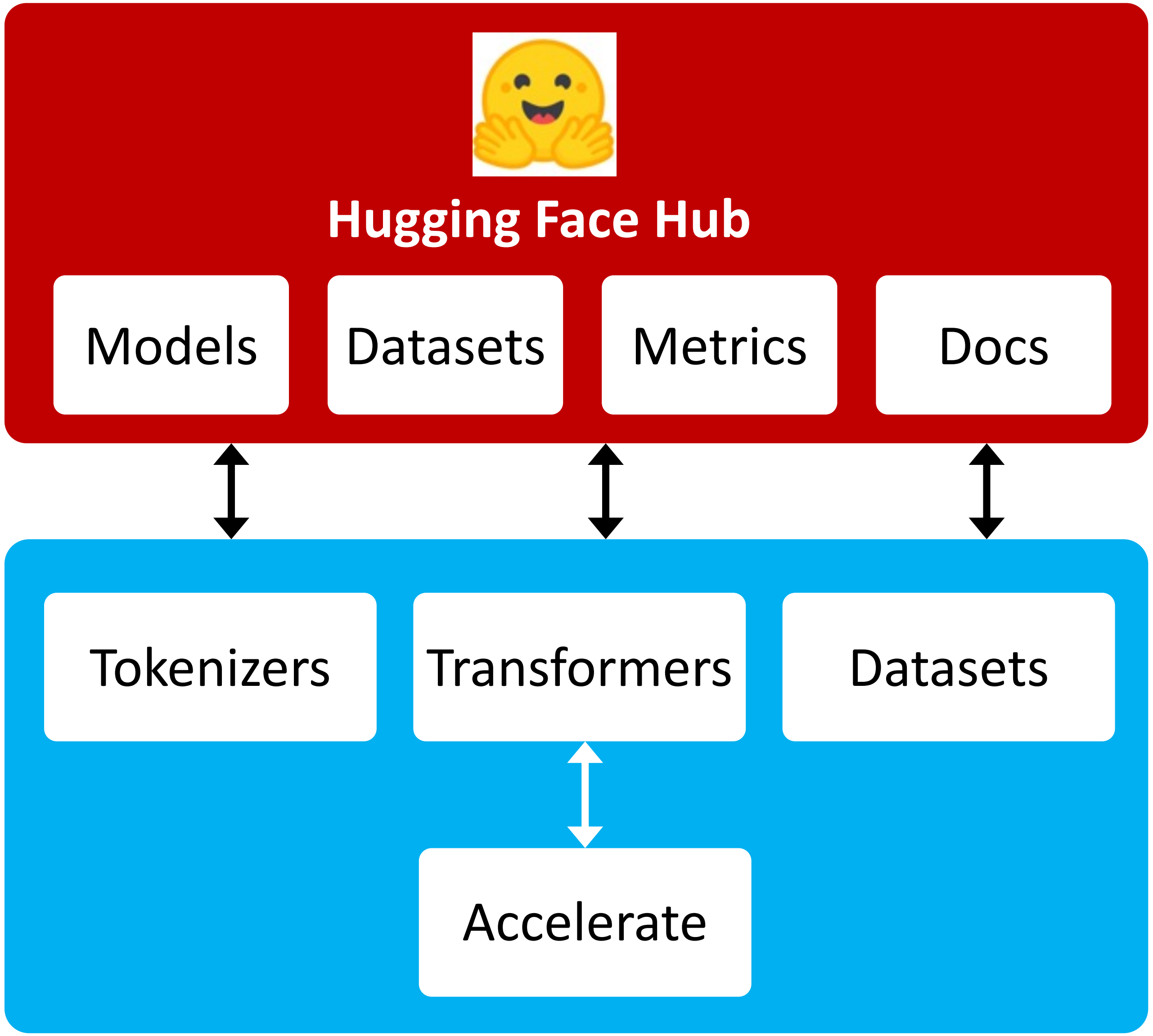
1. Pretraining

2. Domain adaptation

3. Fine-tuning

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

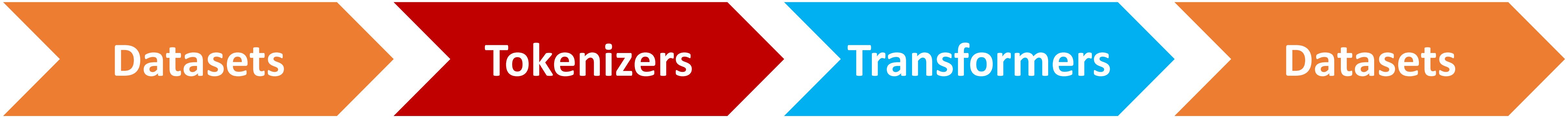
# An overview of the Hugging Face Ecosystem



Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

# A typical pipeline for training transformer models

with the  Datasets,  Tokenizers, and  Transformers libraries



Datasets

Tokenizers

Transformers

Datasets

Load and process datasets

Tokenize input texts

Load models, train and infer

Load metrics evaluate models

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers:  Building Language Applications with Hugging Face,  O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

# NLP with Transformers

```
!git clone https://github.com/nlp-with-transformers/notebooks.git
%cd notebooks
from install import *
install_requirements()
```

```
from utils import *
setup_chapter()
```

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

# Text Classification

```
text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
from your online store in Germany. Unfortunately, when I opened the package, \
I discovered to my horror that I had been sent an action figure of Megatron \
instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
dilemma. To resolve the issue, I demand an exchange of Megatron for the \
Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
```

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

# Text Classification

```
text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
from your online store in Germany. Unfortunately, when I opened the package, \
I discovered to my horror that I had been sent an action figure of Megatron \
instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
dilemma. To resolve the issue, I demand an exchange of Megatron for the \
Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
```

```
from transformers import pipeline
classifier = pipeline("text-classification")
```

```
import pandas as pd
outputs = classifier(text)
pd.DataFrame(outputs)
```

```
      label      score
0  NEGATIVE   0.901546
```

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

# Text Classification

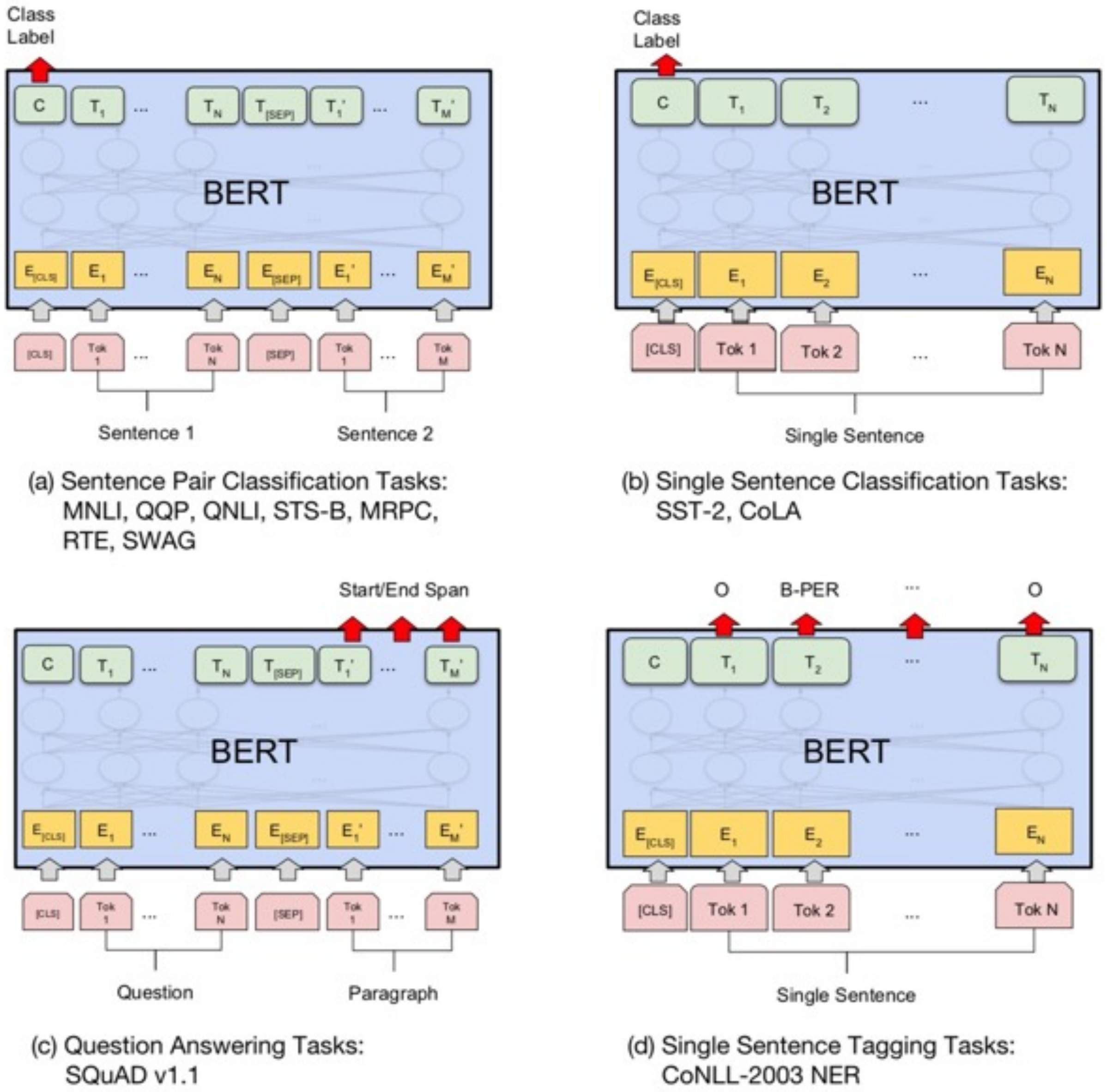
```python
from transformers import pipeline
classifier = pipeline("text-classification")
```

```python
import pandas as pd
outputs = classifier(text)
pd.DataFrame(outputs)
```

|  | label | score |
|---|---|---|
| 0 | NEGATIVE | 0.901546 |

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

# Fine-tuning BERT on NLP Tasks



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

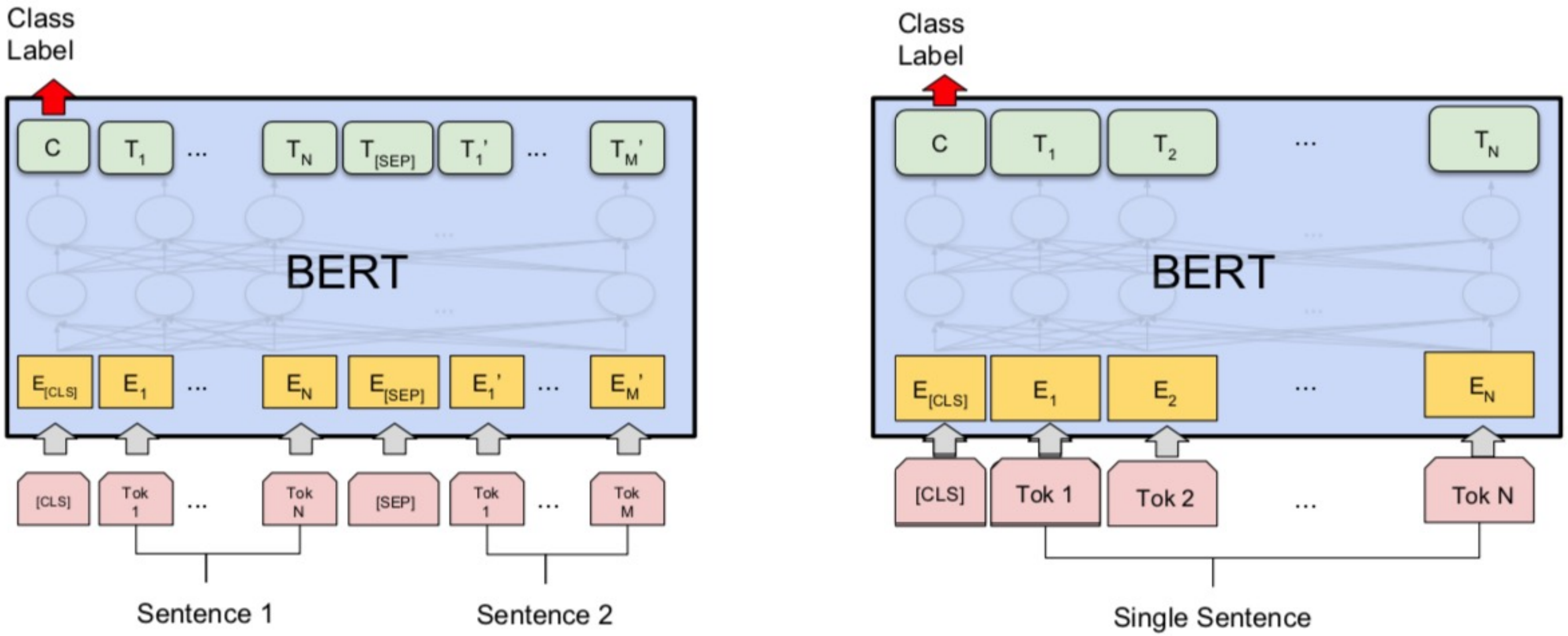
(b) Single Sentence Classification Tasks: SST-2, CoLA

(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

# BERT Sequence-level tasks

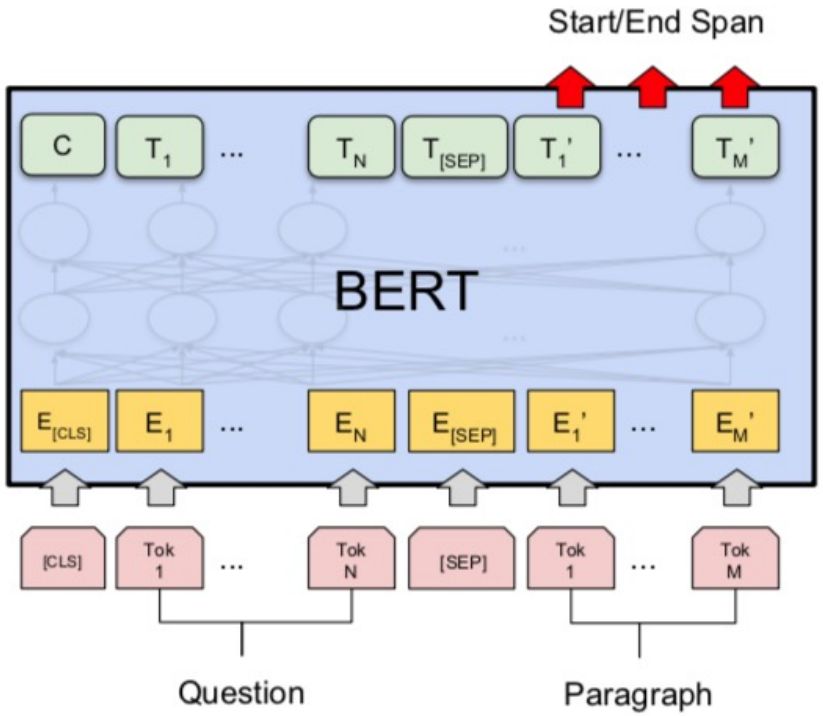


(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

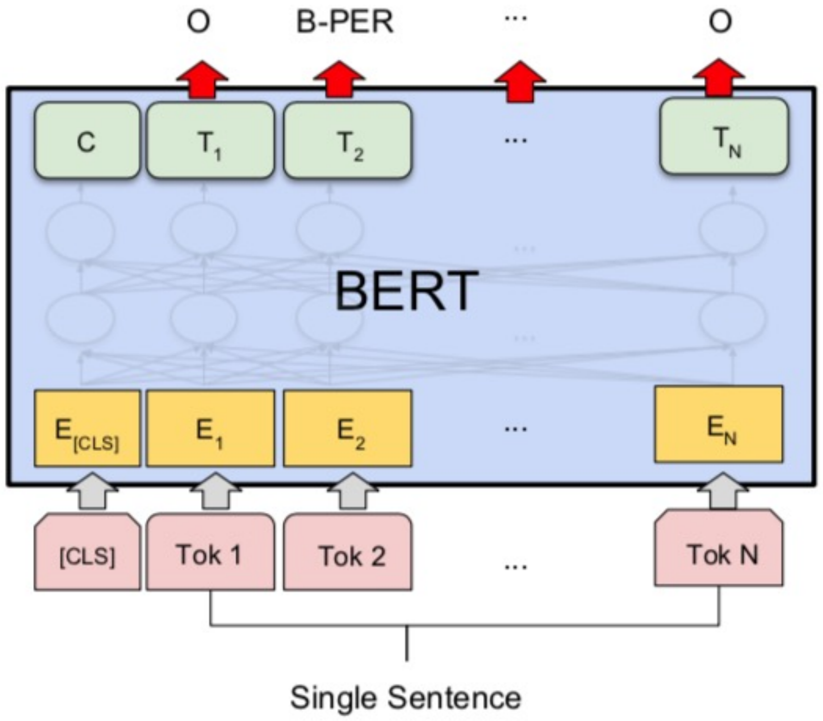
(b) Single Sentence Classification Tasks: SST-2, CoLA

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

# BERT Token-level tasks



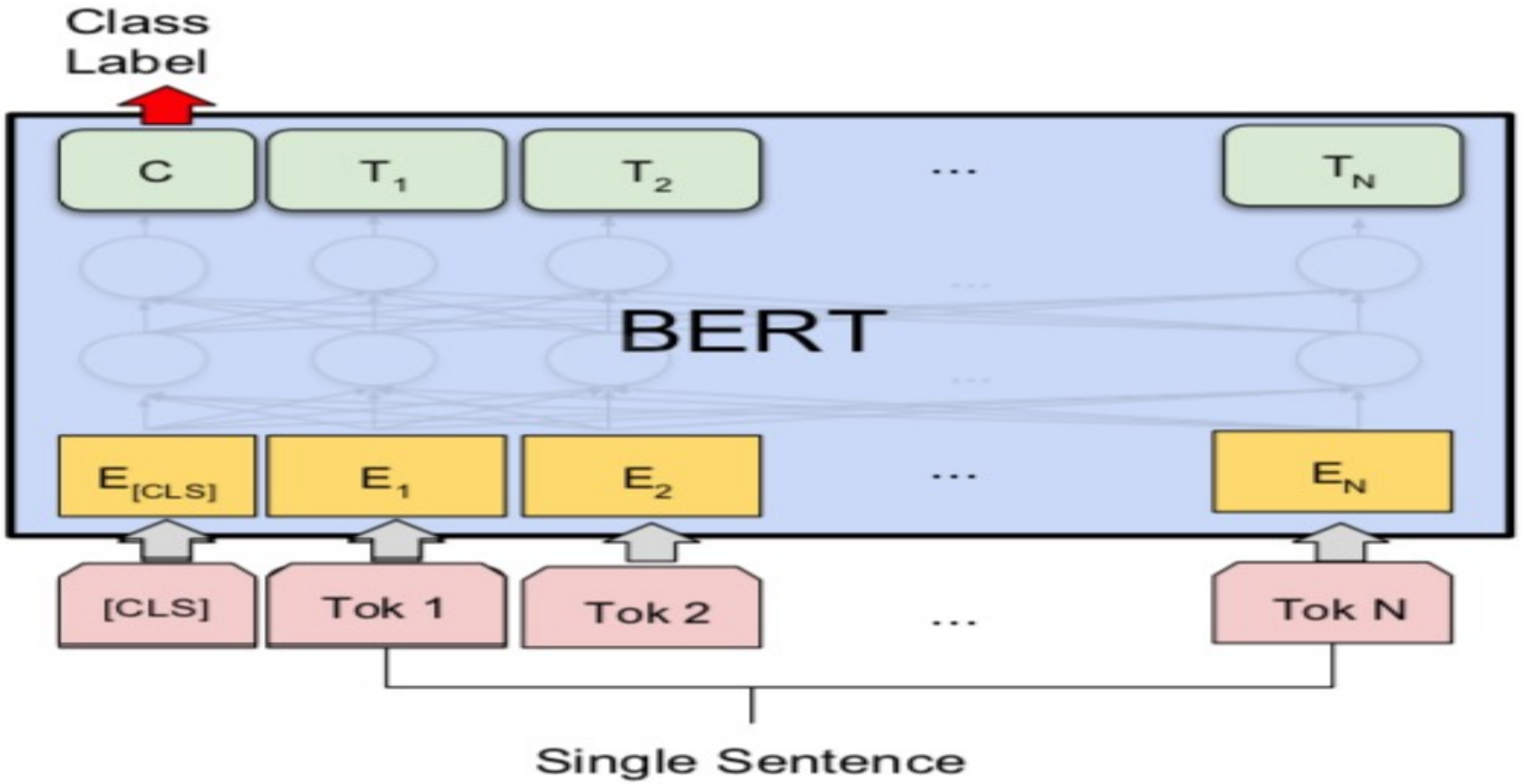
(c) Question Answering Tasks:
SQuAD v1.1



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

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

# Sentiment Analysis:
# Single Sentence Classification



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

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

# Character Tokenization

```python
text = "Tokenizing text is a core task of NLP."
tokenized_text = list(text)
print(tokenized_text)
```

```
['T', 'o', 'k', 'e', 'n', 'i', 'z', 'i', 'n', 'g', ' ', 't', 'e', 'x', 't', ' ',
'i', 's', ' ', 'a', ' ', 'c', 'o', 'r', 'e', ' ', 't', 'a', 's', 'k', ' ', 'o',
'f', ' ', 'N', 'L', 'P', '.']
```

```python
token2idx = {ch: idx for idx, ch in enumerate(sorted(set(tokenized_text)))}
print(token2idx)
```

```
{' ': 0, '.': 1, 'L': 2, 'N': 3, 'P': 4, 'T': 5, 'a': 6, 'c': 7, 'e': 8, 'f': 9,
'g': 10, 'i': 11, 'k': 12, 'n': 13, 'o': 14, 'r': 15, 's': 16, 't': 17, 'x': 18,
'z': 19}
```

```python
input_ids = [token2idx[token] for token in tokenized_text]
print(input_ids)
```

```
[5, 14, 12, 8, 13, 11, 19, 11, 13, 10, 0, 17, 8, 18, 17, 0, 11, 16, 0, 6, 0, 7,
14, 15, 8, 0, 17, 6, 16, 12, 0, 14, 9, 0, 3, 2, 4, 1]
```

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

# Word Tokenization

```
text = "Tokenizing text is a core task of NLP."
tokenized_text = text.split()
print(tokenized_text)
```

```
['Tokenizing', 'text', 'is', 'a', 'core', 'task', 'of', 'NLP.']
```

