Artificial Intelligence

国立臺北大學
National Taipei University

Deep Learning for Natural Language Processing

1111AI08
MBA, IM, NTPU (M6132) (Fall 2022)
Wed 2, 3, 4 (9:10-12:00) (B8F40)

https://meet.google.com/miy-fbif-max

aws educate Cloud Ambassador 2020 Cohort

Min-Yuh Day, Ph.D,
Associate Professor
Institute of Information Management, National Taipei University
https://web.ntpu.edu.tw/~myday

2022-11-16

Syllabus

Week	Date	Subject/Topics
1	2022/09/14	Introduction to Artificial Intelligence
2	2022/09/21	Artificial Intelligence and Intelligent Agents
3	2022/09/28	Problem Solving
4	2022/10/05	Knowledge, Reasoning and Knowledge Representation; Uncertain Knowledge and Reasoning
5	2022/10/12	Case Study on Artificial Intelligence I
6	2022/10/19	Machine Learning: Supervised and Unsupervised Learning

Syllabus

Week Date Subject/Topics

7 2022/10/26 The Theory of Learning and Ensemble Learning

8 2022/11/02 Midterm Project Report

9 2022/11/09 Deep Learning and Reinforcement Learning

10 2022/11/16 Deep Learning for Natural Language Processing

11 2022/11/23 Invited Talk: AI for Information Retrieval

12 2022/11/30 Case Study on Artificial Intelligence II

Syllabus

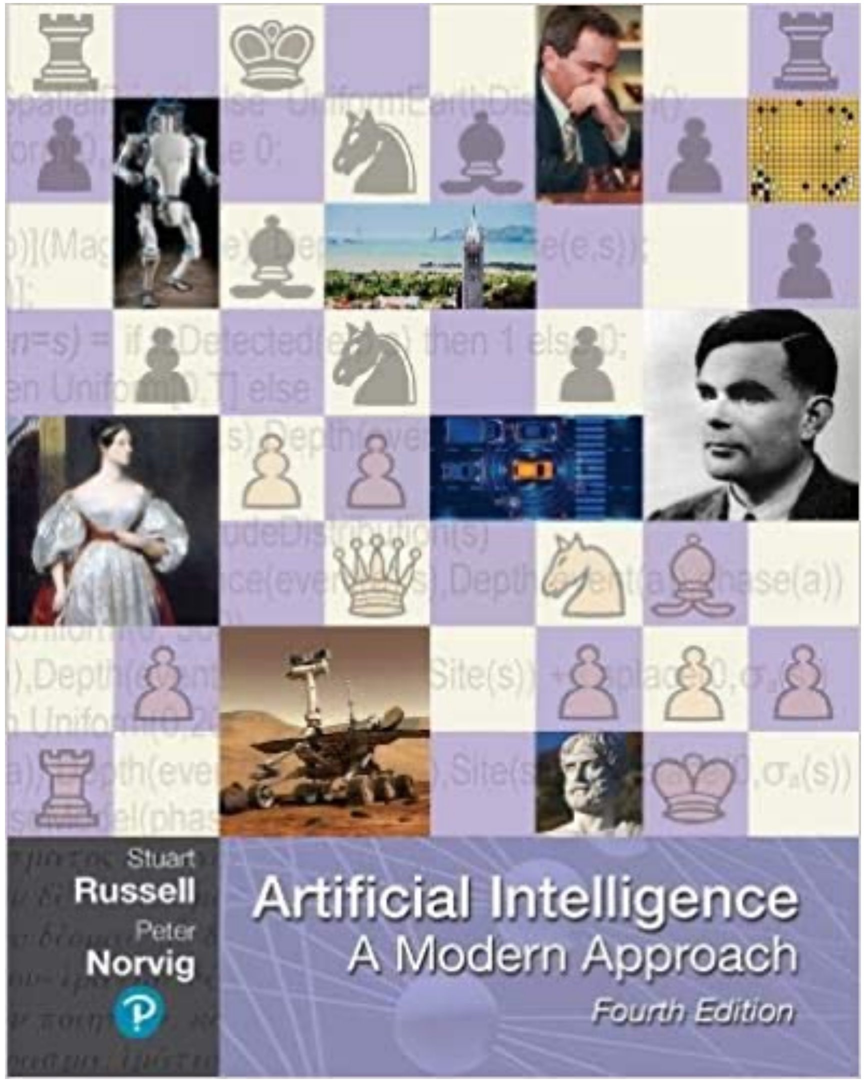
Week	Date	Subject/Topics
13	2022/12/07	Computer Vision and Robotics
14	2022/12/14	Philosophy and Ethics of AI and the Future of AI
15	2022/12/21	Final Project Report I
16	2022/12/28	Final Project Report II
17	2023/01/04	Self-learning
18	2023/01/11	Self-learning