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

# Subword Tokenization

```
from transformers import AutoTokenizer
model_ckpt = "distilbert-base-uncased"
tokenizer = AutoTokenizer.from_pretrained(model_ckpt)
```

```
text = "Tokenizing text is a core task of NLP."
encoded_text = tokenizer(text)
print(encoded_text)
```

```
{'input_ids': [101, 19204, 6026, 3793, 2003, 1037, 4563, 4708, 1997, 17953, 2361,
1012, 102], 'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}
```

```
tokens = tokenizer.convert_ids_to_tokens(encoded_text.input_ids)
print(tokens)
```

```
['[CLS]', 'token', '##izing', 'text', 'is', 'a', 'core', 'task', 'of', 'nl',
'##p', '.', '[SEP]']
```

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

# Subword Tokenization

```
print(tokenizer.convert_tokens_to_string(tokens))
```

```
[CLS] tokenizing text is a core task of nlp. [SEP]
```

```
tokenizer.vocab_size
```

```
30522
```

```
tokenizer.model_max_length
```

```
512
```

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

# Tokenizing the Whole Dataset

```
def tokenize(batch):
      return tokenizer(batch["text"], padding=True, truncation=True)
```

```
print(tokenize(emotions["train"][:2]))
```

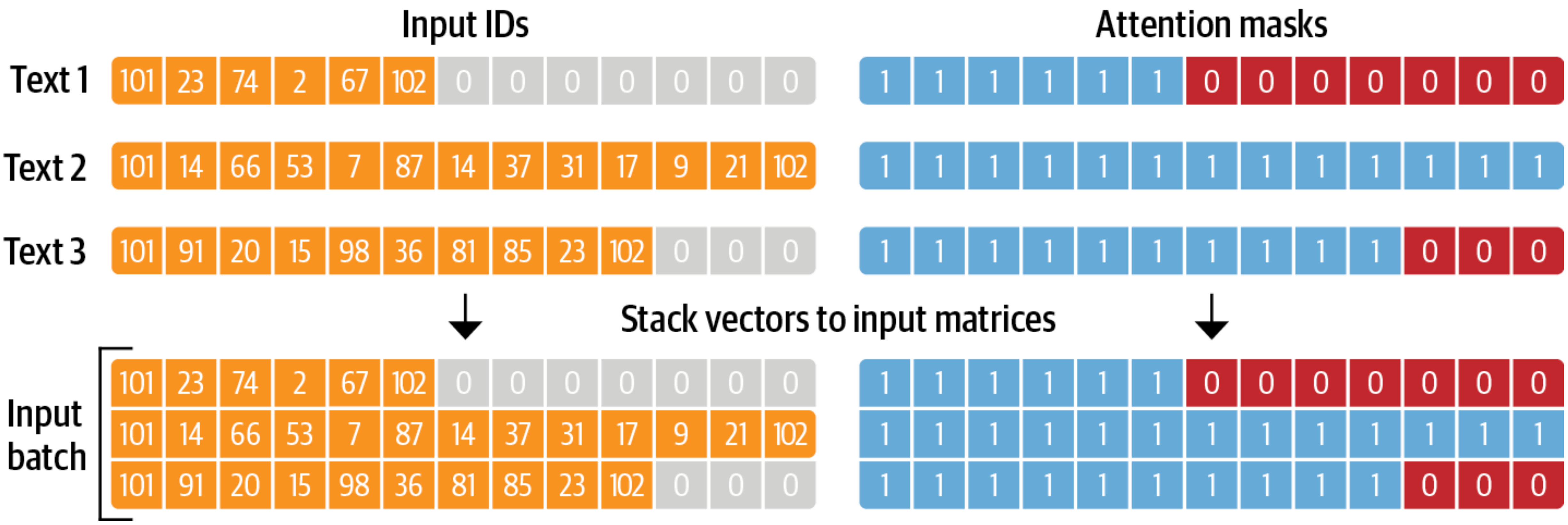
```
{'input_ids': [[101, 1045, 2134, 2102, 2514, 26608, 102, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [101, 1045, 2064, 2175, 2013, 3110, 2061,
20625, 2000, 2061, 9636, 17772, 2074, 2013, 2108, 2105, 2619, 2040, 14977,
1998, 2003, 8300, 102]], 'attention_mask': [[1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]}
```

```
tokens2ids = list(zip(tokenizer.all_special_tokens,
tokenizer.all_special_ids))
data = sorted(tokens2ids, key=lambda x : x[-1])
df = pd.DataFrame(data, columns=["Special Token", "Special Token ID"])
df.T
```

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

# From Text to Tokens

## For each batch, the input sequences are padded to the maximum sequence length in the batch; the attention mask is used in the model to ignore the padded areas of the input tensors



Input IDs

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Text 1 | 101 | 23 | 74 | 2 | 67 | 102 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Text 2 | 101 | 14 | 66 | 53 | 7 | 87 | 14 | 37 | 31 | 17 | 9 | 21 | 102 |
| Text 3 | 101 | 91 | 20 | 15 | 98 | 36 | 81 | 85 | 23 | 102 | 0 | 0 | 0 |

Attention masks

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

Stack vectors to input matrices

Input batch

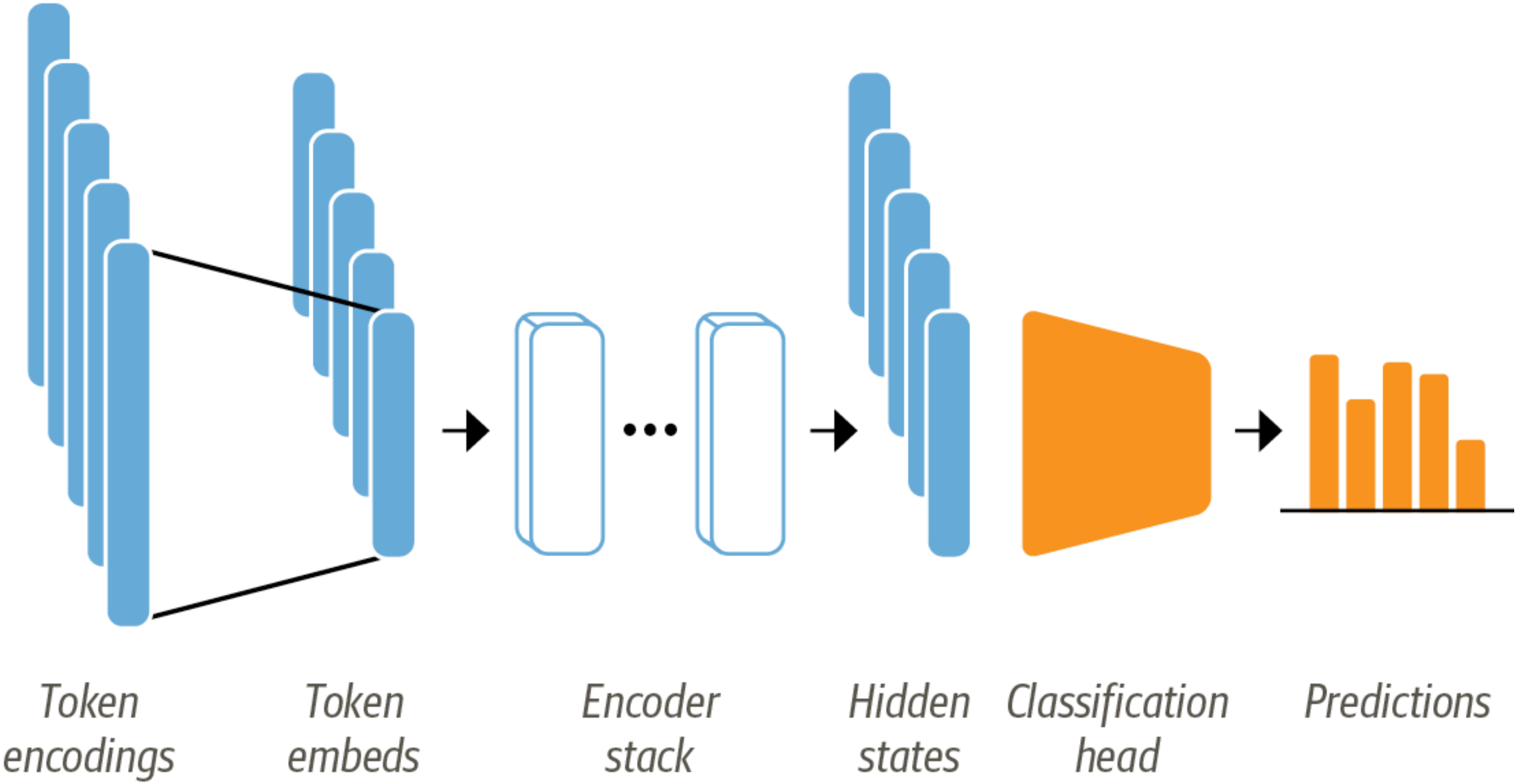
| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 101 | 23 | 74 | 2 | 67 | 102 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 101 | 14 | 66 | 53 | 7 | 87 | 14 | 37 | 31 | 17 | 9 | 21 | 102 |
| 101 | 91 | 20 | 15 | 98 | 36 | 81 | 85 | 23 | 102 | 0 | 0 | 0 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

# Training a Text Classifier

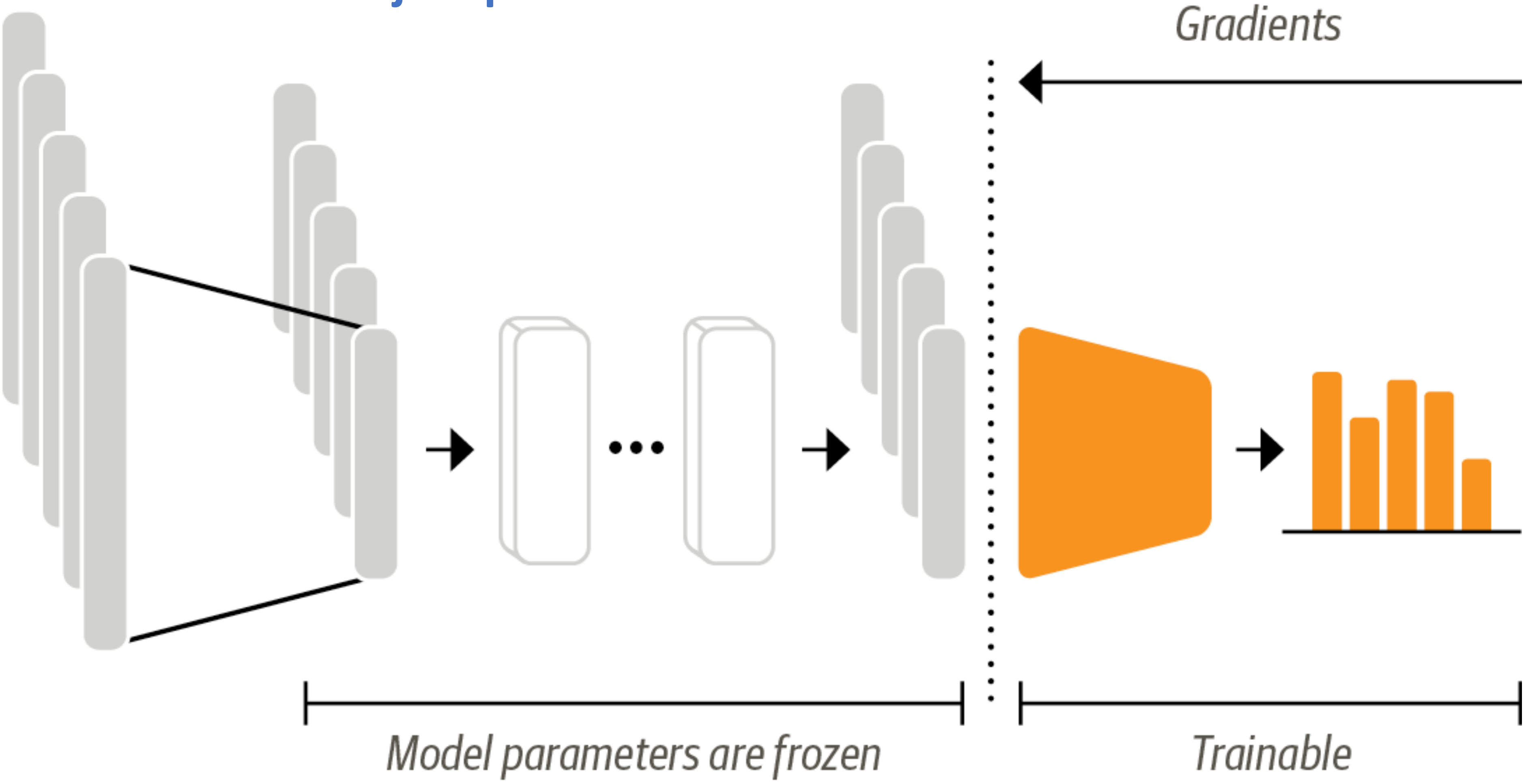
## The architecture used for sequence classification with an encoder-based transformer ; it consists of the model’s pretrained body combined with a custom classification head



Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

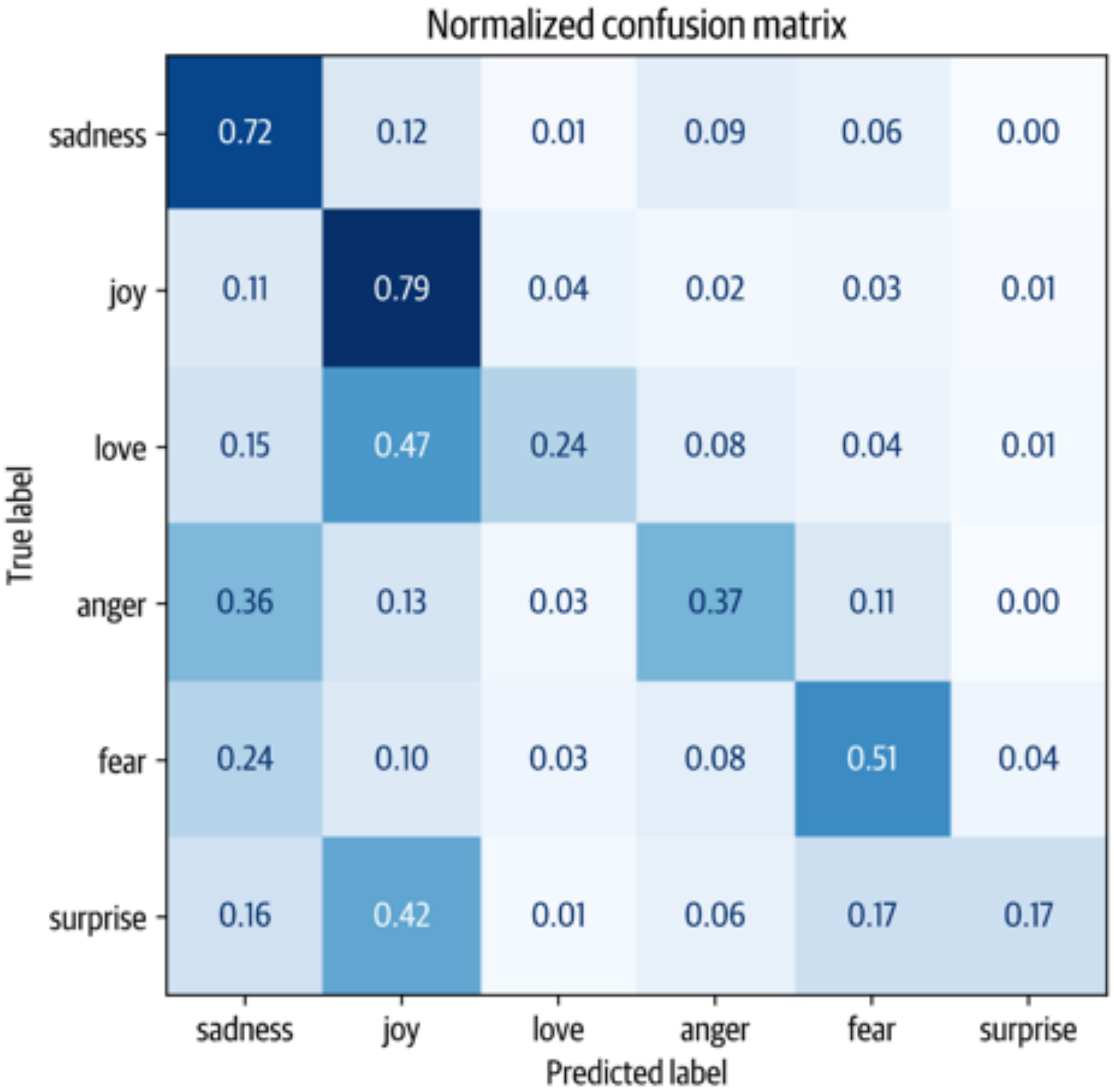
# Transformers as Feature Extractors

## In the feature-based approach, the DistilBERT model is frozen and just provides features for a classifier



Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

# Training a Simple Classifier

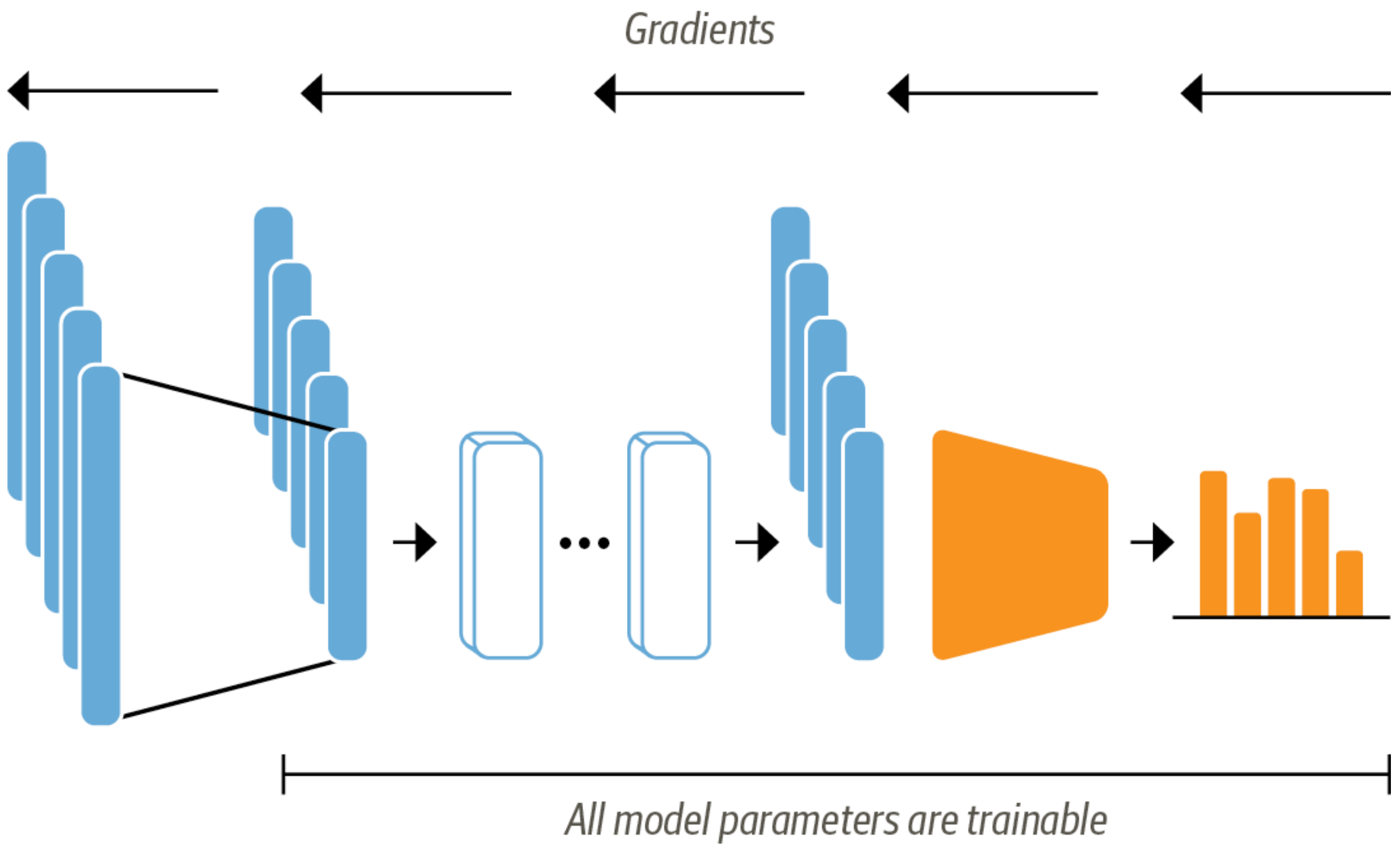


| | sadness | joy | love | anger | fear | surprise |
|---|---|---|---|---|---|---|
| sadness | 0.72 | 0.12 | 0.01 | 0.09 | 0.06 | 0.00 |
| joy | 0.11 | 0.79 | 0.04 | 0.02 | 0.03 | 0.01 |
| love | 0.15 | 0.47 | 0.24 | 0.08 | 0.04 | 0.01 |
| anger | 0.36 | 0.13 | 0.03 | 0.37 | 0.11 | 0.00 |
| fear | 0.24 | 0.10 | 0.03 | 0.08 | 0.51 | 0.04 |
| surprise | 0.16 | 0.42 | 0.01 | 0.06 | 0.17 | 0.17 |

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

# Fine-Tuning Transformers

## When using the fine-tuning approach the whole DistilBERT model is trained along with the classification head



Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

# Fine-Tuning Transformers
# Loading a pretrained model

```python
from transformers import AutoModelForSequenceClassification

num_labels = 6
model = (AutoModelForSequenceClassification
         .from_pretrained(model_ckpt, num_labels=num_labels)
         .to(device))
```

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

# Defining the performance metrics

```
from sklearn.metrics import accuracy_score, f1_score

def compute_metrics(pred):
     labels = pred.label_ids
     preds = pred.predictions.argmax(-1)
     f1 = f1_score(labels, preds, average="weighted")
     acc = accuracy_score(labels, preds)
     return {"accuracy": acc, "f1": f1}
```

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

# Train the model

```
from huggingface_hub import notebook_login

notebook_login()
```

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

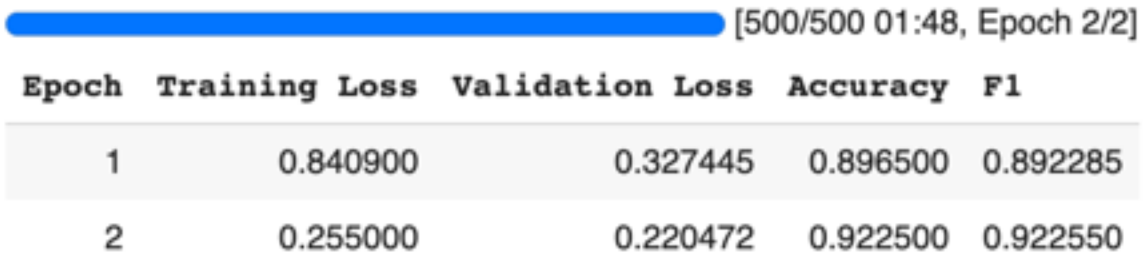
# Train the model

```
from transformers import Trainer, TrainingArguments

batch_size = 64
logging_steps = len(emotions_encoded["train"]) // batch_size
model_name = f"{model_ckpt}-finetuned-emotion"
training_args = TrainingArguments(output_dir=model_name,
                    num_train_epochs=2,
                    learning_rate=2e-5,
                    per_device_train_batch_size=batch_size,
                    per_device_eval_batch_size=batch_size,
                    weight_decay=0.01,
                    evaluation_strategy="epoch",
                    disable_tqdm=False,
                    logging_steps=logging_steps,
                    push_to_hub=True,
                    log_level="error")
```

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

# Train the model

```python
from transformers import Trainer

trainer = Trainer(model=model, args=training_args,
          compute_metrics=compute_metrics,
          train_dataset=emotions_encoded["train"],
          eval_dataset=emotions_encoded["validation"],
          tokenizer=tokenizer)
trainer.train();
```



[500/500 01:48, Epoch 2/2]

| Epoch | Training Loss | Validation Loss | Accuracy | F1 |
|---|---|---|---|---|
| 1 | 0.840900 | 0.327445 | 0.896500 | 0.892285 |
| 2 | 0.255000 | 0.220472 | 0.922500 | 0.922550 |

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

# Train the model

```
preds_output =
trainer.predict(emotions_encoded["validation"])
```

```
preds_output.metrics
```

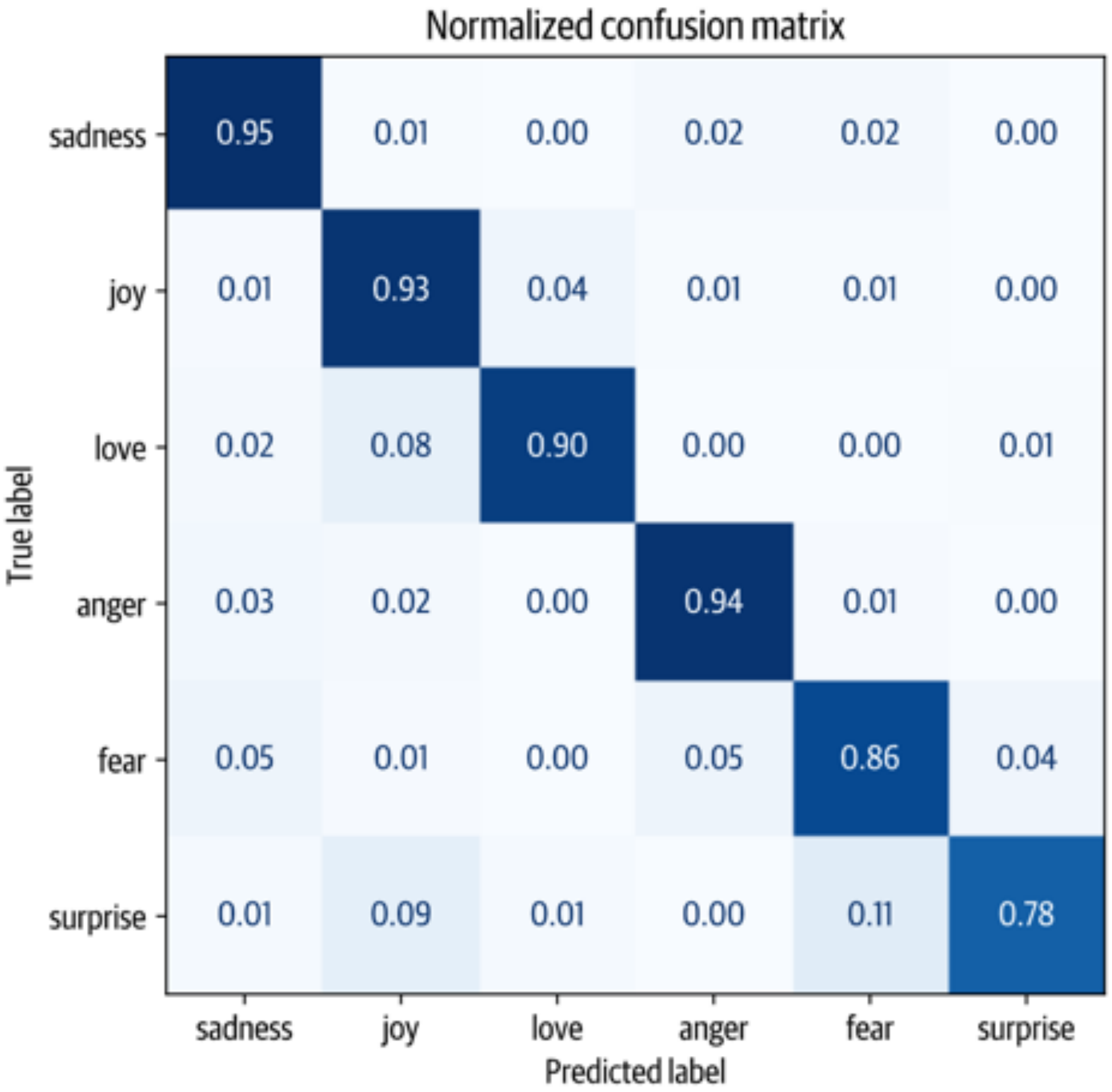
```
{'test_loss': 0.22047173976898193, 'test_accuracy': 0.9225, 'test_f1':
0.9225500751072866, 'test_runtime': 1.6357, 'test_samples_per_second':
1222.725, 'test_steps_per_second': 19.564}
```

```
y_preds = np.argmax(preds_output.predictions, axis=1)
```

```
plot_confusion_matrix(y_preds, y_valid, labels)
```

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

# Fine-Tuning Transformers

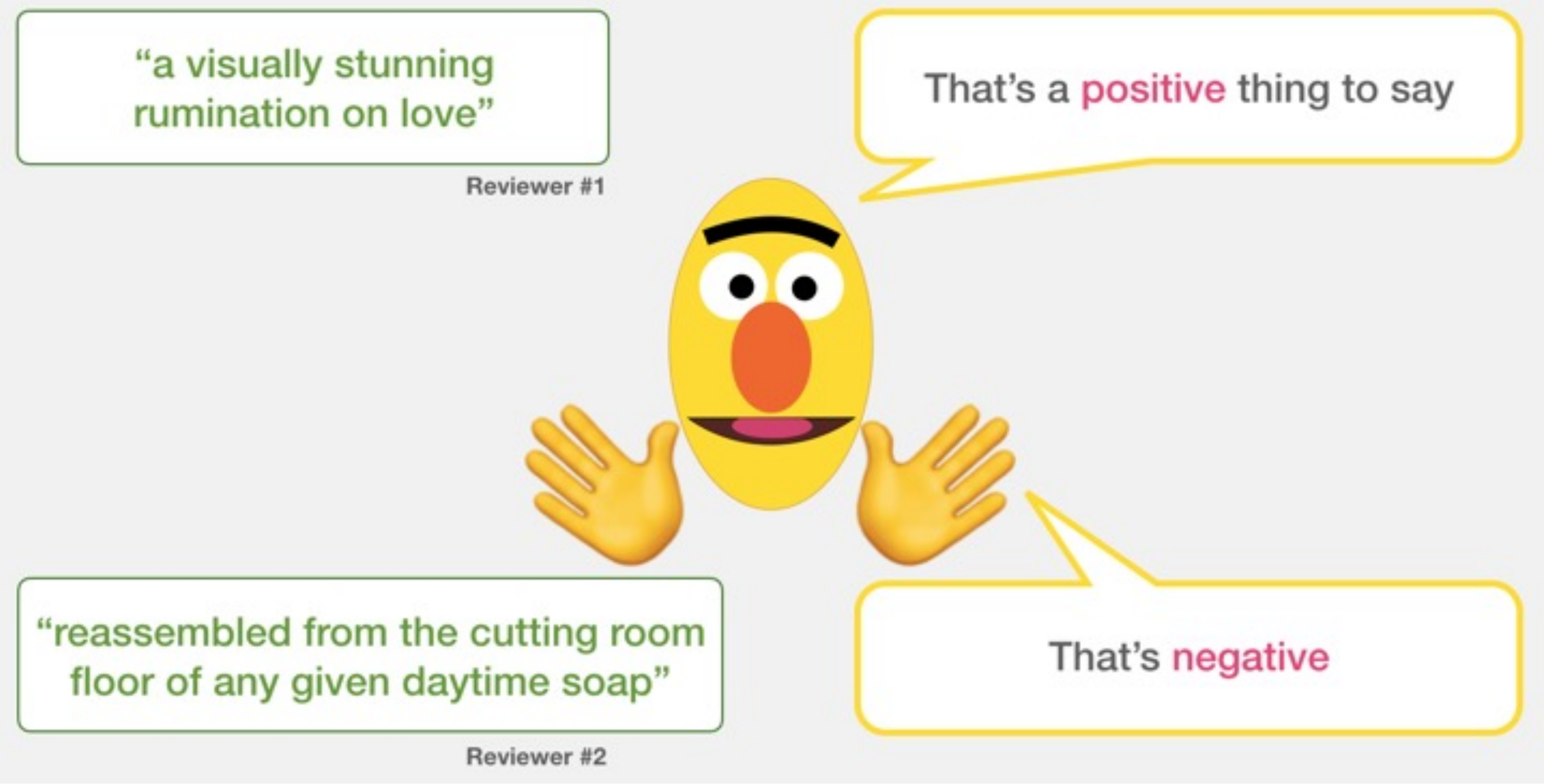


| | sadness | joy | love | anger | fear | surprise |
|---|---|---|---|---|---|---|
| sadness | 0.95 | 0.01 | 0.00 | 0.02 | 0.02 | 0.00 |
| joy | 0.01 | 0.93 | 0.04 | 0.01 | 0.01 | 0.00 |
| love | 0.02 | 0.08 | 0.90 | 0.00 | 0.00 | 0.01 |
| anger | 0.03 | 0.02 | 0.00 | 0.94 | 0.01 | 0.00 |
| fear | 0.05 | 0.01 | 0.00 | 0.05 | 0.86 | 0.04 |
| surprise | 0.01 | 0.09 | 0.01 | 0.00 | 0.11 | 0.78 |

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

# 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

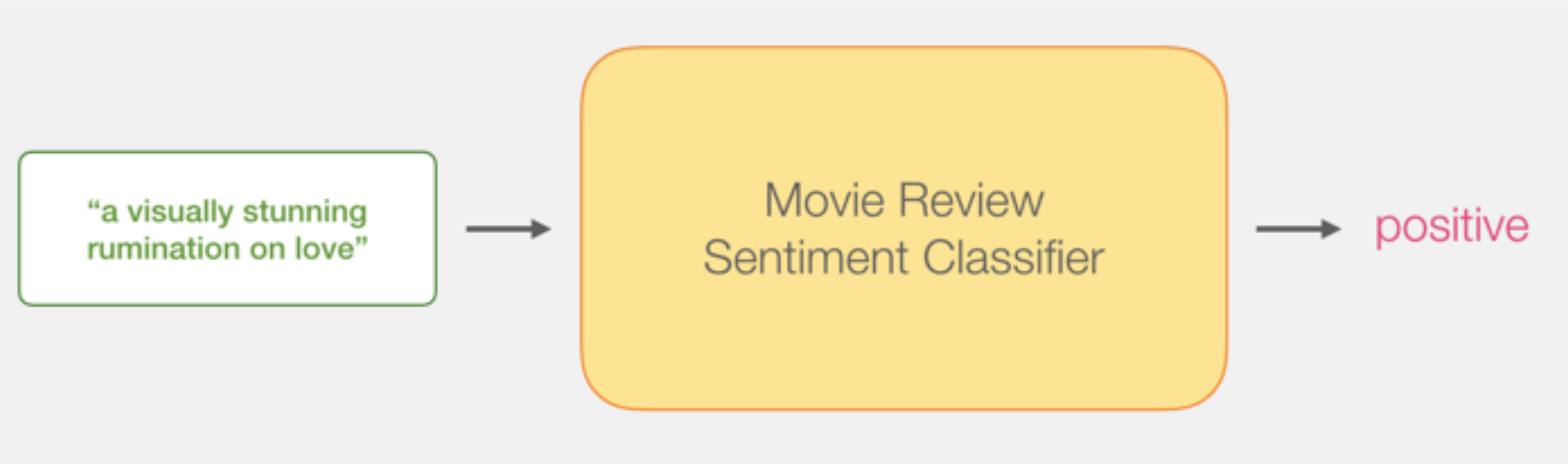
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

# Sentiment Classification: SST2
# Sentences from movie reviews

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

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

# Movie Review Sentiment Classifier



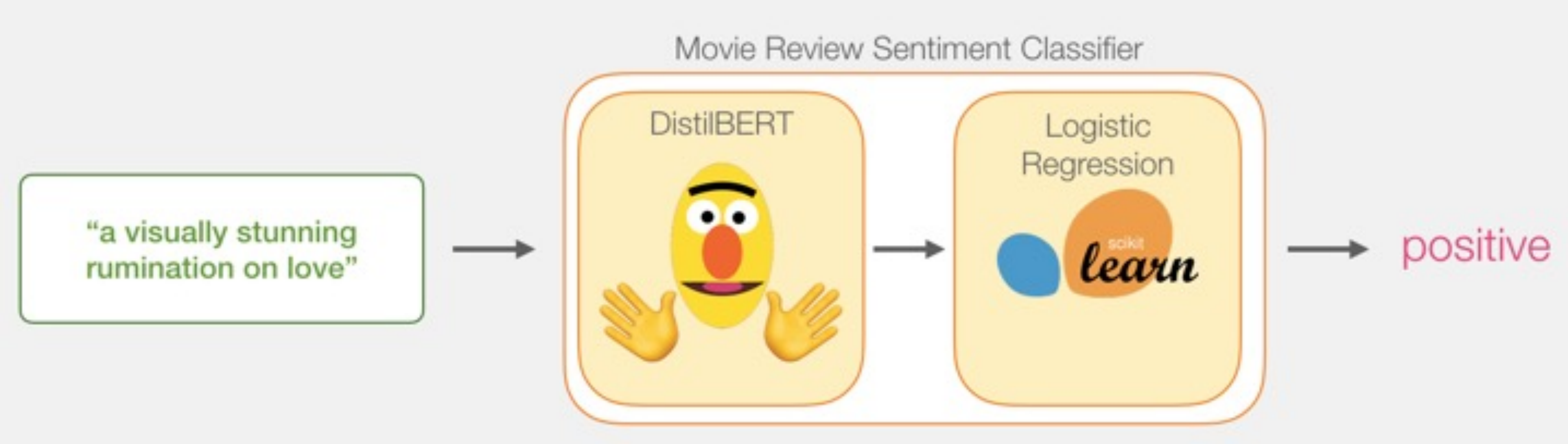
“a visually stunning rumination on love”

Movie Review Sentiment Classifier

positive

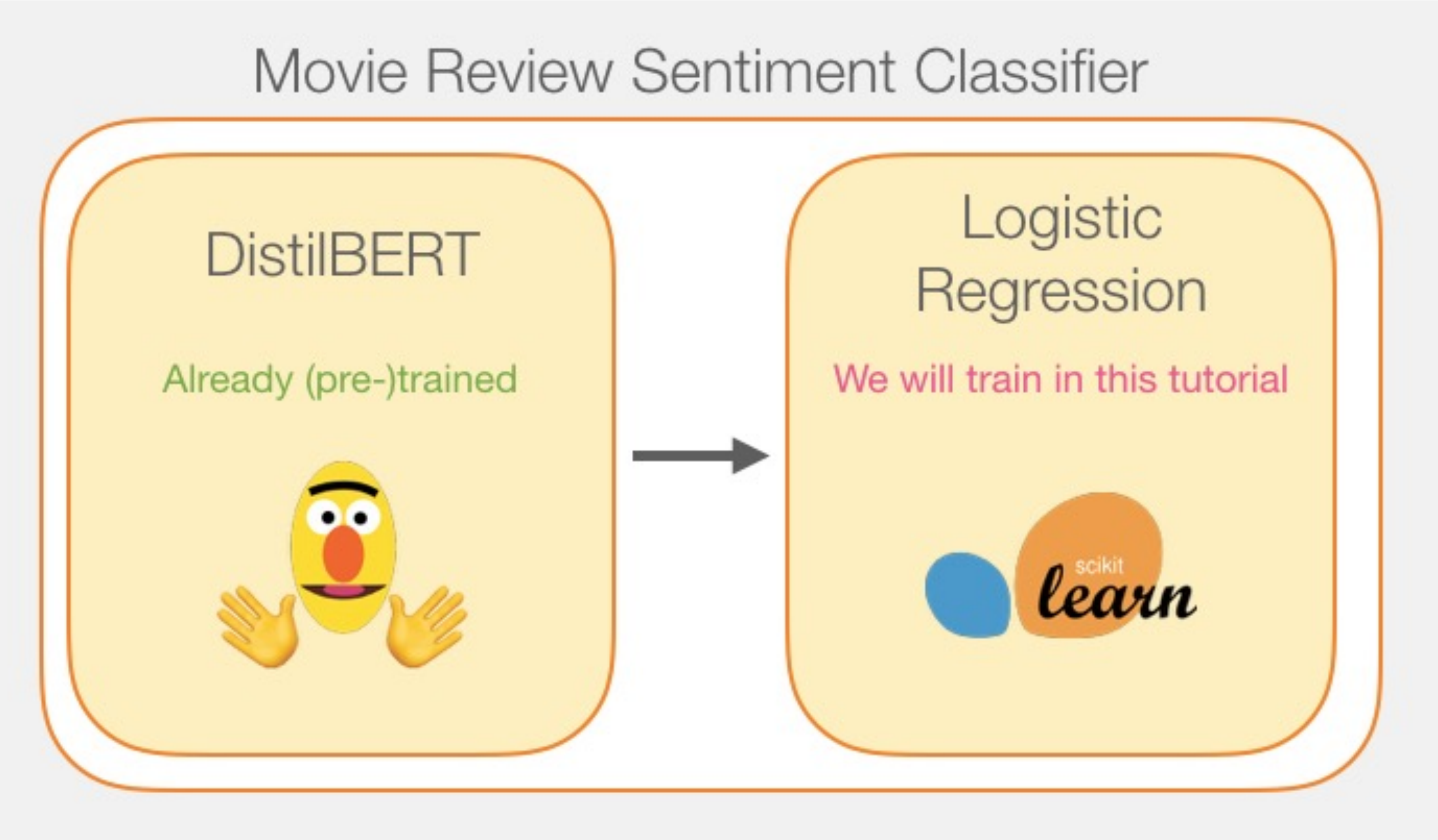
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

# Movie Review Sentiment Classifier


Movie Review Sentiment Classifier
"a visually stunning rumination on love"
DistilBERT
Logistic Regression
scikit learn
positive

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

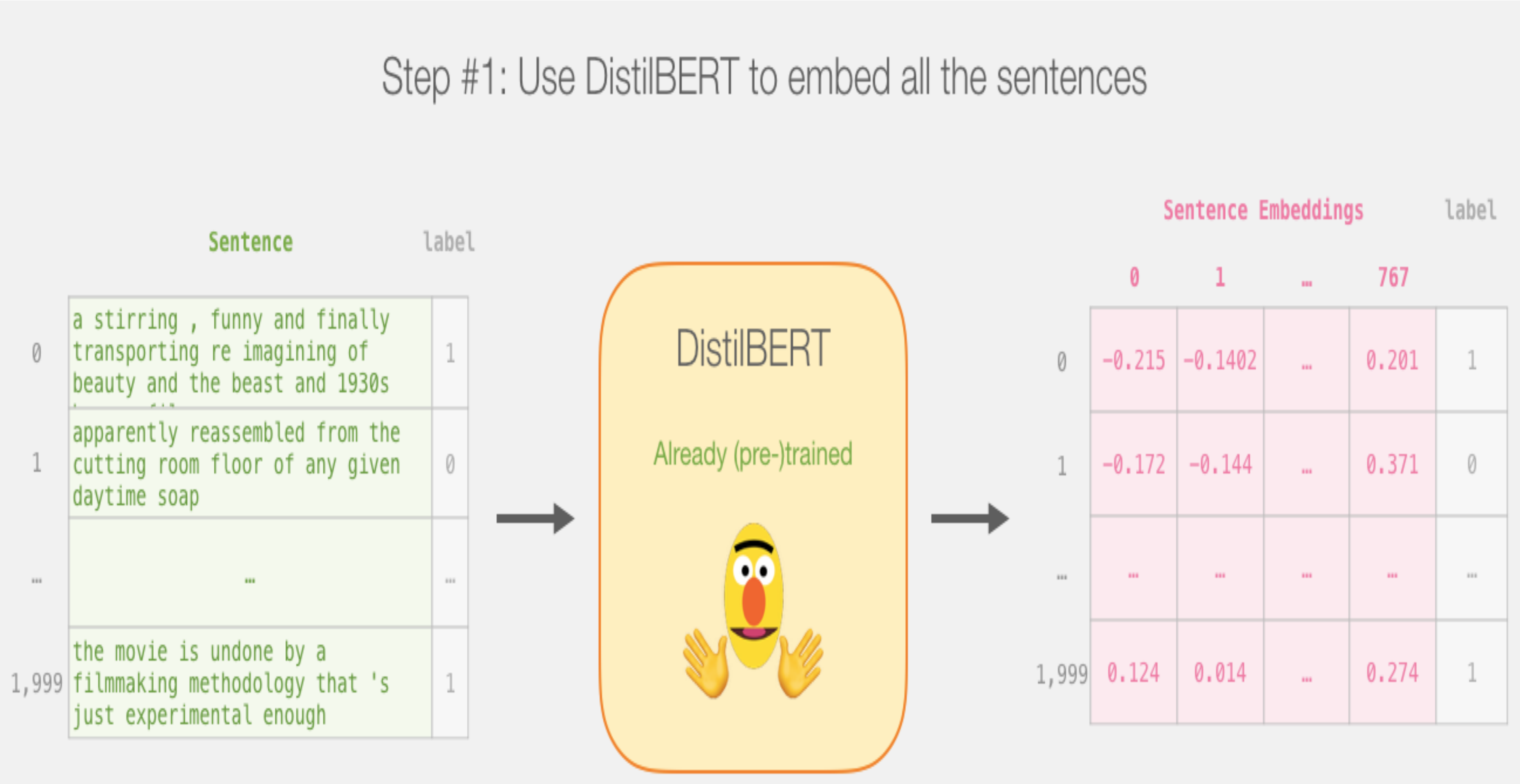
# Movie Review Sentiment Classifier Model Training