Deep Learning for Natural Language Processing

Outline

- **Word Embeddings**
- **Recurrent Neural Networks for NLP**
- **Sequence-to-Sequence Models**
- **The Transformer Architecture**
- **Pretraining and Transfer Learning**
- **State of the art (SOTA)**

Stuart Russell and Peter Norvig (2020),
Artificial Intelligence: A Modern Approach,
4th Edition, Pearson



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

https://www.amazon.com/Artificial-Intelligence-A-Modern-Approach/dp/0134610997/

Artificial Intelligence: A Modern Approach

1. Artificial Intelligence
2. Problem Solving
3. Knowledge and Reasoning
4. Uncertain Knowledge and Reasoning
5. Machine Learning
6. Communicating, Perceiving, and Acting
7. Philosophy and Ethics of AI

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Artificial Intelligence: Communicating, perceiving, and acting

Artificial Intelligence:

6. Communicating, Perceiving, and Acting

- Natural Language Processing
- Deep Learning for Natural Language Processing
- Computer Vision
- Robotics

Artificial Intelligence:
Natural Language Processing

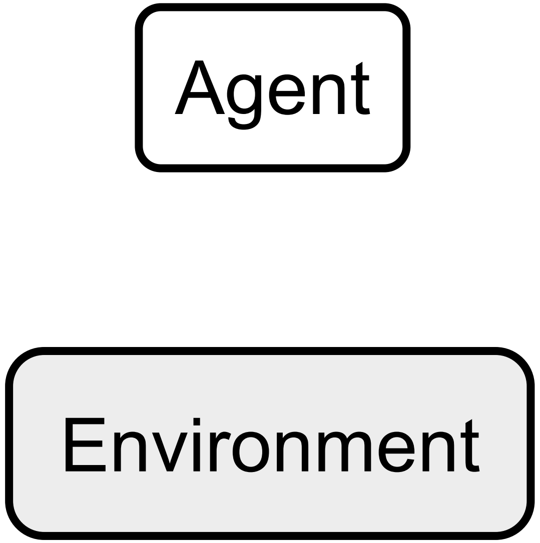
- Language Models
- Grammar
- Parsing
- Augmented Grammars
- Complications of Real Natural Language
- Natural Language Tasks