Movie Review Sentiment Classifier
DistilBERT
Already (pre-)trained
Logistic Regression
We will train in this tutorial
scikit learn

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

# Step # 1 Use distilBERT to Generate Sentence Embeddings



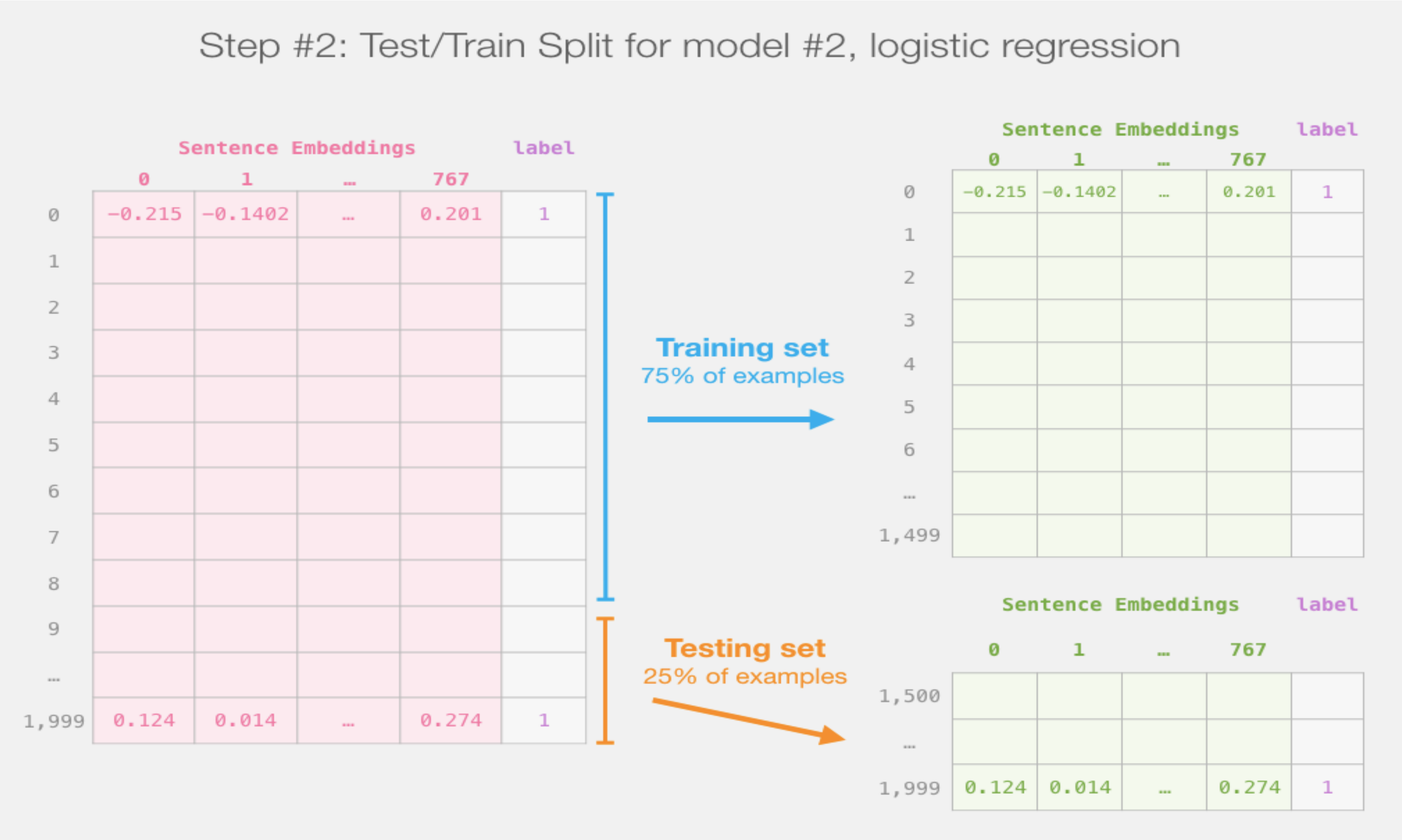
Step #1: Use DistilBERT to embed all the sentences

| | Sentence | label |
|---|---|---|
| 0 | a stirring , funny and finally transporting re imagining of beauty and the beast and 1930s | 1 |
| 1 | apparently reassembled from the cutting room floor of any given daytime soap | 0 |
| … | … | … |
| 1,999 | the movie is undone by a filmmaking methodology that 's just experimental enough | 1 |

| | Sentence Embeddings | | | | label |
|---|---|---|---|---|---|
| | 0 | 1 | … | 767 | |
| 0 | -0.215 | -0.1402 | … | 0.201 | 1 |
| 1 | -0.172 | -0.144 | … | 0.371 | 0 |
| … | … | … | … | … | … |
| 1,999 | 0.124 | 0.014 | … | 0.274 | 1 |

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

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



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

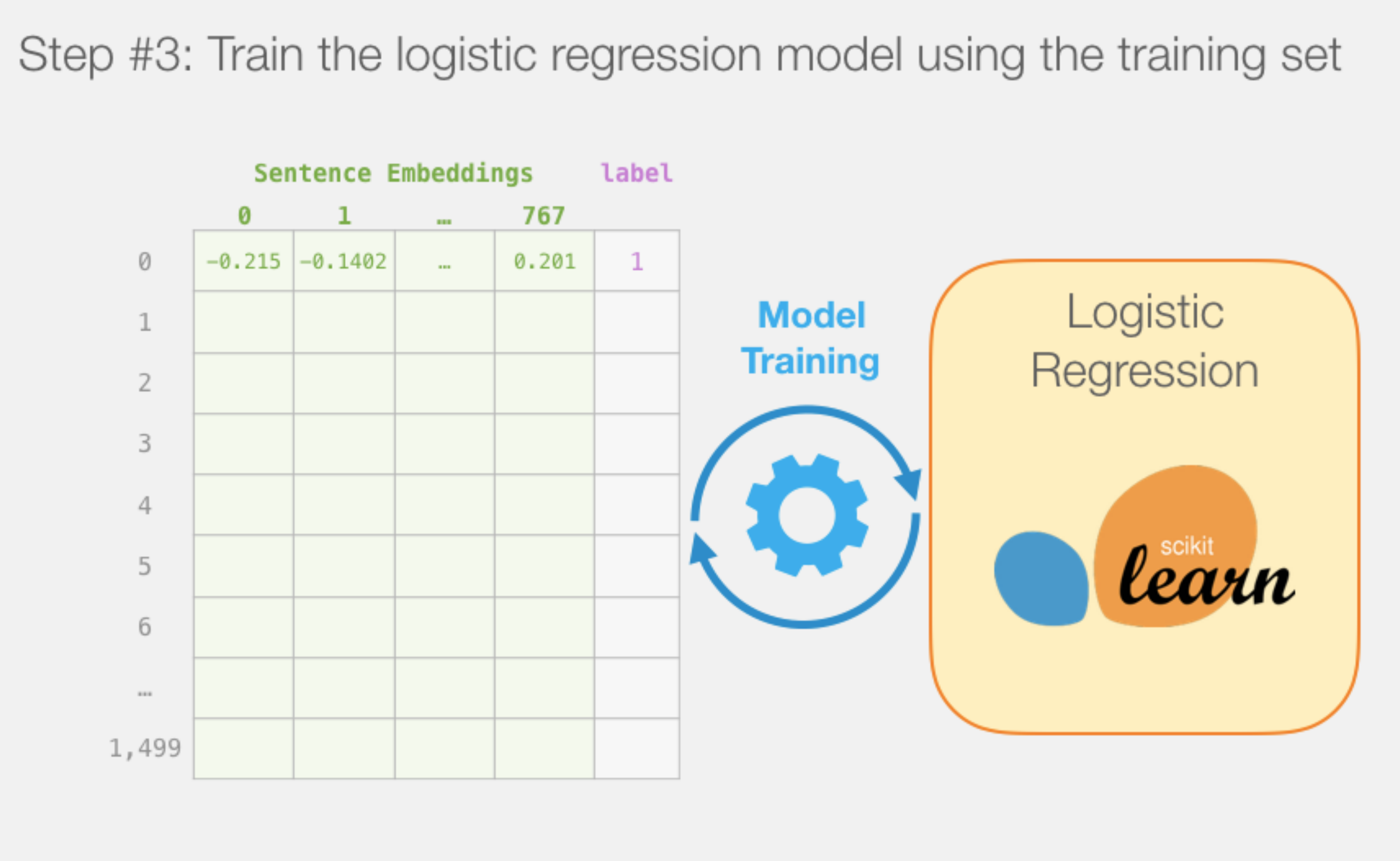
| | Sentence Embeddings | | | | label |
|---|---|---|---|---|---|
| | 0 | 1 | … | 767 | |
| 0 | -0.215 | -0.1402 | … | 0.201 | 1 |
| 1 | | | | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | | | | | |
| 5 | | | | | |
| 6 | | | | | |
| 7 | | | | | |
| 8 | | | | | |
| 9 | | | | | |
| … | | | | | |
| 1,999 | 0.124 | 0.014 | … | 0.274 | 1 |

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

| | Sentence Embeddings | | | | label |
|---|---|---|---|---|---|
| | 0 | 1 | … | 767 | |
| 1,500 | | | | | |
| … | | | | | |
| 1,999 | 0.124 | 0.014 | … | 0.274 | 1 |

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

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



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

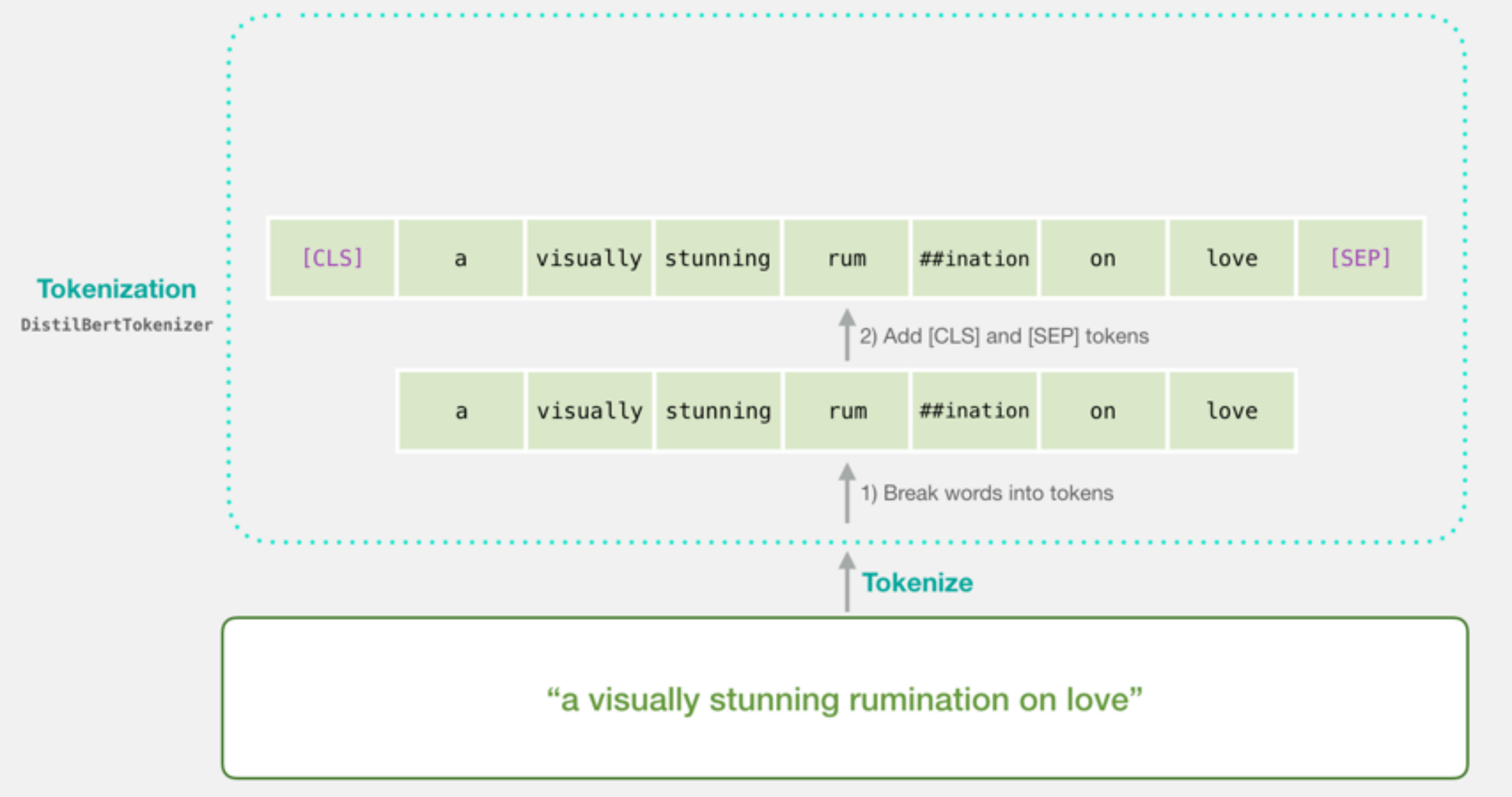
|  | Sentence Embeddings |  |  |  | label |
|---|---|---|---|---|---|
|  | 0 | 1 | … | 767 |  |
| 0 | -0.215 | -0.1402 | … | 0.201 | 1 |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 4 |  |  |  |  |  |
| 5 |  |  |  |  |  |
| 6 |  |  |  |  |  |
| … |  |  |  |  |  |
| 1,499 |  |  | 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

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

# Tokenization

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

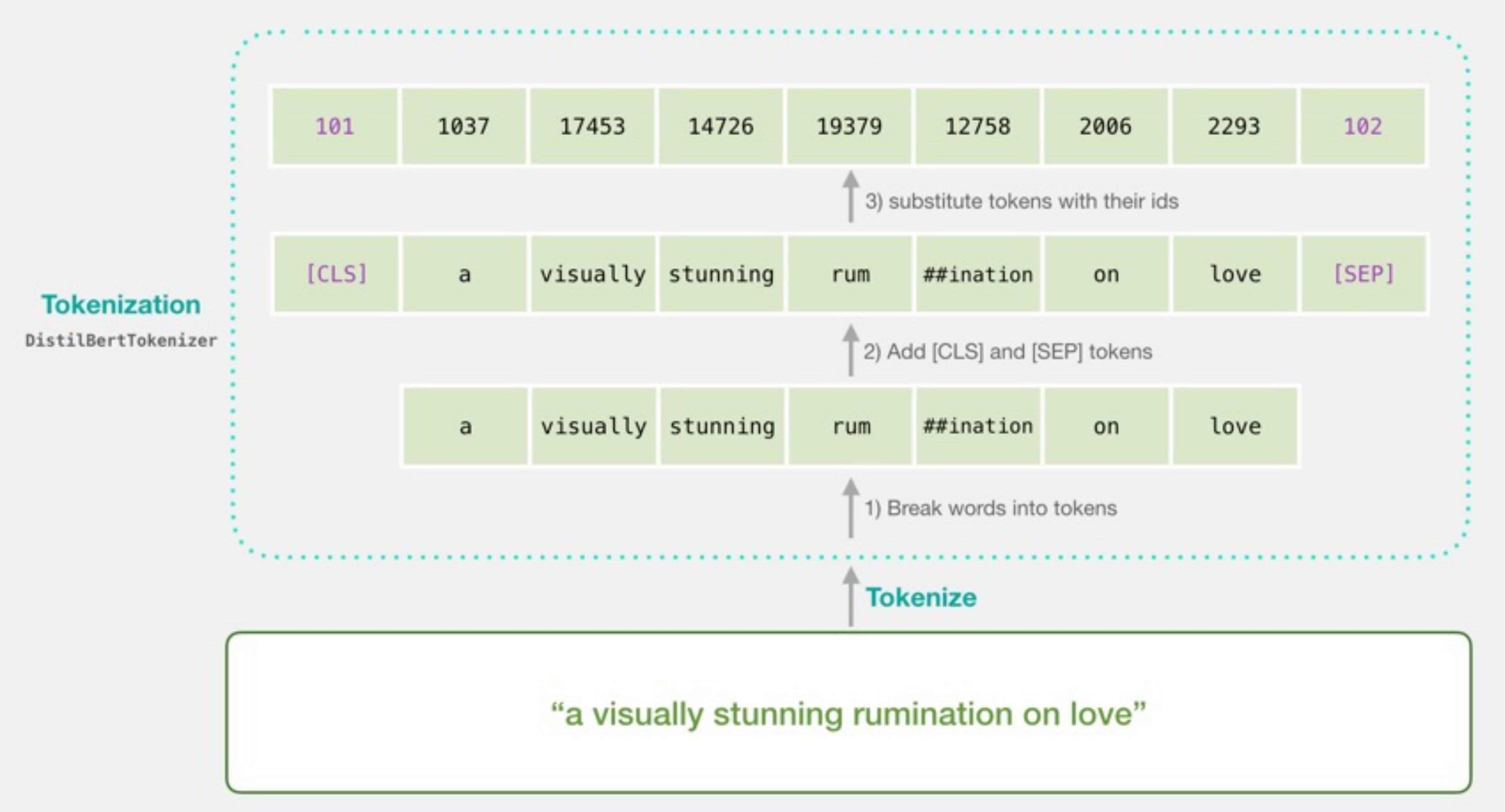
a visually stunning rumination on love



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

# Tokenization

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



Tokenization

DistilBertTokenizer

| 101 | 1037 | 17453 | 14726 | 19379 | 12758 | 2006 | 2293 | 102 |
|---|---|---|---|---|---|---|---|---|

3) substitute tokens with their ids

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

2) Add [CLS] and [SEP] tokens

| a | visually | stunning | rum | ##ination | on | love |
|---|---|---|---|---|---|---|

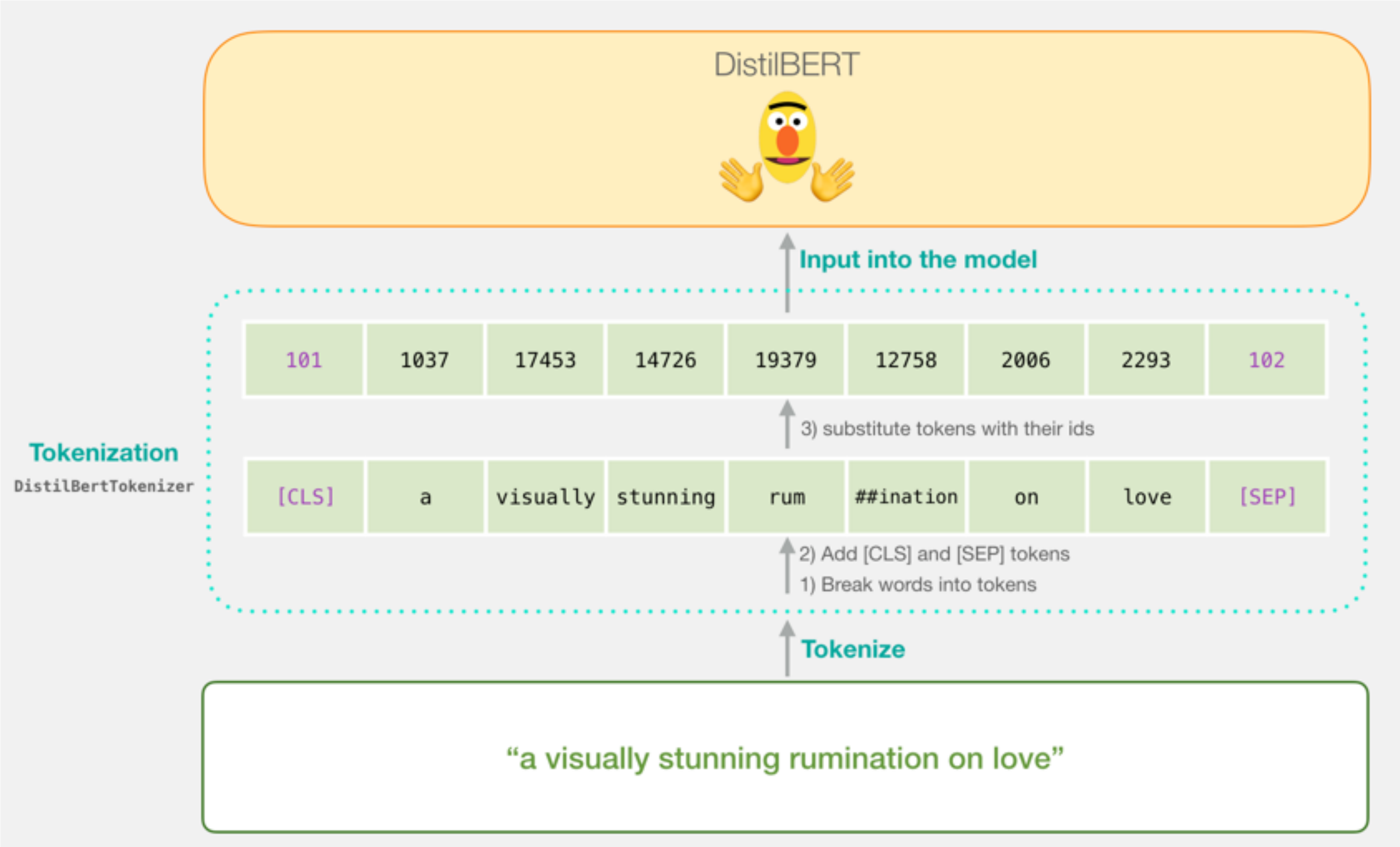
1) Break words into tokens

Tokenize

"a visually stunning rumination on love"

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

# Tokenization for BERT Model



DistilBERT

Input into the model

|  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|
| 101 | 1037 | 17453 | 14726 | 19379 | 12758 | 2006 | 2293 | 102 |

3) substitute tokens with their ids

Tokenization

DistilBertTokenizer

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

2) Add [CLS] and [SEP] tokens

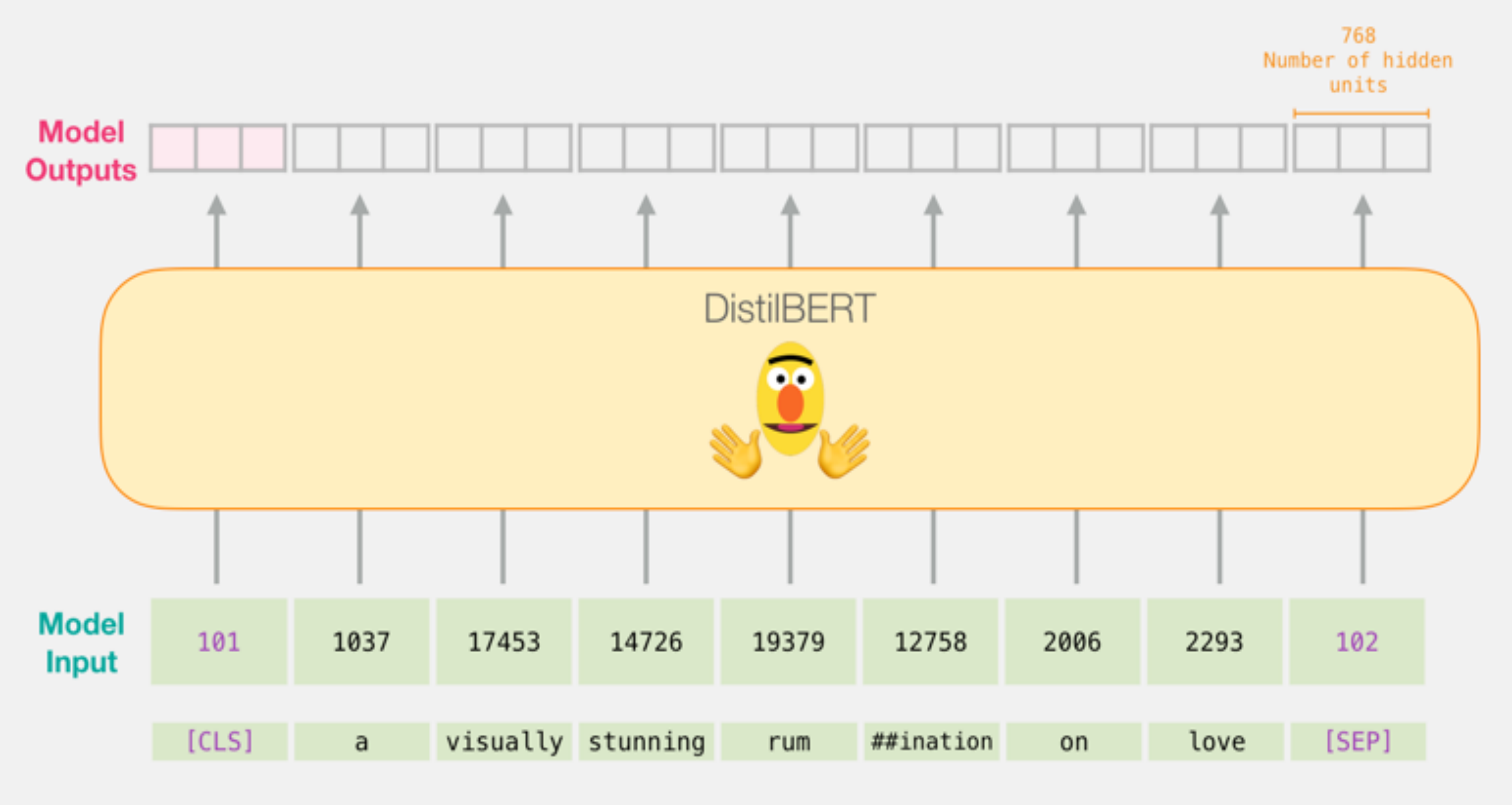
1) Break words into tokens

Tokenize

“a visually stunning rumination on love”

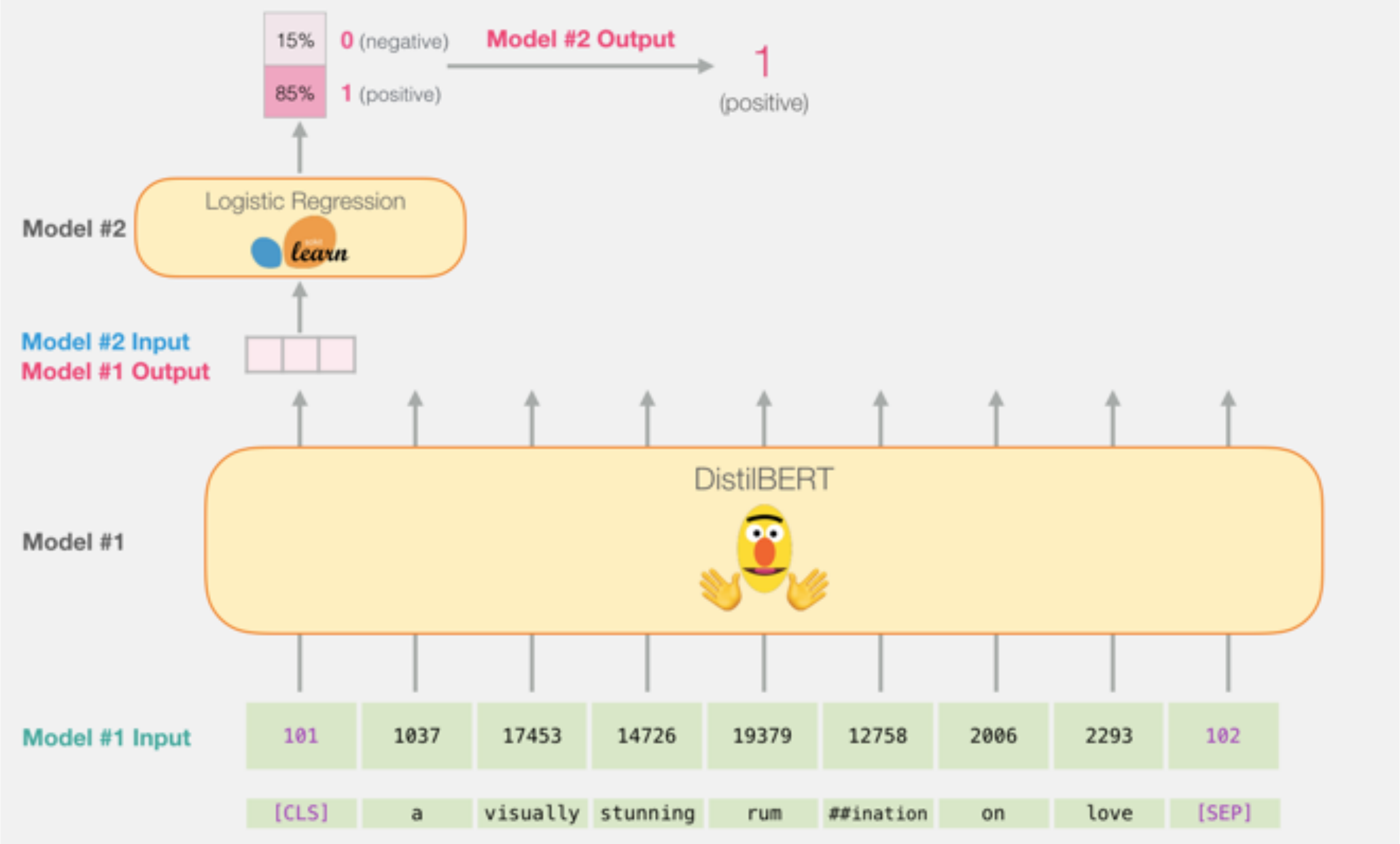
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

# Flowing Through DistilBERT (768 features)



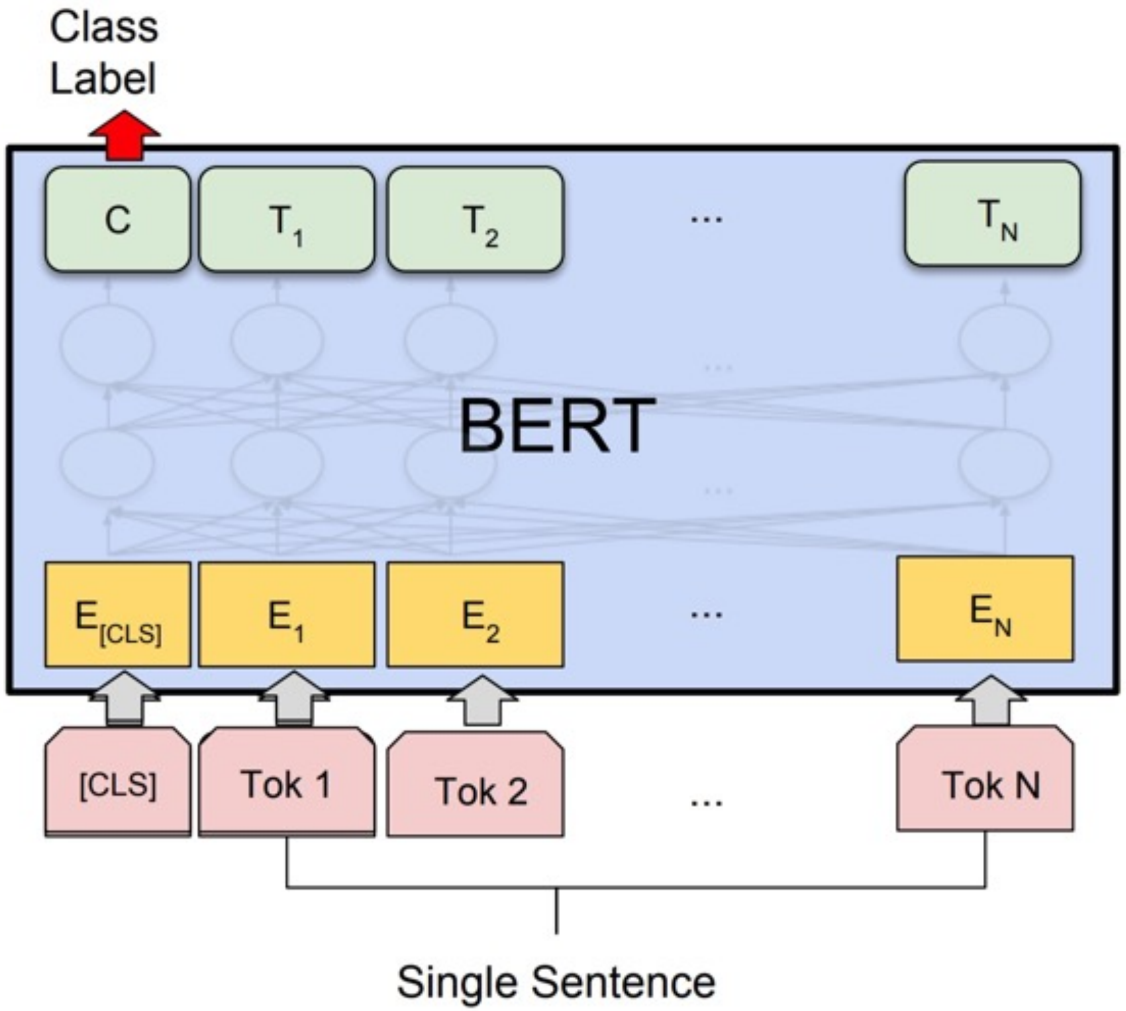
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

# Model #1 Output Class vector as Model #2 Input



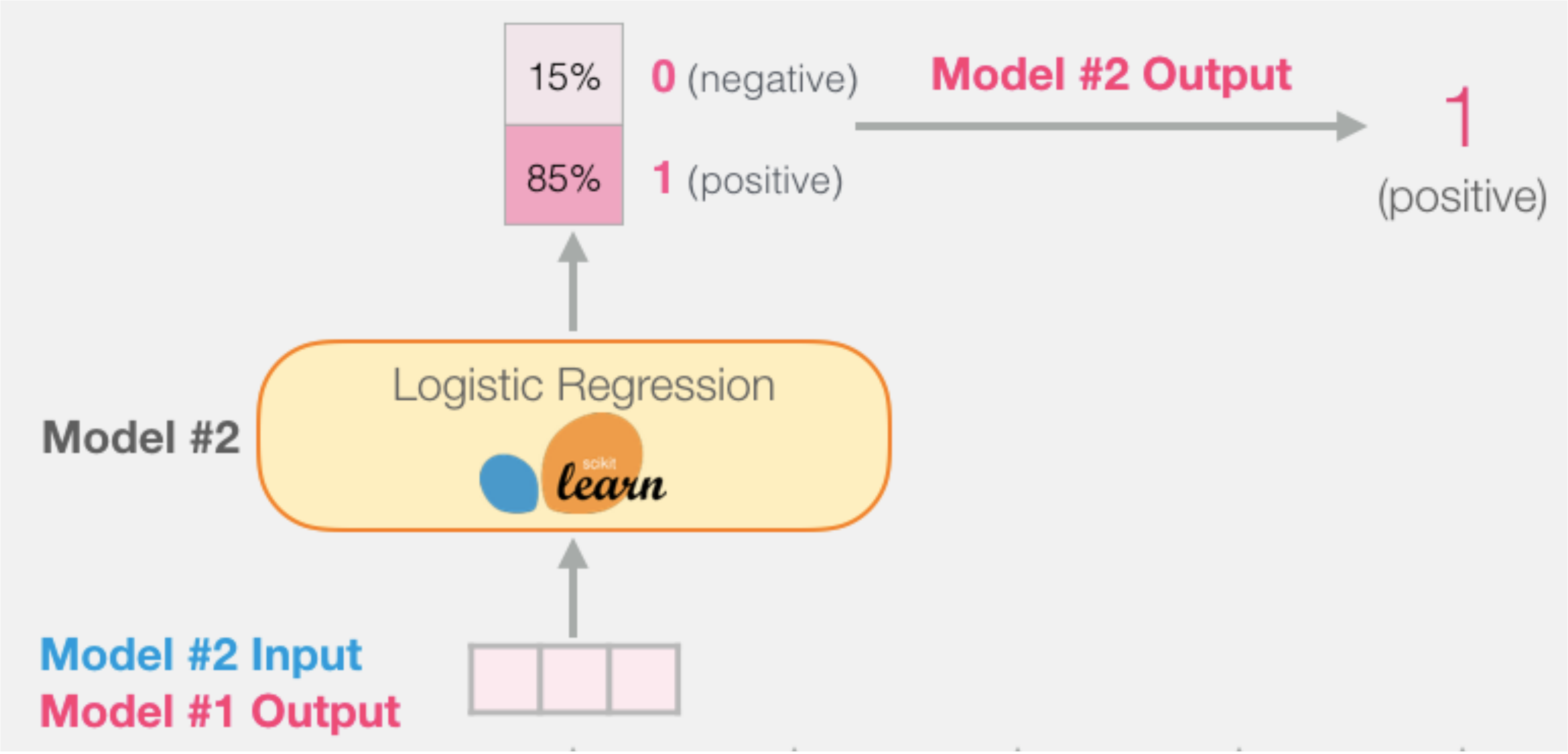
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

# Fine-tuning BERT on Single Sentence Classification Tasks



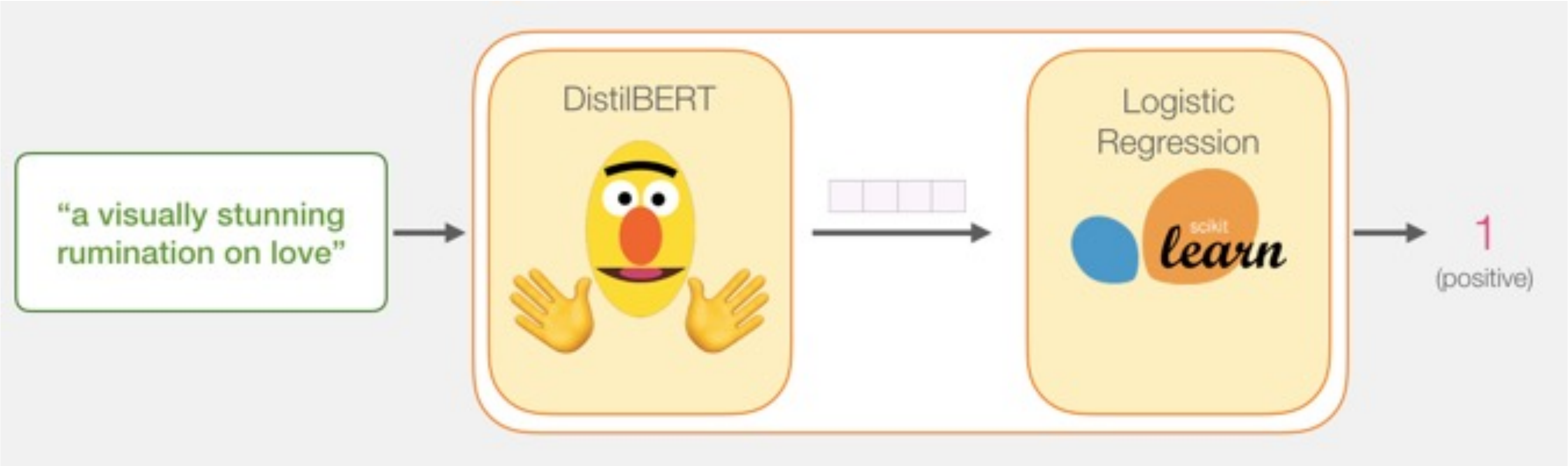
Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

# Model #1 Output Class vector as Model #2 Input



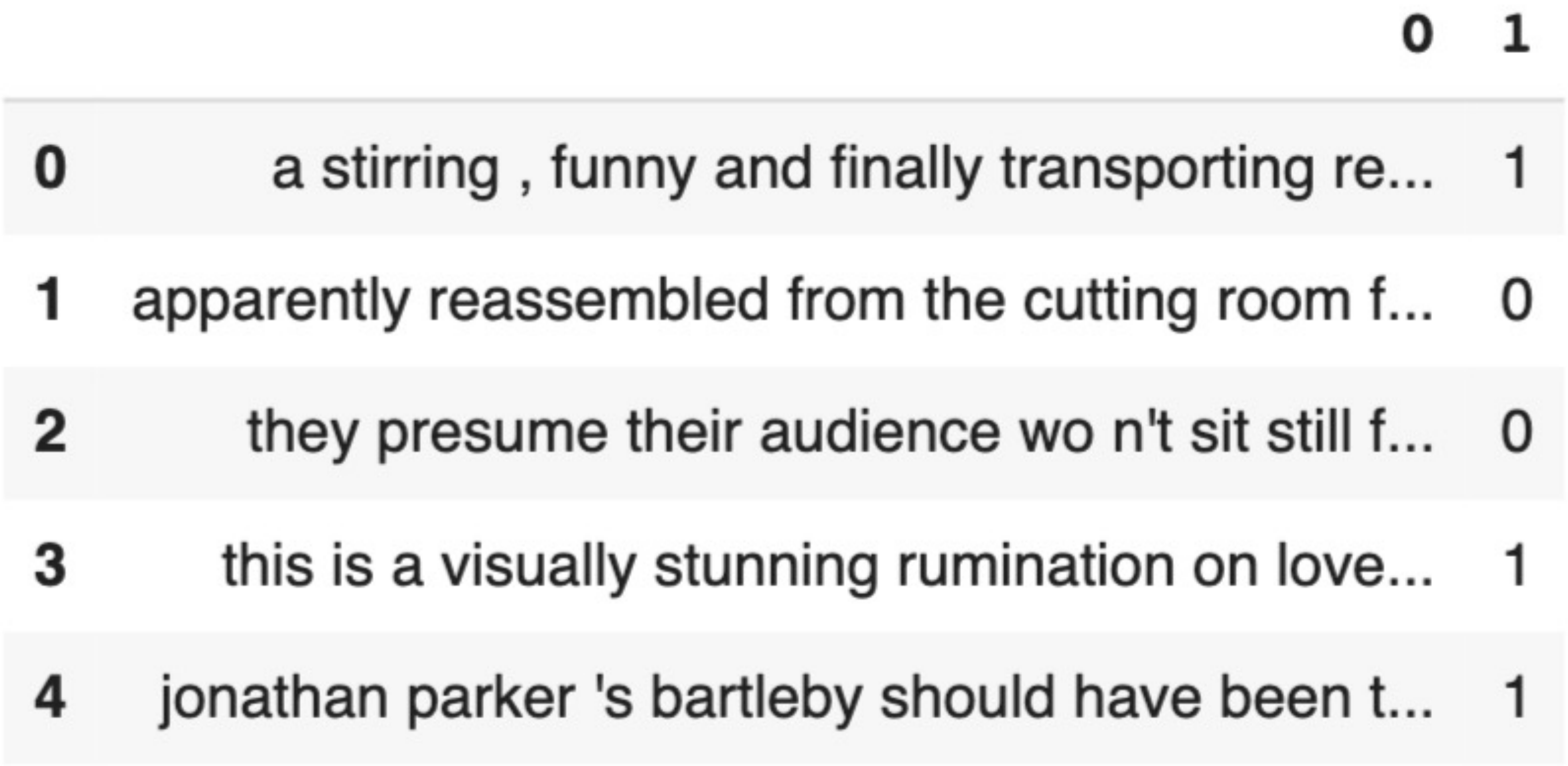
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