Artificial Intelligence: Deep Learning for Natural Language Processing

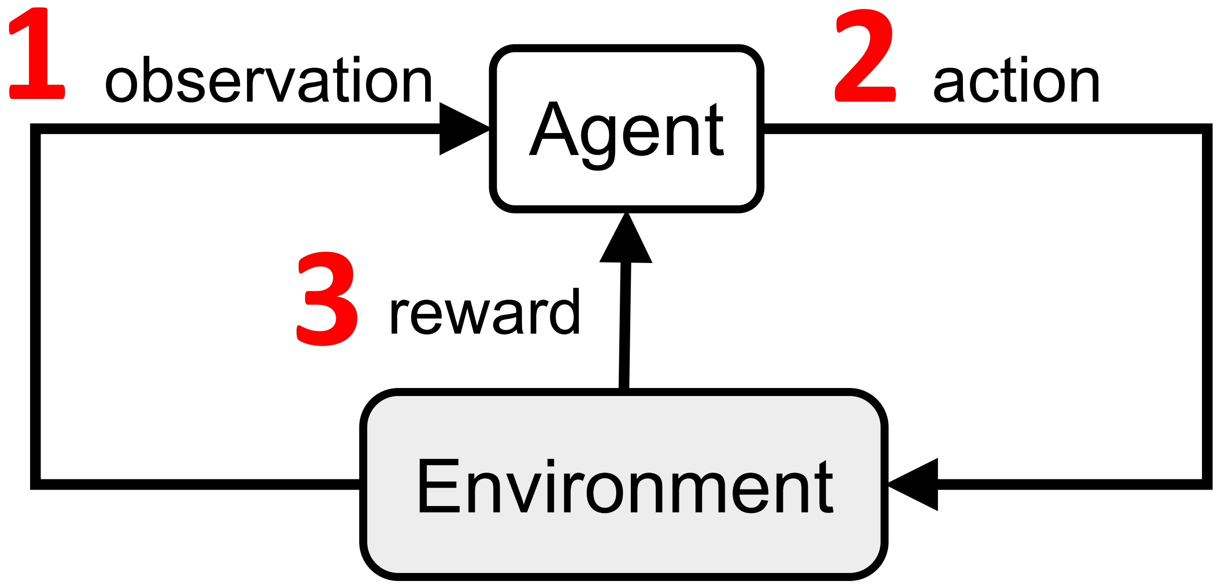
- Word Embeddings
- Recurrent Neural Networks for NLP
- Sequence-to-Sequence Models
- The Transformer Architecture
- Pretraining and Transfer Learning
- State of the art (SOTA)

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Reinforcement Learning (DL)

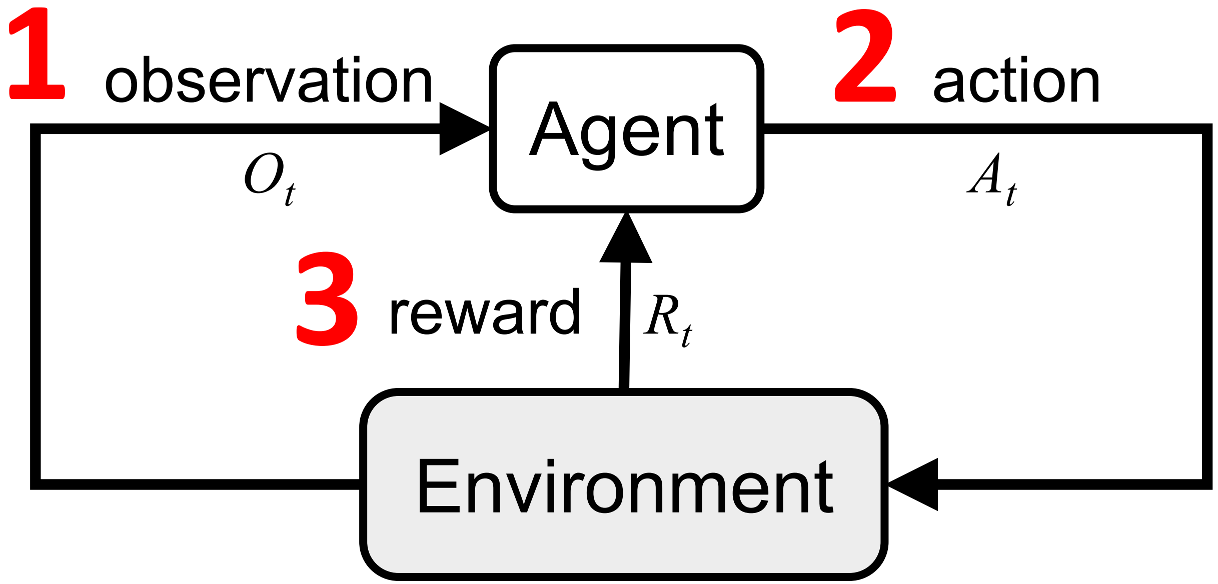


Reinforcement Learning (DL)