# Logistic Regression Model to classify Class vector



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

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

df.head()
```

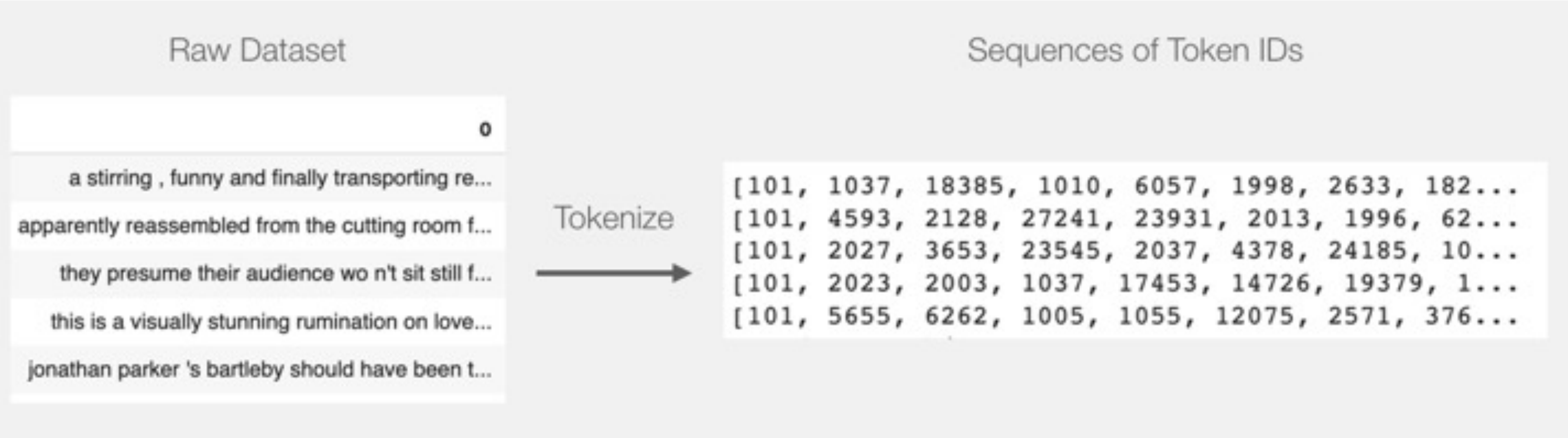


|   | 0 | 1 |
|---|---|---|
| **0** | a stirring , funny and finally transporting re... | 1 |
| **1** | apparently reassembled from the cutting room f... | 0 |
| **2** | they presume their audience wo n't sit still f... | 0 |
| **3** | this is a visually stunning rumination on love... | 1 |
| **4** |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

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

# Tokenization

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



Raw Dataset

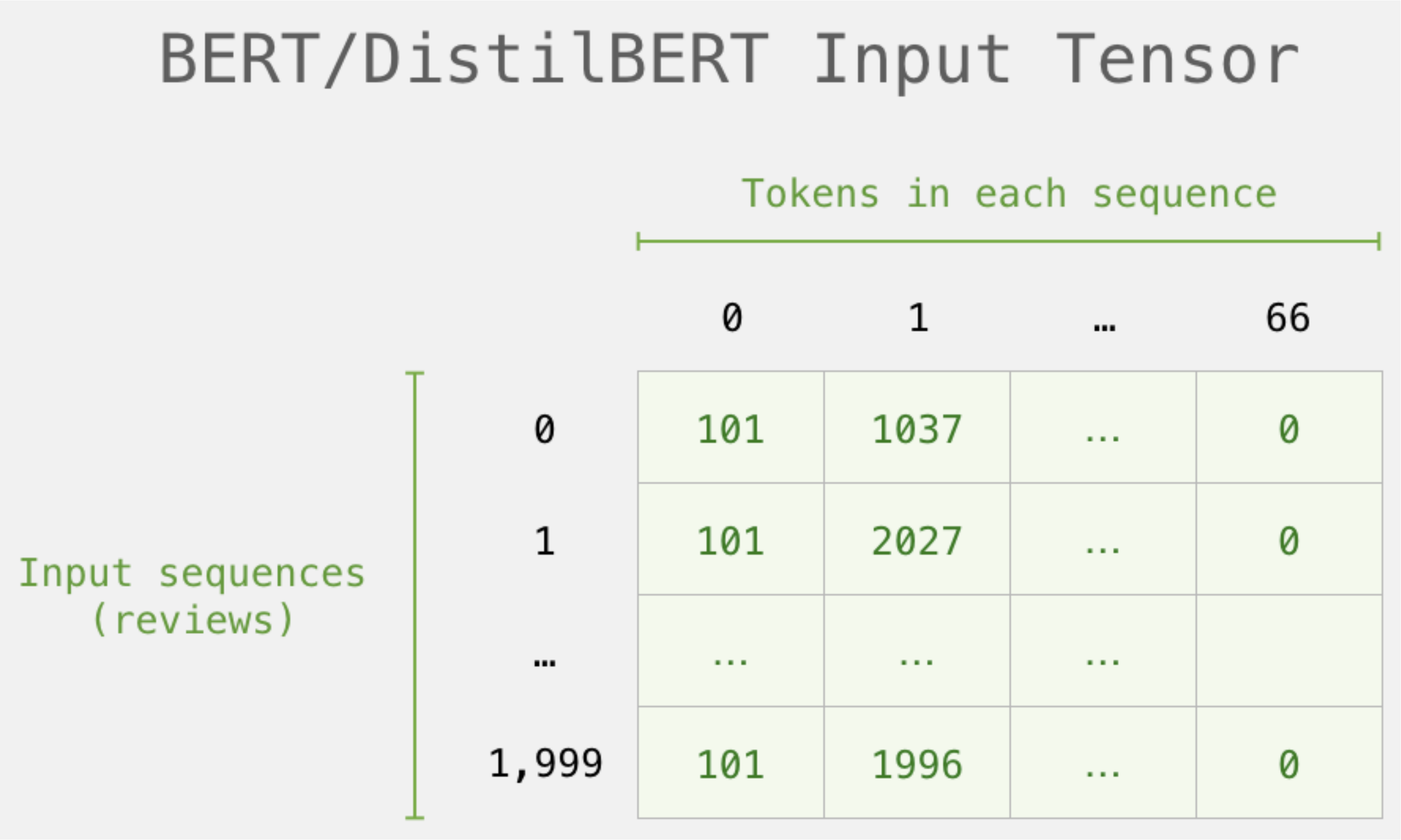
| 0 |
|---|
| a stirring , funny and finally transporting re... |
| apparently reassembled from the cutting room f... |
| they presume their audience wo n't sit still f... |
| this is a visually stunning rumination on love... |
| jonathan parker 's bartleby should have been t... |

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...
```

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

# BERT Input Tensor



BERT/DistilBERT Input Tensor

Tokens in each sequence

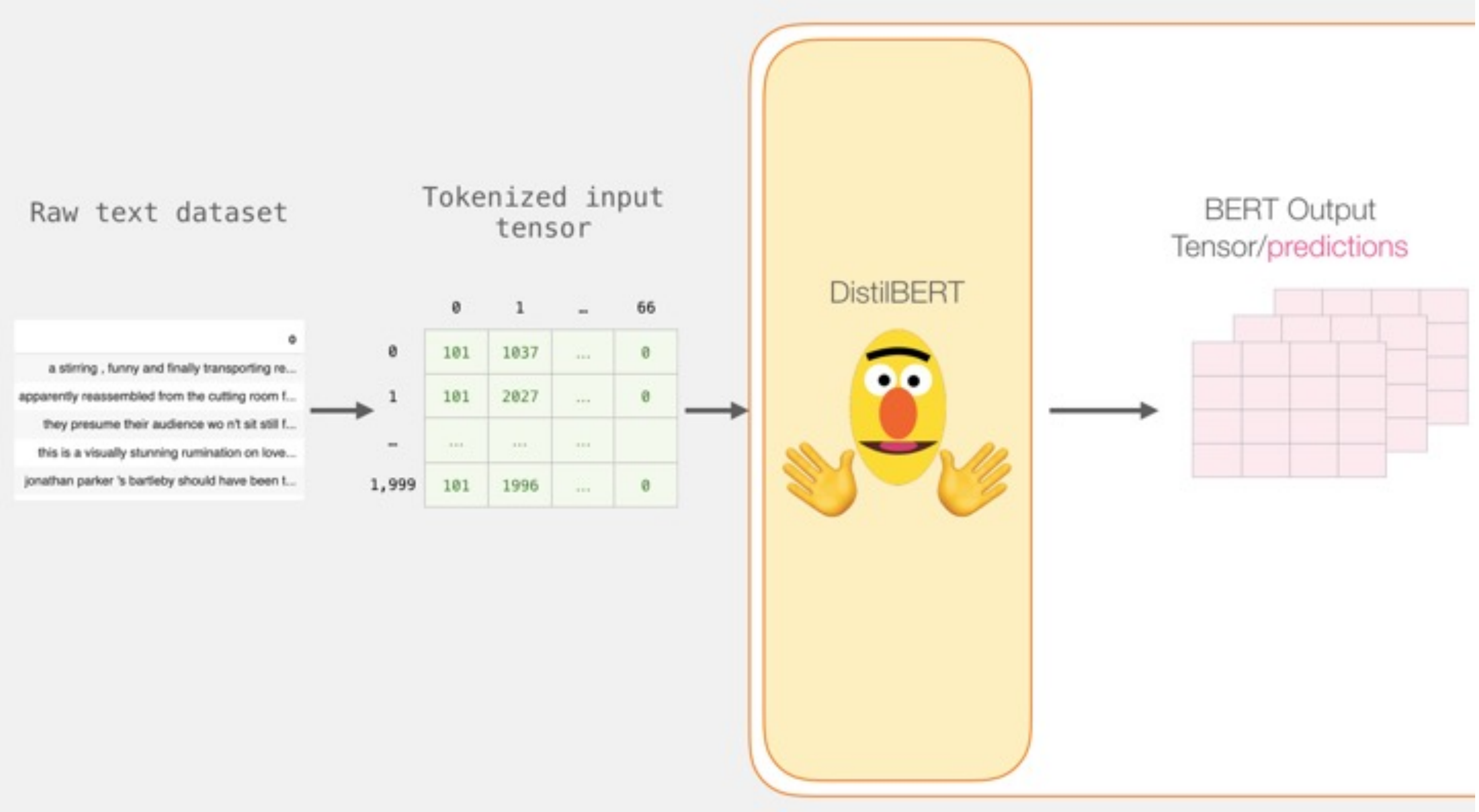
Input sequences (reviews)

|  | 0 | 1 | … | 66 |
|---|---|---|---|---|
| 0 | 101 | 1037 | … | 0 |
| 1 | 101 | 2027 | … | 0 |
| … | … | … | … |  |
| 1,999 | 101 | 1996 | … | 0 |

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

# Processing with DistilBERT

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

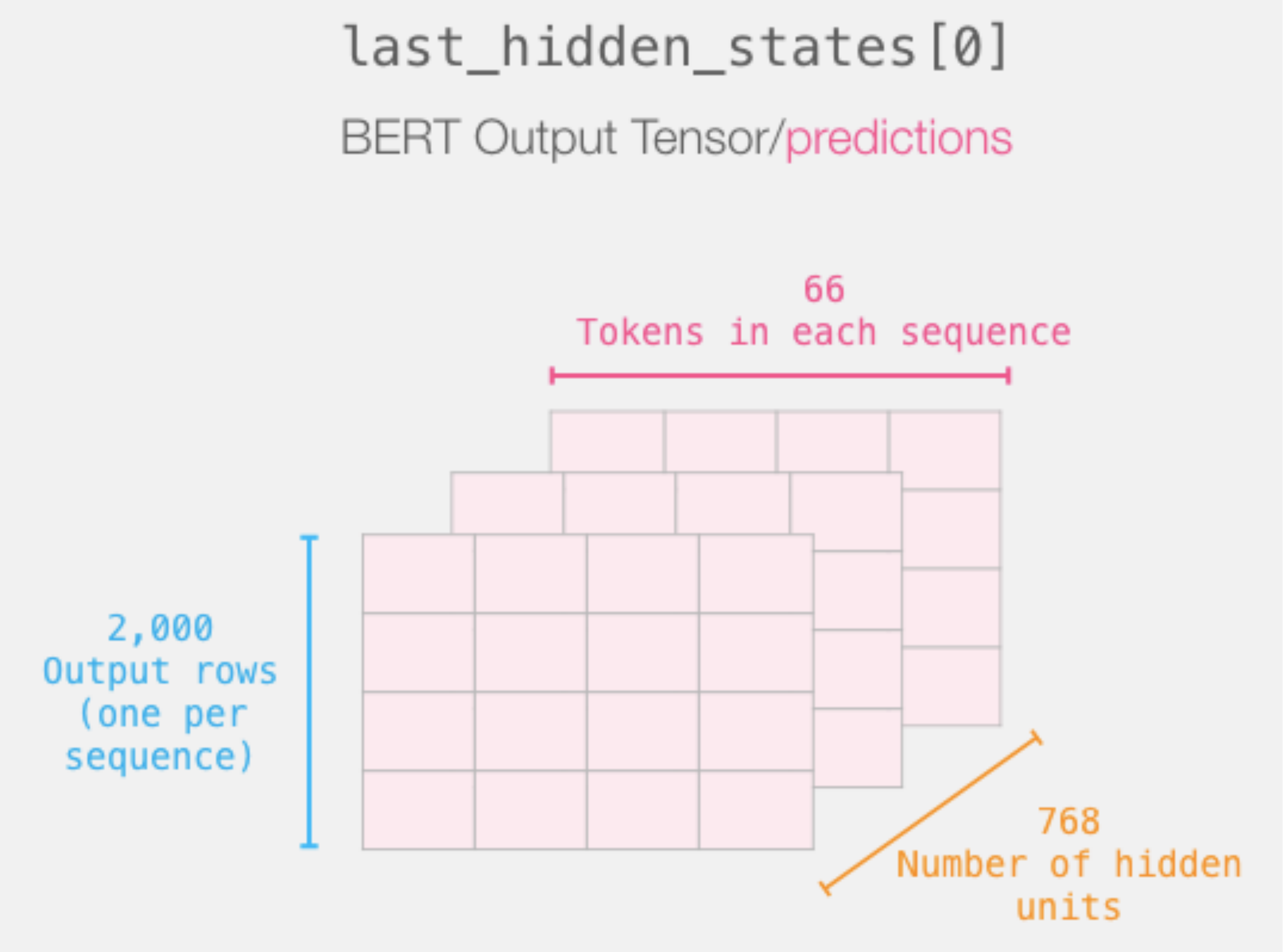


| 0 |
|---|
| a stirring , funny and finally transporting re... |
| apparently reassembled from the cutting room f... |
| they presume their audience wo n't sit still f... |
| this is a visually stunning rumination on love... |
| jonathan parker 's bartleby should have been t... |

|  | 0 | 1 | … | 66 |
|---|---|---|---|---|
| 0 | 101 | 1037 | ... | 0 |
| 1 | 101 | 2027 | ... | 0 |
| … | ... | ... | ... |  |
| 1,999 | 101 | 1996 | ... | 0 |

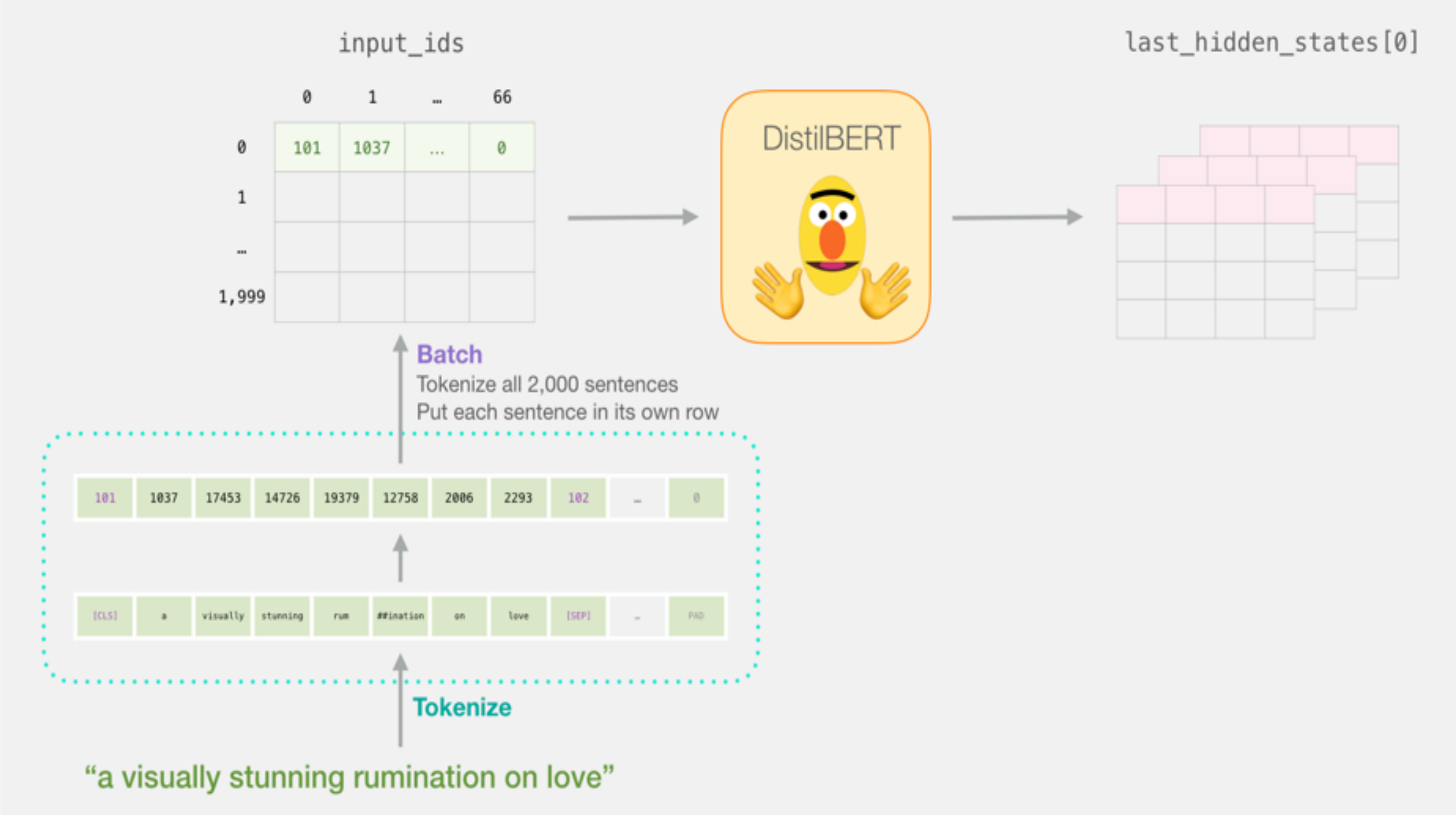
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

# Unpacking the BERT output tensor



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

# Sentence to last_hidden_state[0]



|  | 0 | 1 | … | 66 |
|---|---|---|---|---|
| 0 | 101 | 1037 | … | 0 |
| 1 |  |  |  |  |
| … |  |  |  |  |
| 1,999 |  |  |  |  |

**Batch**
Tokenize all 2,000 sentences
Put each sentence in its own row

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 101 | 1037 | 17453 | 14726 | 19379 | 12758 | 2006 | 2293 | 102 | … | 0 |
| [CLS] | a | visually | stunning | rum | ##ination | on | love | [SEP] | … | PAD |

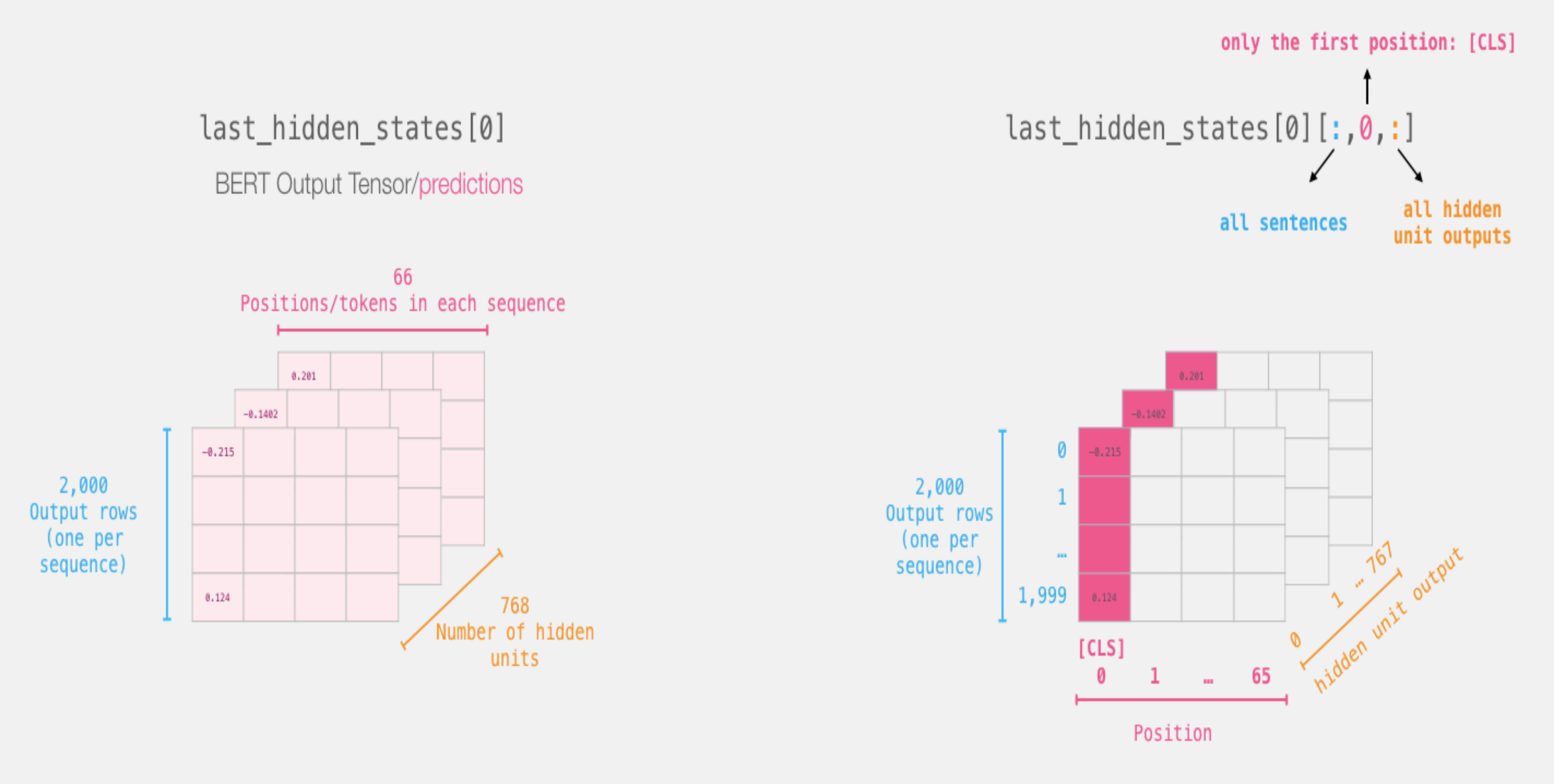
**Tokenize**

“a visually stunning rumination on love”