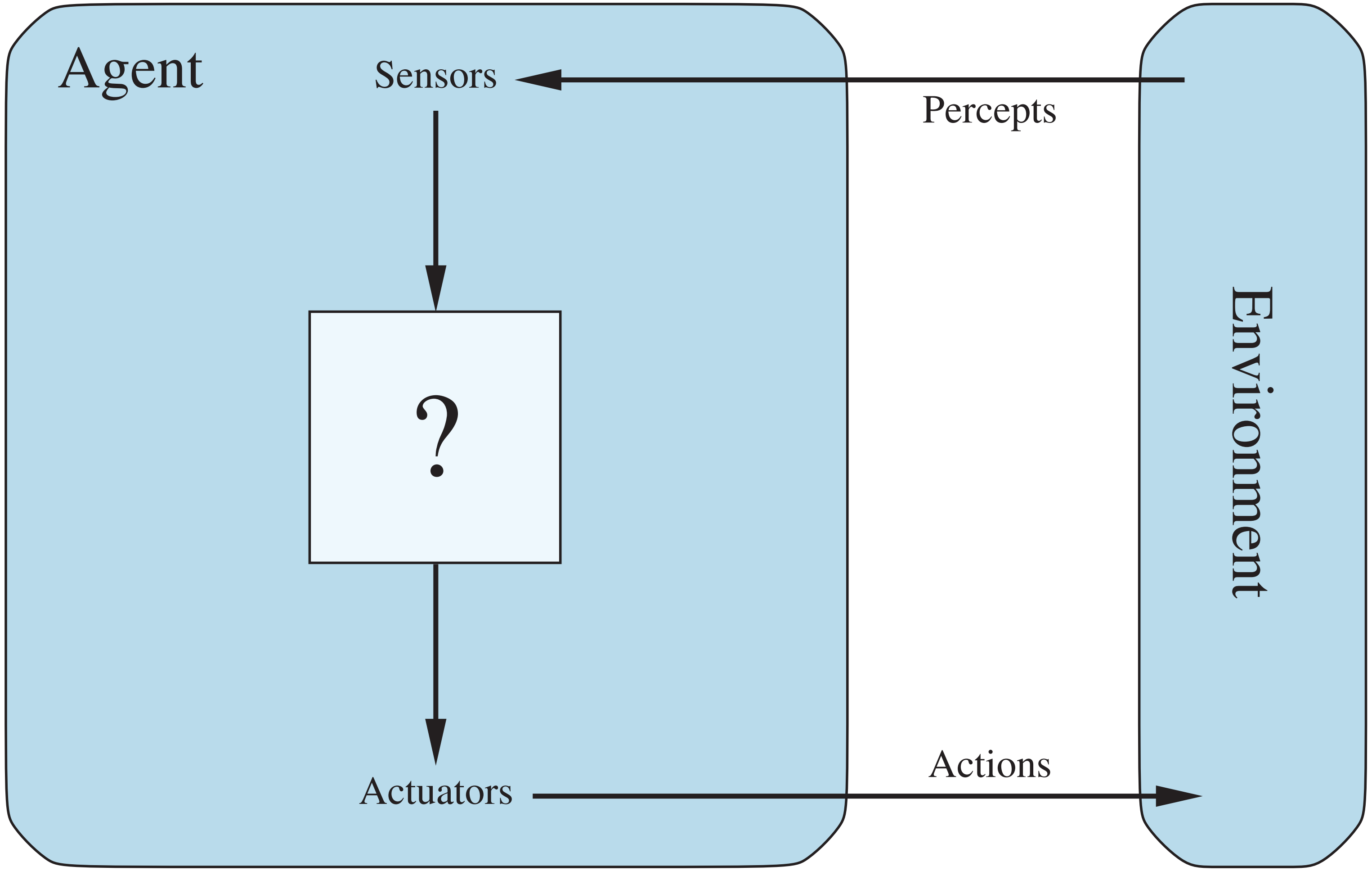
Source: Richard S. Sutton & Andrew G. Barto (2018), Reinforcement Learning: An Introduction, 2nd Edition, A Bradford Book.

Reinforcement Learning (DL)



Source: Richard S. Sutton & Andrew G. Barto (2018), Reinforcement Learning: An Introduction, 2nd Edition, A Bradford Book.

Agents interact with environments through sensors and actuators



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

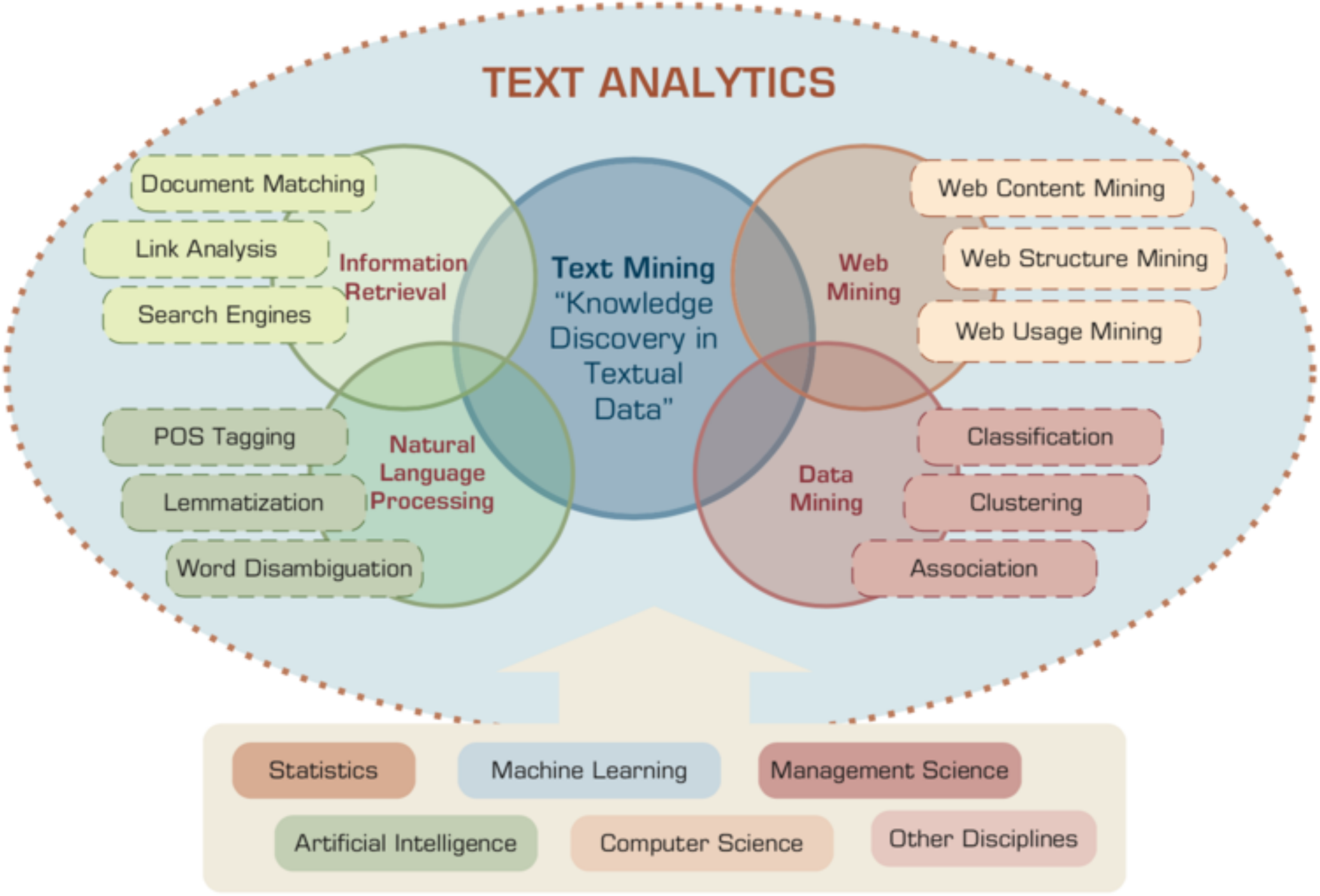
AI Acting Humanly:
The Turing Test Approach
(Alan Turing, 1950)

- Knowledge Representation
- Automated Reasoning
- Machine Learning (ML)
  - Deep Learning (DL)
- Computer Vision (Image, Video)
- Natural Language Processing (NLP)
- Robotics