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

# BERT’s output for the [CLS] tokens

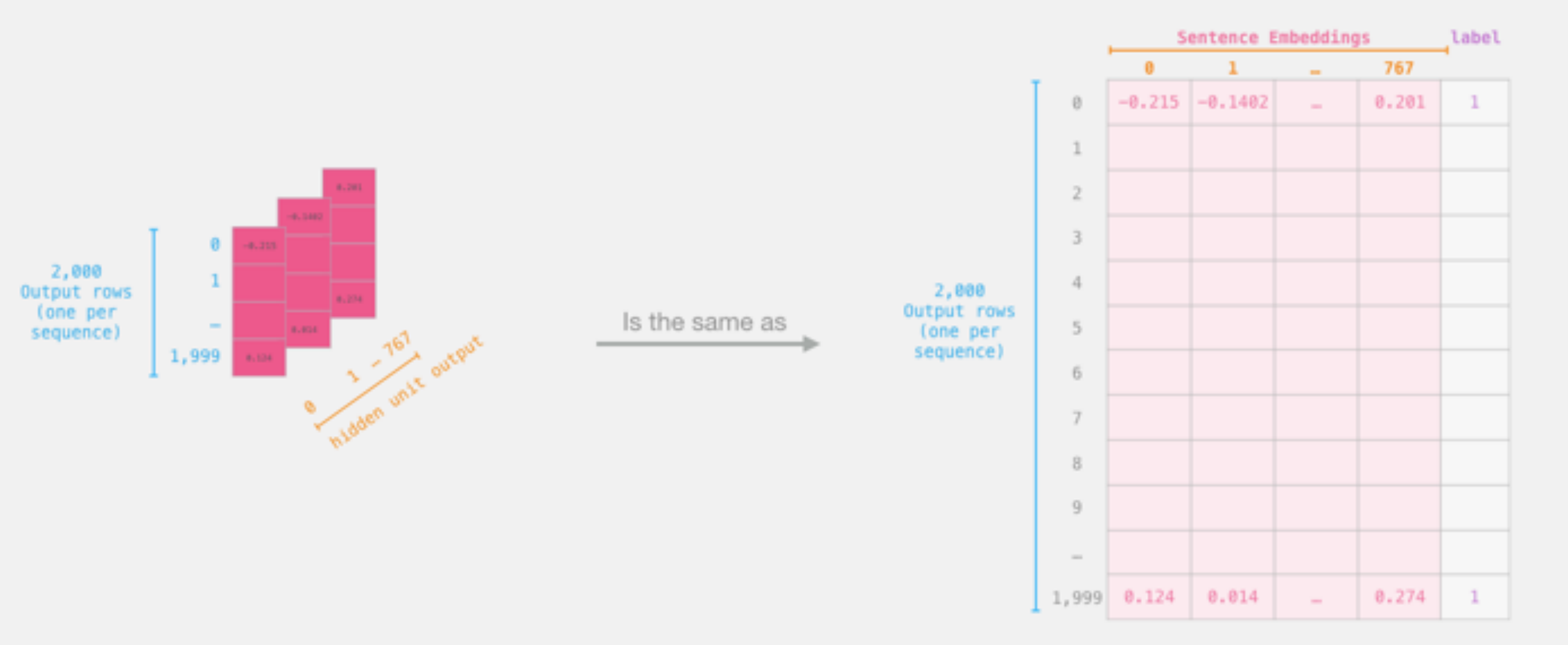
```
# Slice the output for the first position for all the
sequences, take all hidden unit outputs
features = last_hidden_states[0][:,0,:].numpy()
```



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

# The tensor sliced from BERT's output

## Sentence Embeddings

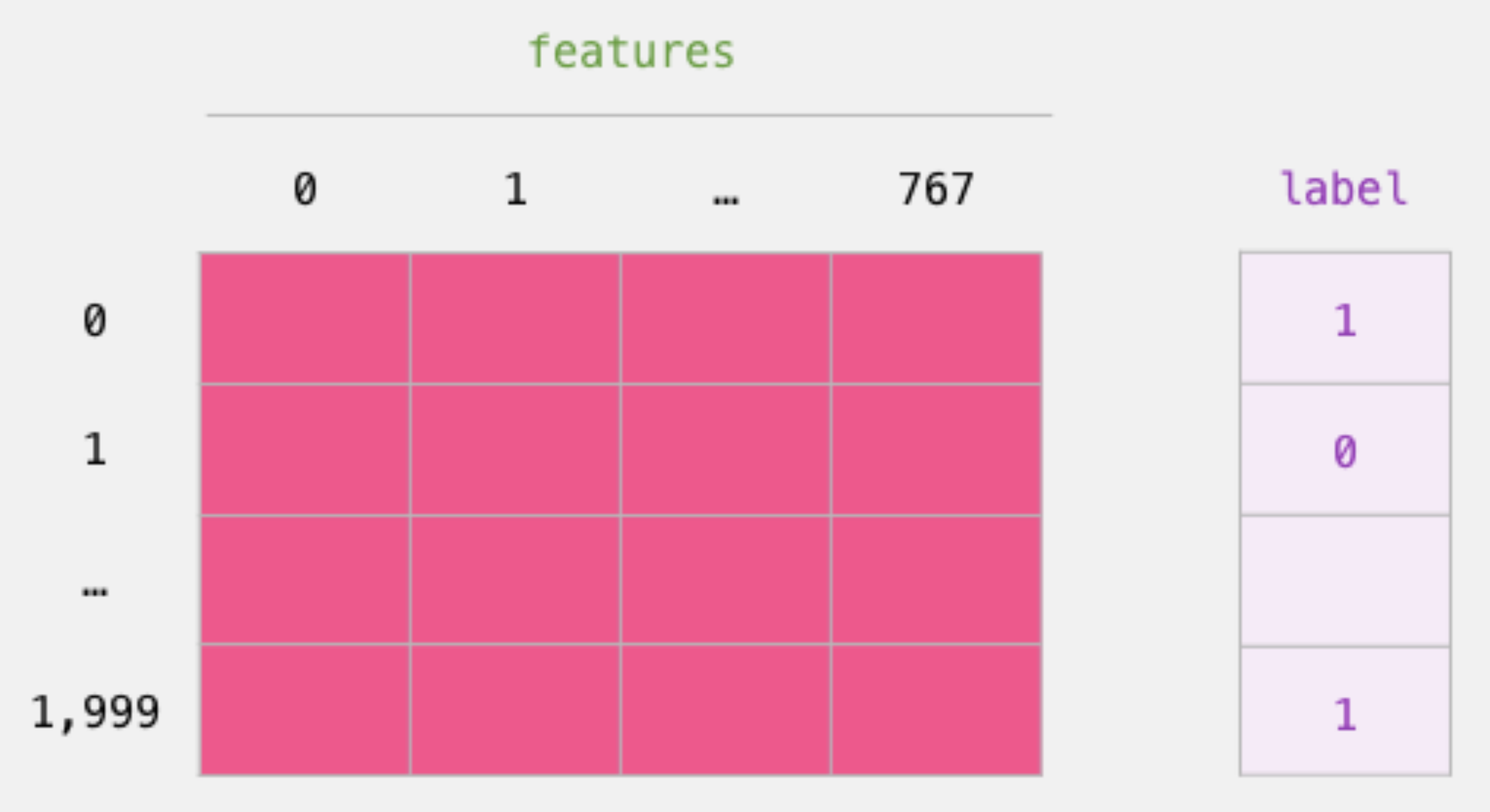


| | Sentence Embeddings | | | | label |
|---|---|---|---|---|---|
| | 0 | 1 | … | 767 | |
| 0 | -0.215 | -0.1402 | … | 0.201 | 1 |
| 1 | | | | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | | | | | |
| 5 | | | | | |
| 6 | | | | | |
| 7 | | | | | |
| 8 | | | | | |
| 9 | | | | | |
| … | | | | | |
| 1,999 | 0.124 | 0.014 | … | 0.274 | 1 |

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

# Dataset for Logistic Regression (768 Features)

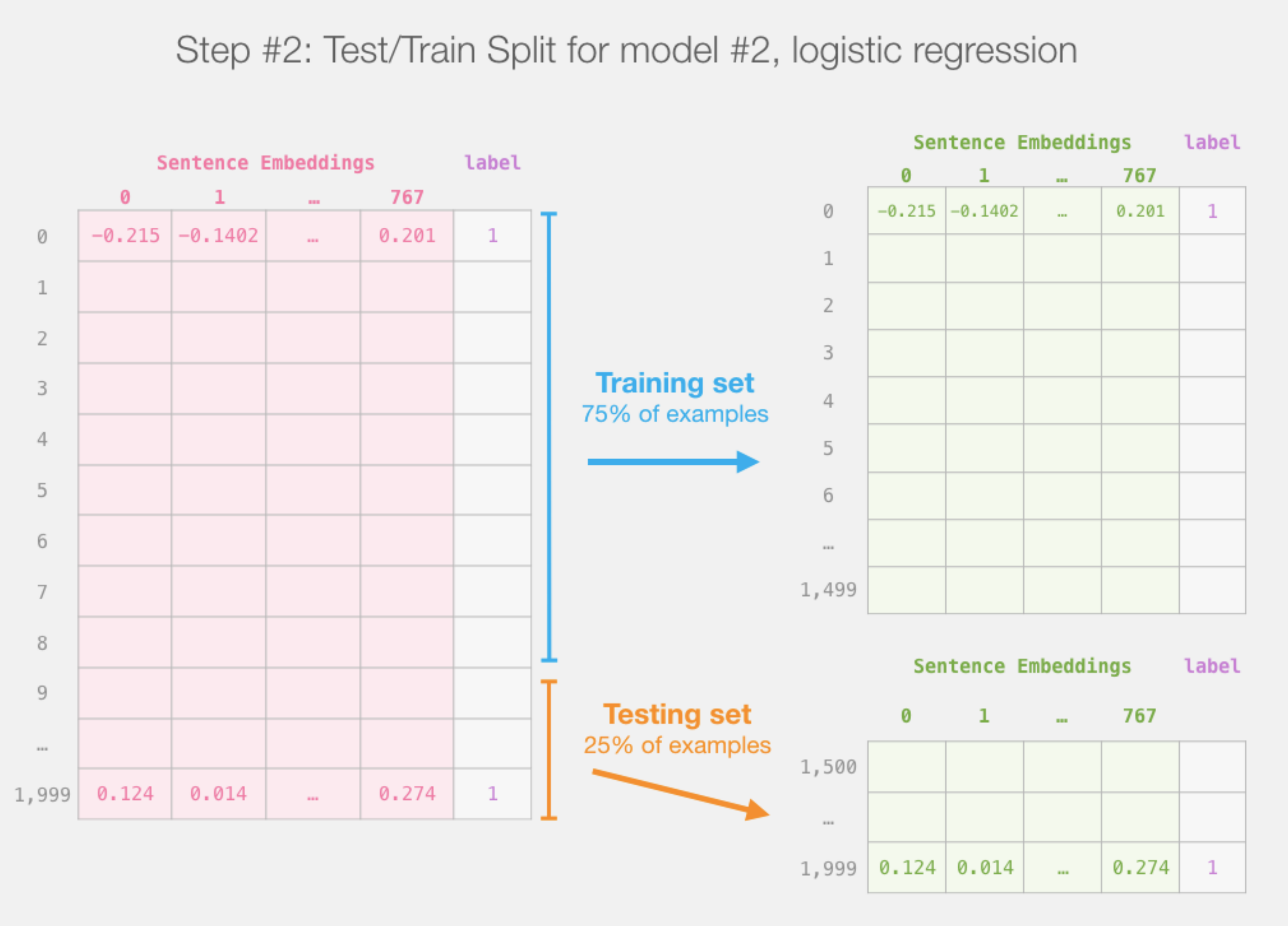
**The features are the output vectors of BERT for the [CLS] token (position #0)**



| | features | | | | |
|---|---|---|---|---|---|
| | 0 | 1 | … | 767 | label |
| 0 | | | | | 1 |
| 1 | | | | | 0 |
| … | | | | | |
| 1,999 | | | | | 1 |