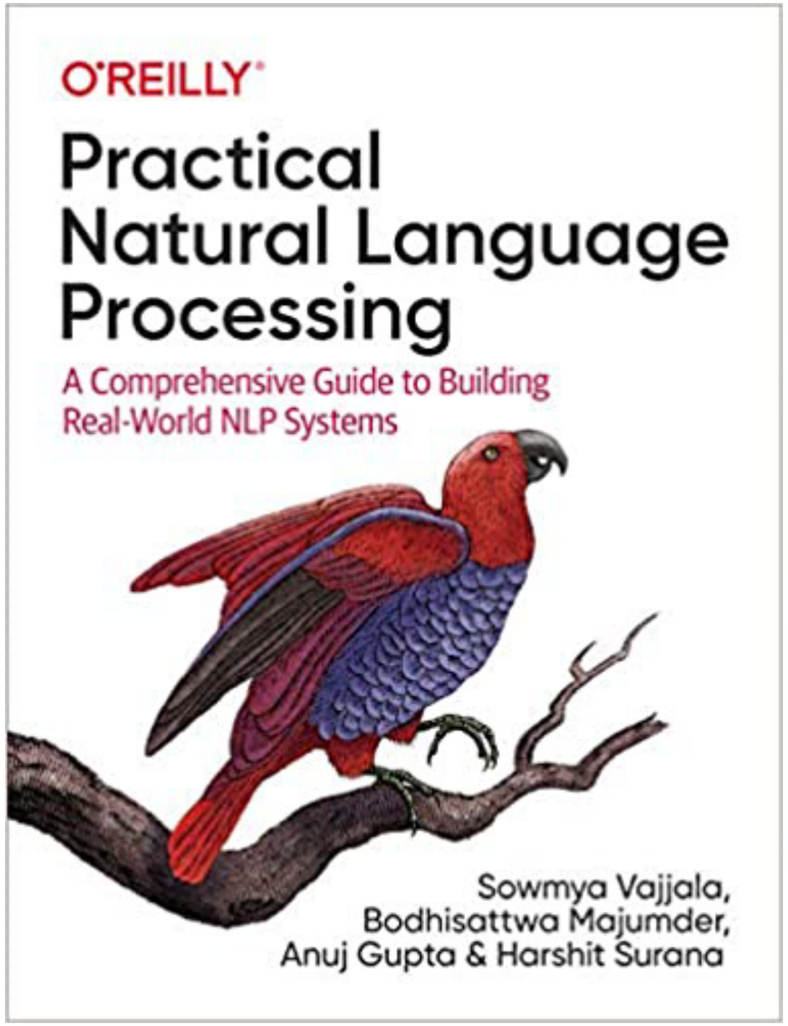
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

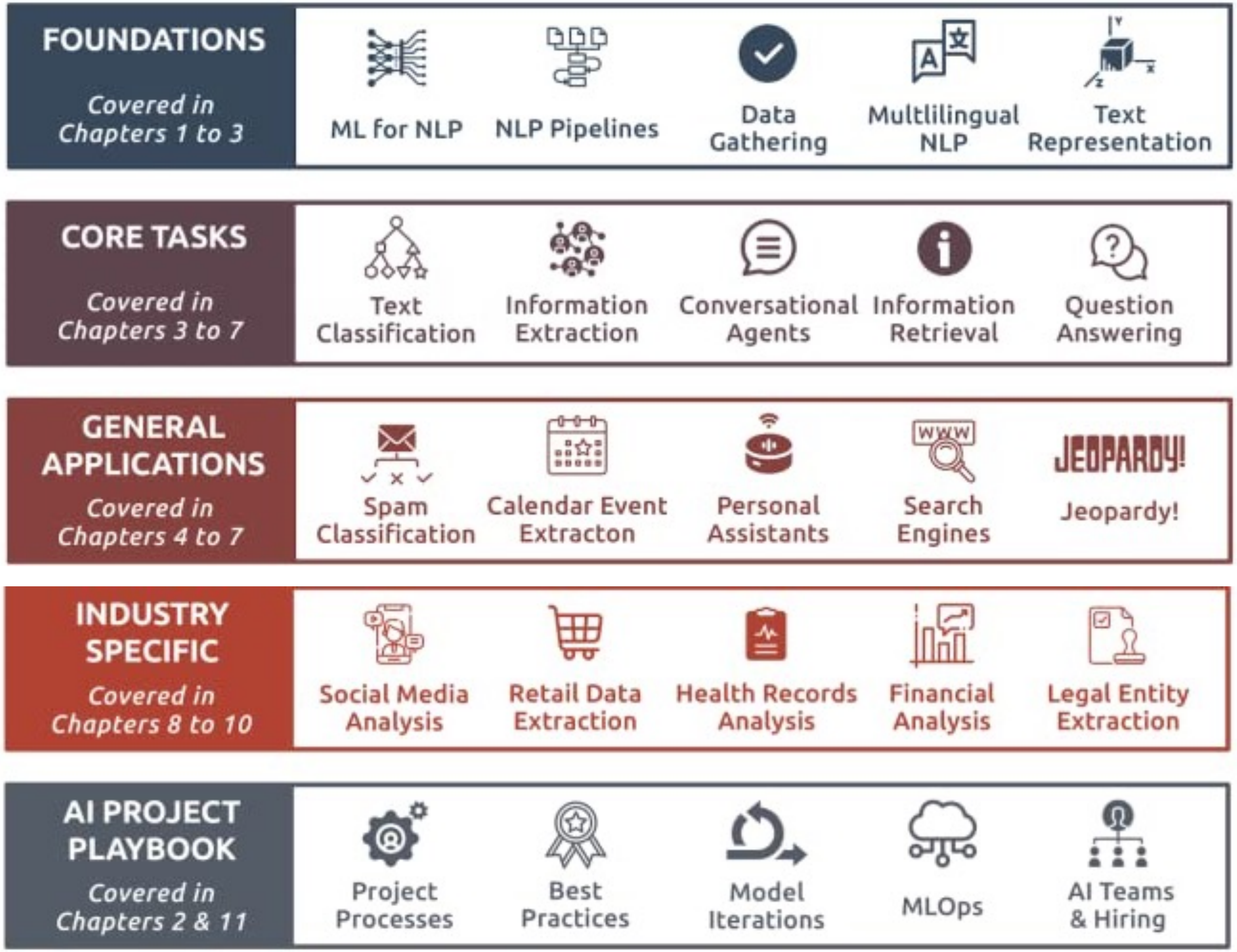
Deep Learning for Natural Language Processing