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

```
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

|  | Sentence Embeddings |  |  |  | label |
|---|---|---|---|---|---|
|  | 0 | 1 | … | 767 |  |
| 0 | -0.215 | -0.1402 | … | 0.201 | 1 |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 4 |  |  |  |  |  |
| 5 |  |  |  |  |  |
| 6 |  |  |  |  |  |
| 7 |  |  |  |  |  |
| 8 |  |  |  |  |  |
| 9 |  |  |  |  |  |
| … |  |  |  |  |  |
| 1,999 | 0.124 | 0.014 | … | 0.274 | 1 |

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

|  | Sentence Embeddings |  |  |  | label |
|---|---|---|---|---|---|
|  | 0 | 1 | … | 767 |  |
| 1,500 |  |  |  |  |  |
| … |  |  |  |  |  |
| 1,999 | 0.124 | 0.014 | … | 0.274 | 1 |

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

# Score Benchmarks
# Logistic Regression Model
# on SST-2 Dataset

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

#Testing
lr_clf.score(test_features, test_labels)

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

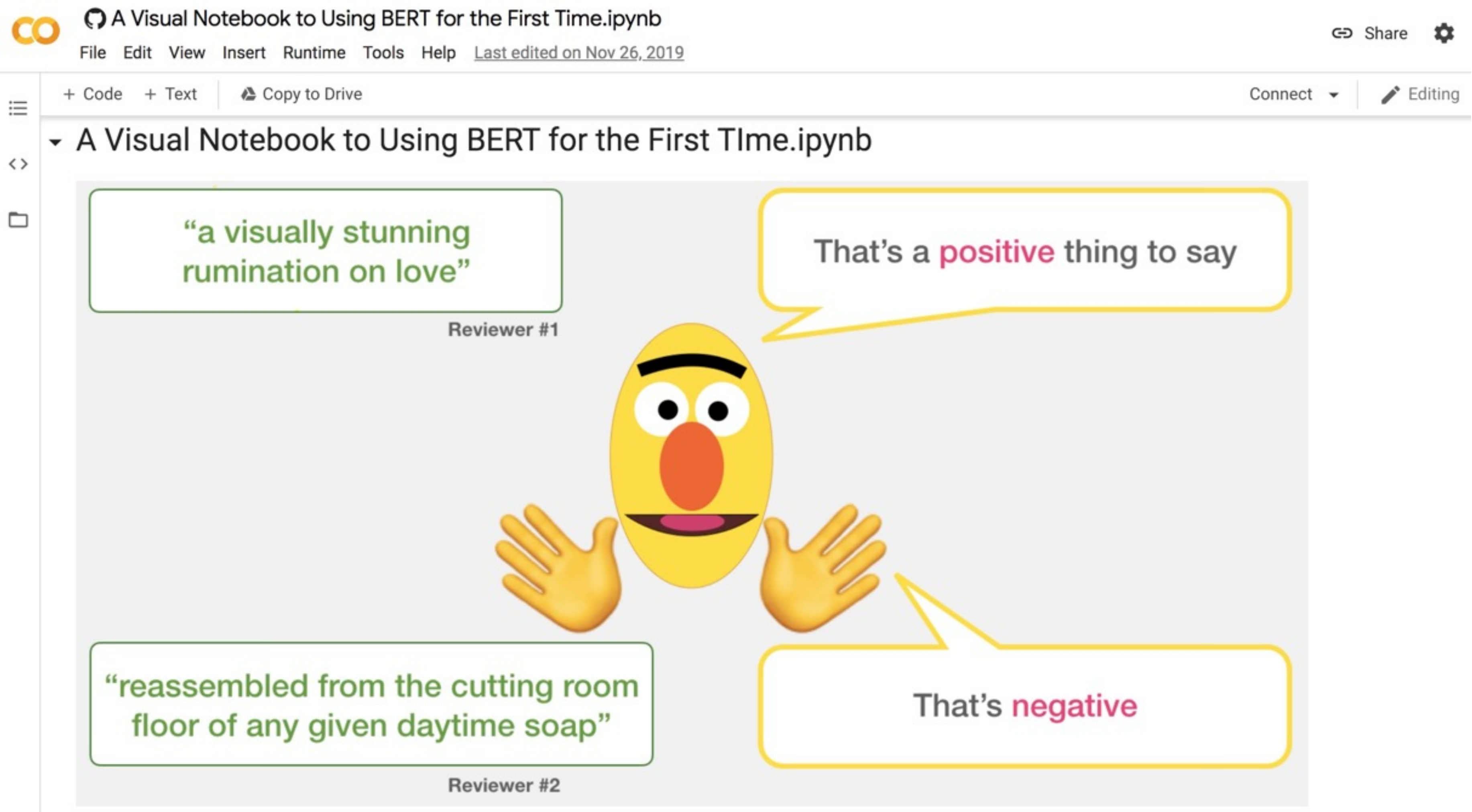
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

# Sentiment Classification: SST2
# Sentences from movie reviews

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

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

# A Visual Notebook to Using BERT for the First Time



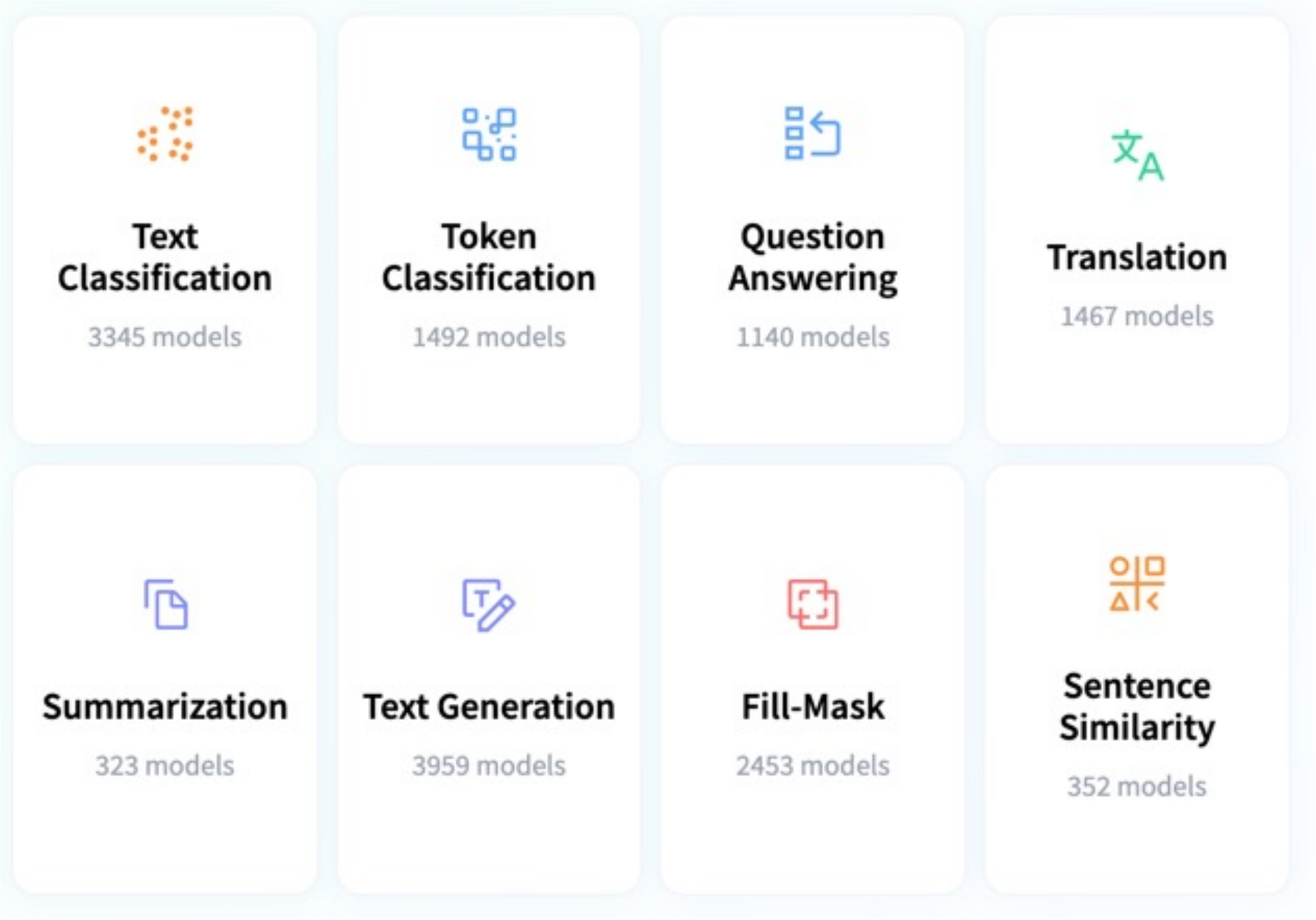
A Visual Notebook to Using BERT for the First Time.ipynb
File Edit View Insert Runtime Tools Help Last edited on Nov 26, 2019
Share
+ Code + Text Copy to Drive
Connect
Editing

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

"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

# Hugging Face Tasks
# Natural Language Processing



Text Classification
3345 models

Token Classification
1492 models

Question Answering
1140 models

Translation
1467 models

Summarization
323 models

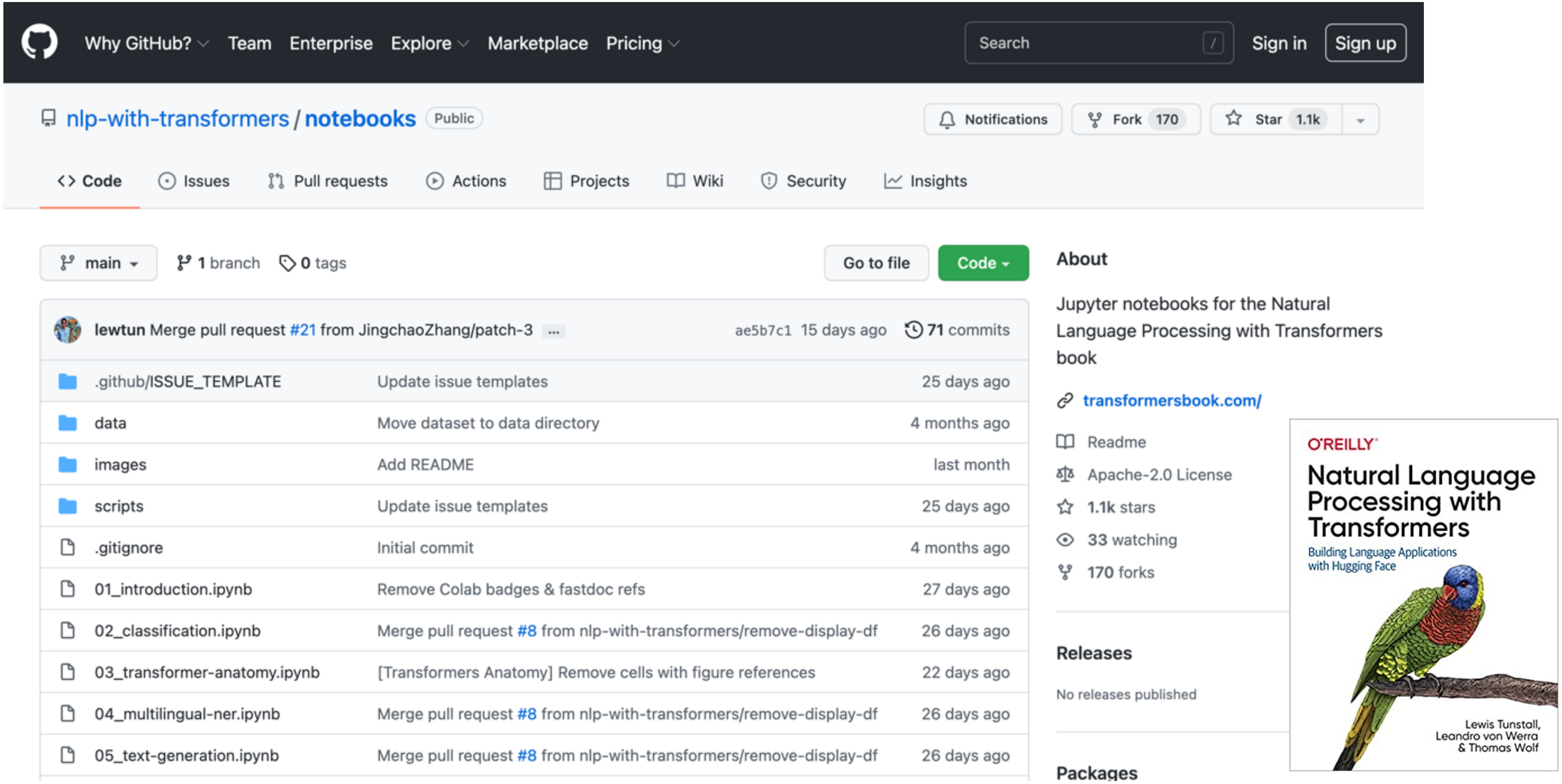
Text Generation
3959 models

Fill-Mask
2453 models

Sentence Similarity
352 models

https://huggingface.co/tasks

# NLP with Transformers Github



nlp-with-transformers / notebooks Public

lewtun Merge pull request #21 from JingchaoZhang/patch-3 … ae5b7c1 15 days ago 71 commits

| | | |
|---|---|---|
| .github/ISSUE_TEMPLATE | Update issue templates | 25 days ago |
| data | Move dataset to data directory | 4 months ago |
| images | Add README | last month |
| scripts | Update issue templates | 25 days ago |
| .gitignore | Initial commit | 4 months ago |
| 01_introduction.ipynb | Remove Colab badges & fastdoc refs | 27 days ago |
| 02_classification.ipynb | Merge pull request #8 from nlp-with-transformers/remove-display-df | 26 days ago |
| 03_transformer-anatomy.ipynb | [Transformers Anatomy] Remove cells with figure references | 22 days ago |
| 04_multilingual-ner.ipynb | Merge pull request #8 from nlp-with-transformers/remove-display-df | 26 days ago |
| 05_text-generation.ipynb | Merge pull request #8 from nlp-with-transformers/remove-display-df | 26 days ago |

About

Jupyter notebooks for the Natural Language Processing with Transformers book

transformersbook.com/

Readme

Apache-2.0 License

1.1k stars

33 watching

170 forks

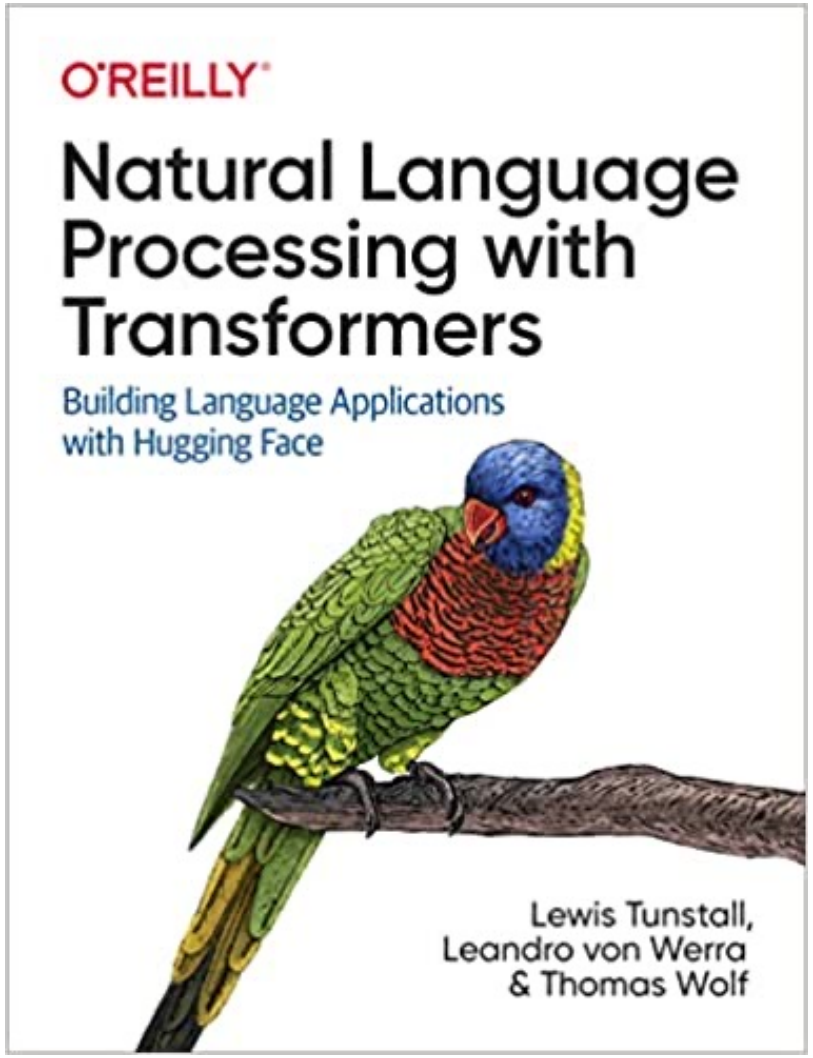
Releases

No releases published

Packages

https://github.com/nlp-with-transformers/notebooks

# NLP with Transformers Github Notebooks

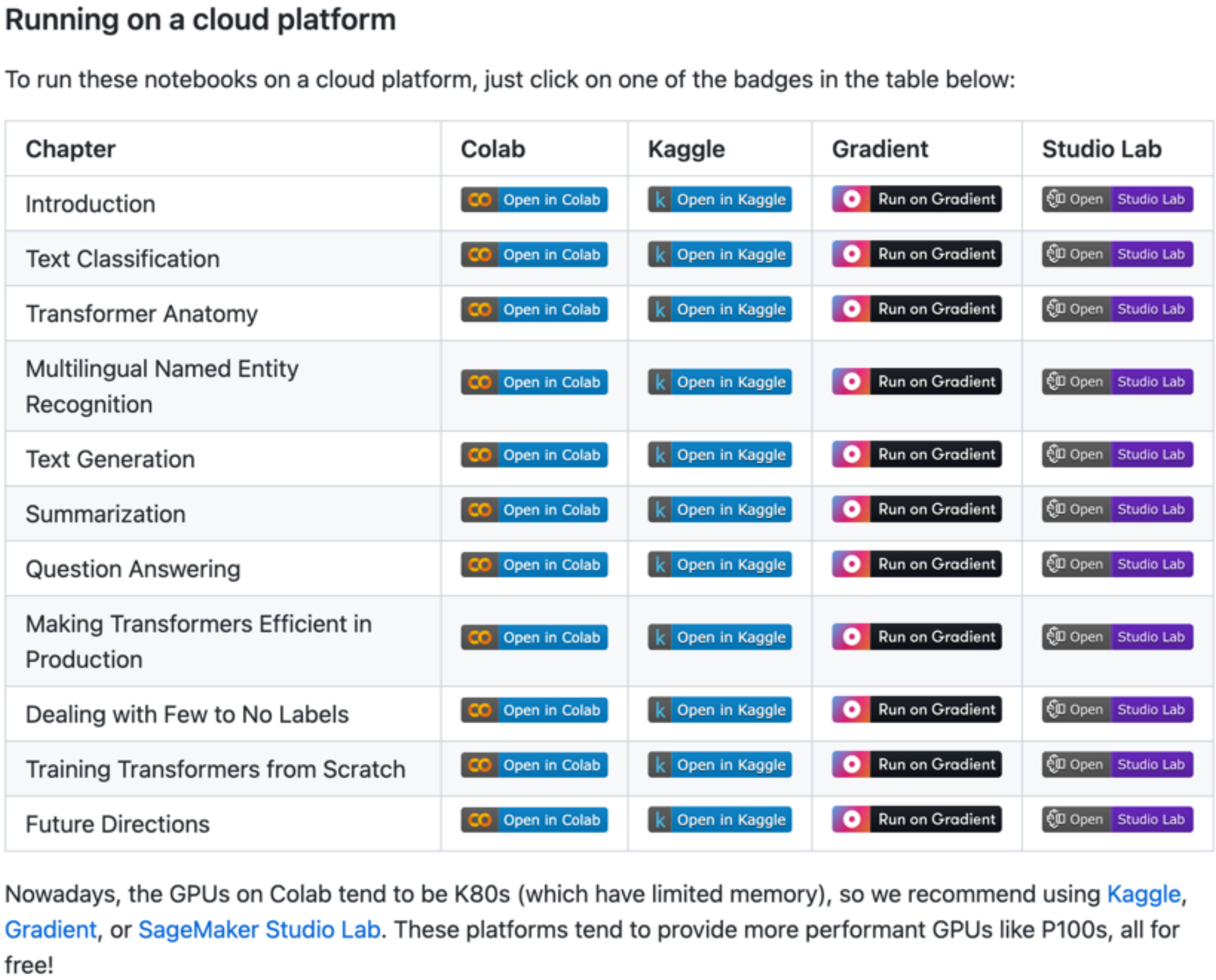


O'REILLY

Natural Language Processing with Transformers

Building Language Applications with Hugging Face

Lewis Tunstall, Leandro von Werra & Thomas Wolf



### Running on a cloud platform

To run these notebooks on a cloud platform, just click on one of the badges in the table below:

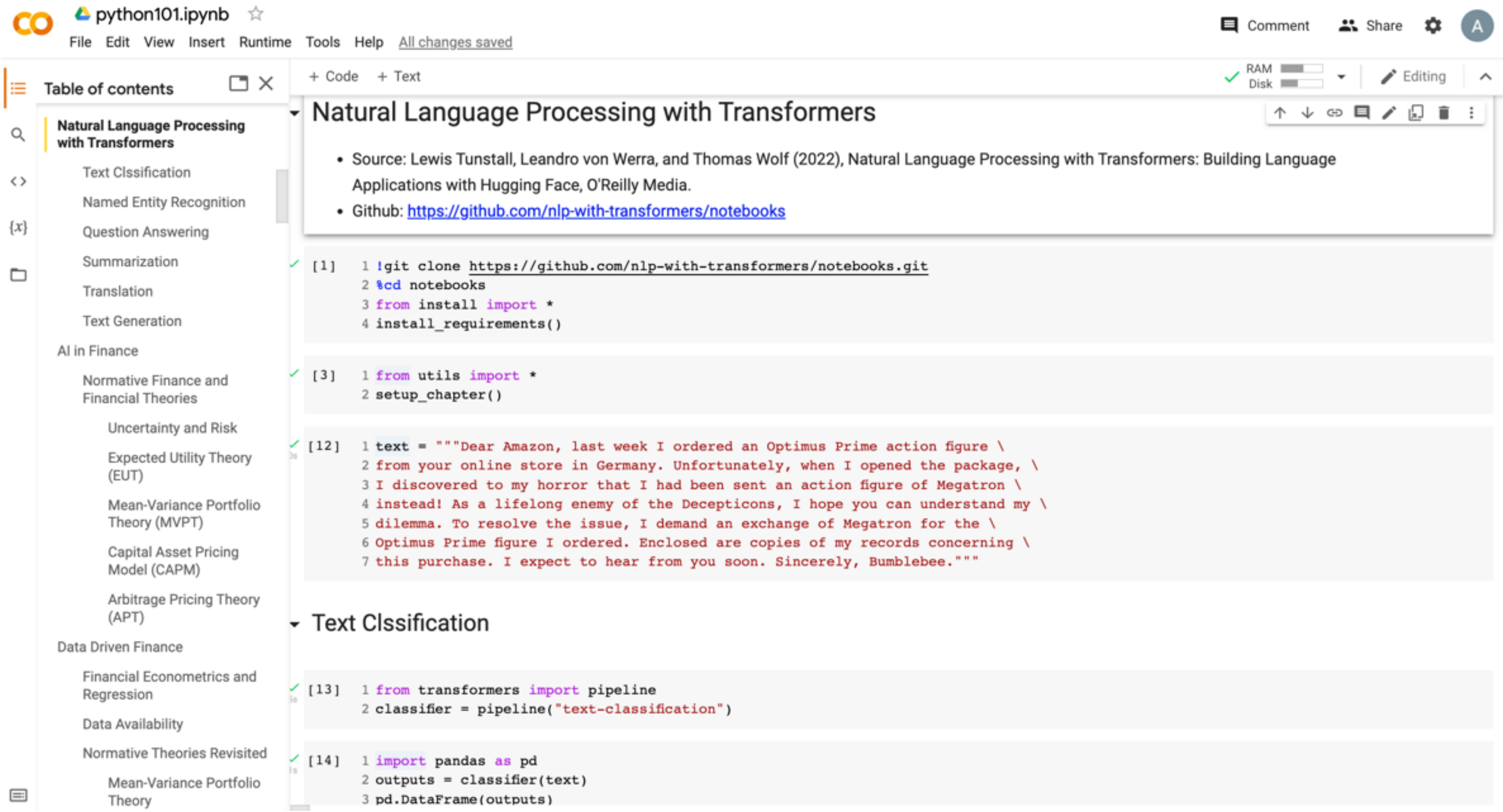
| Chapter | Colab | Kaggle | Gradient | Studio Lab |
|---|---|---|---|---|
| Introduction | Open in Colab | Open in Kaggle | Run on Gradient | Open Studio Lab |
| Text Classification | Open in Colab | Open in Kaggle | Run on Gradient | Open Studio Lab |
| Transformer Anatomy | Open in Colab | Open in Kaggle | Run on Gradient | Open Studio Lab |
| Multilingual Named Entity Recognition | Open in Colab | Open in Kaggle | Run on Gradient | Open Studio Lab |
| Text Generation | Open in Colab | Open in Kaggle | Run on Gradient | Open Studio Lab |
| Summarization | Open in Colab | Open in Kaggle | Run on Gradient | Open Studio Lab |
| Question Answering | Open in Colab | Open in Kaggle | Run on Gradient | Open Studio Lab |
| Making Transformers Efficient in Production | Open in Colab | Open in Kaggle | Run on Gradient | Open Studio Lab |
| Dealing with Few to No Labels | Open in Colab | Open in Kaggle | Run on Gradient | Open Studio Lab |
| Training Transformers from Scratch | Open in Colab | Open in Kaggle | Run on Gradient | Open Studio Lab |
| Future Directions | Open in Colab | Open in Kaggle | Run on Gradient | Open Studio Lab |

Nowadays, the GPUs on Colab tend to be K80s (which have limited memory), so we recommend using Kaggle, Gradient, or SageMaker Studio Lab. These platforms tend to provide more performant GPUs like P100s, all for free!

https://github.com/nlp-with-transformers/notebooks

# Python in Google Colab (Python101)

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



**Natural Language Processing with Transformers**

- Text Clssification
- Named Entity Recognition
- Question Answering
- Summarization
- Translation
- Text Generation

AI in Finance

- Normative Finance and Financial Theories
  - Uncertainty and Risk
  - Expected Utility Theory (EUT)
  - Mean-Variance Portfolio Theory (MVPT)
  - Capital Asset Pricing Model (CAPM)
  - Arbitrage Pricing Theory (APT)

Data Driven Finance

- Financial Econometrics and Regression
- Data Availability
- Normative Theories Revisited
  - Mean-Variance Portfolio Theory

## Natural Language Processing with Transformers

- Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Github: https://github.com/nlp-with-transformers/notebooks

[1]
```
!git clone https://github.com/nlp-with-transformers/notebooks.git
%cd notebooks
from install import *
install_requirements()
```

[3]
```
from utils import *
setup_chapter()
```

[12]
```
text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
from your online store in Germany. Unfortunately, when I opened the package, \
I discovered to my horror that I had been sent an action figure of Megatron \
instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
dilemma. To resolve the issue, I demand an exchange of Megatron for the \
Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
```

## Text Clssification

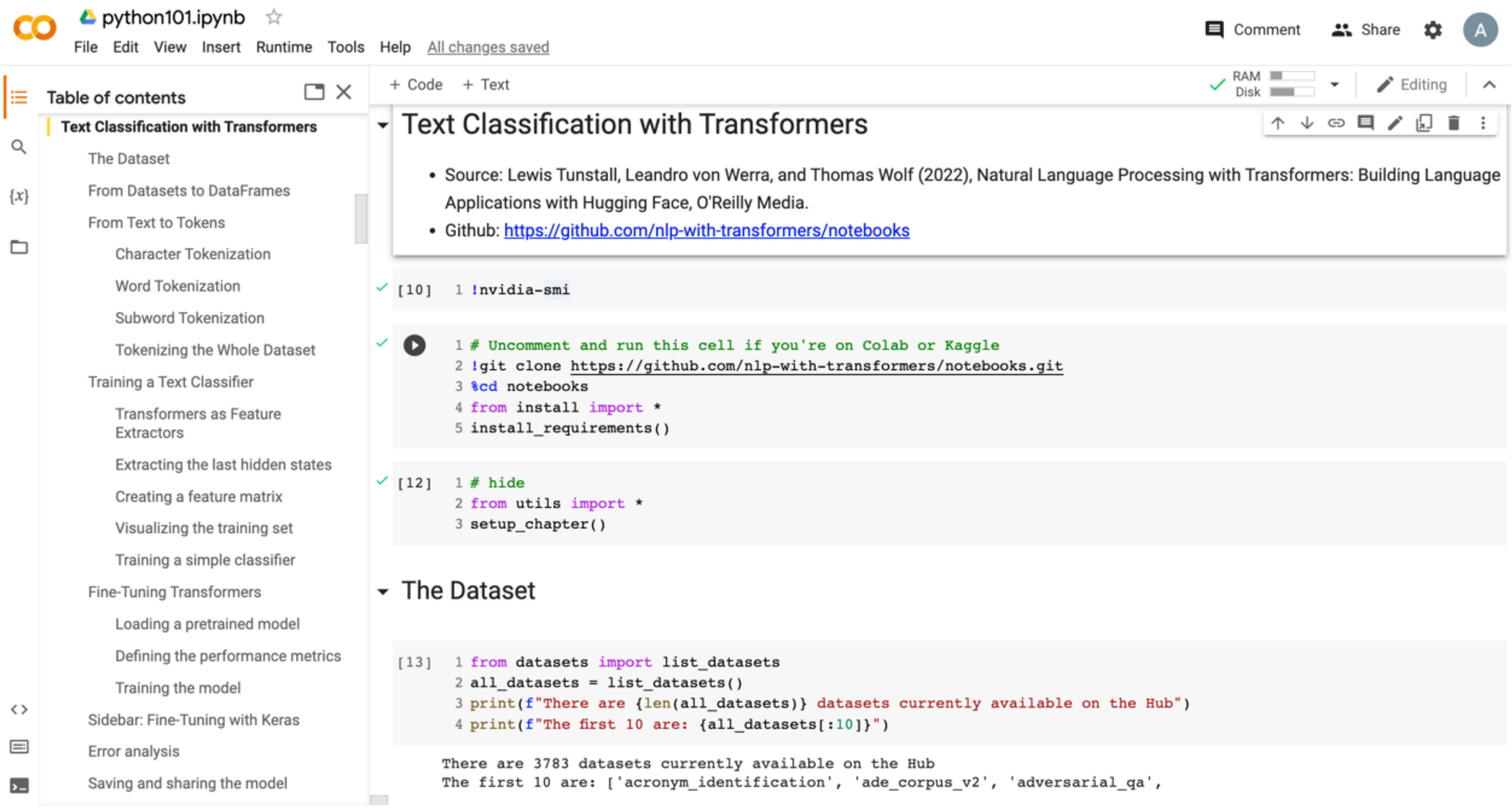
[13]
```
from transformers import pipeline
classifier = pipeline("text-classification")
```

[14]
```
import pandas as pd
outputs = classifier(text)
pd.DataFrame(outputs)
```

https://tinyurl.com/aintpupython101

# Python in Google Colab (Python101)

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



Text Classification with Transformers
The Dataset
From Datasets to DataFrames
From Text to Tokens
Character Tokenization
Word Tokenization
Subword Tokenization
Tokenizing the Whole Dataset
Training a Text Classifier
Transformers as Feature Extractors
Extracting the last hidden states
Creating a feature matrix
Visualizing the training set
Training a simple classifier
Fine-Tuning Transformers
Loading a pretrained model
Defining the performance metrics
Training the model
Sidebar: Fine-Tuning with Keras
Error analysis
Saving and sharing the model

## Text Classification with Transformers

- Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Github: https://github.com/nlp-with-transformers/notebooks

[10]
```
!nvidia-smi
```

```
# Uncomment and run this cell if you're on Colab or Kaggle
!git clone https://github.com/nlp-with-transformers/notebooks.git
%cd notebooks
from install import *
install_requirements()
```

[12]
```
# hide
from utils import *
setup_chapter()
```

## The Dataset

[13]
```
from datasets import list_datasets
all_datasets = list_datasets()
print(f"There are {len(all_datasets)} datasets currently available on the Hub")
print(f"The first 10 are: {all_datasets[:10]}")
```

```
There are 3783 datasets currently available on the Hub
The first 10 are: ['acronym_identification', 'ade_corpus_v2', 'adversarial_qa',
```

https://tinyurl.com/aintpupython101

# Summary

- Text Classification and Sentiment Analysis
    - Dataset
    - Tokenizer
    - Training a Text Classifier
    - Fine-Tuning Transformers

# References

- Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers:  Building Language Applications with Hugging Face, O'Reilly Media.
- Denis Rothman (2021), Transformers for Natural Language Processing: Build innovative deep neural network architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and more, Packt Publishing.
- Savaş Yıldırım and Meysam Asgari-Chenaghlu (2021), Mastering Transformers: Build state-of-the-art models from scratch with advanced natural language processing techniques, Packt Publishing.
- Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao (2021), "Deep learning--based text classification: a comprehensive review." ACM Computing Surveys (CSUR) 54, no. 3 (2021): 1-40.
- Sowmya Vajjala, Bodhisattwa Majumder, Anuj Gupta (2020), Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems, O'Reilly Media.
- 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.
- Charu C. Aggarwal (2018), Machine Learning for Text, Springer.
- 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.
- Jake VanderPlas (2016), Python Data Science Handbook: Essential Tools for Working with Data, O'Reilly Media.
- 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.
- The Super Duper NLP Repo, https://notebooks.quantumstat.com/
- Jay Alammar (2018), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/
- 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/
- NLP with Transformer, https://github.com/nlp-with-transformers/notebooks
- Min-Yuh Day (2022), Python 101, https://tinyurl.com/aintpupython101