AI for Text Analytics



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

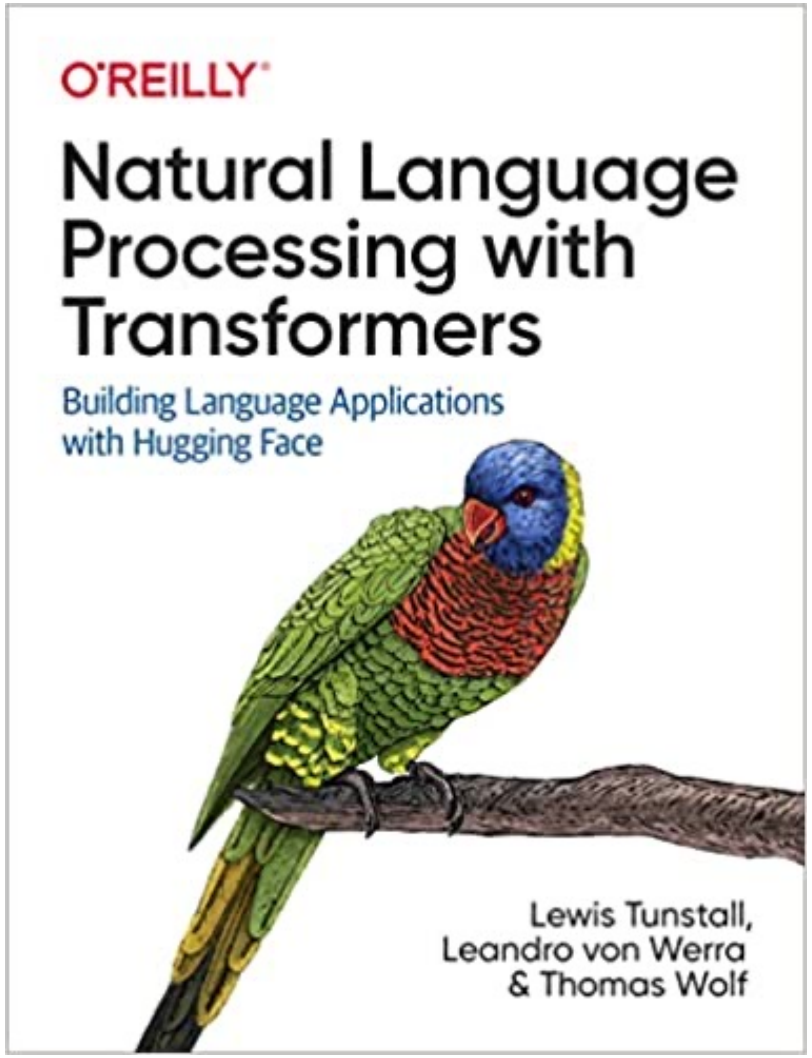




Source: Sowmya Vajjala, Bodhisattwa Majumder, Anuj Gupta (2020), Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems, O'Reilly Media.

Source: https://www.amazon.com/Practical-Natural-Language-Processing-Pragmatic/dp/1492054054

NLP with Transformers Github Notebooks



Running on a cloud platform

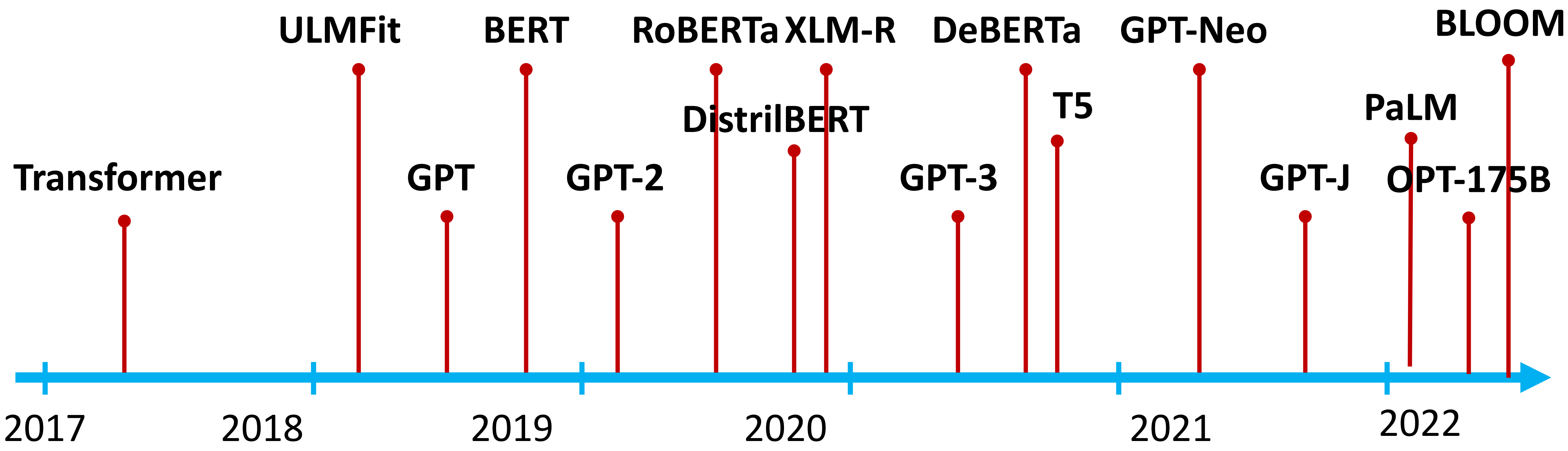
To run these notebooks on a cloud platform, just click on one of the badges in the table below:

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!

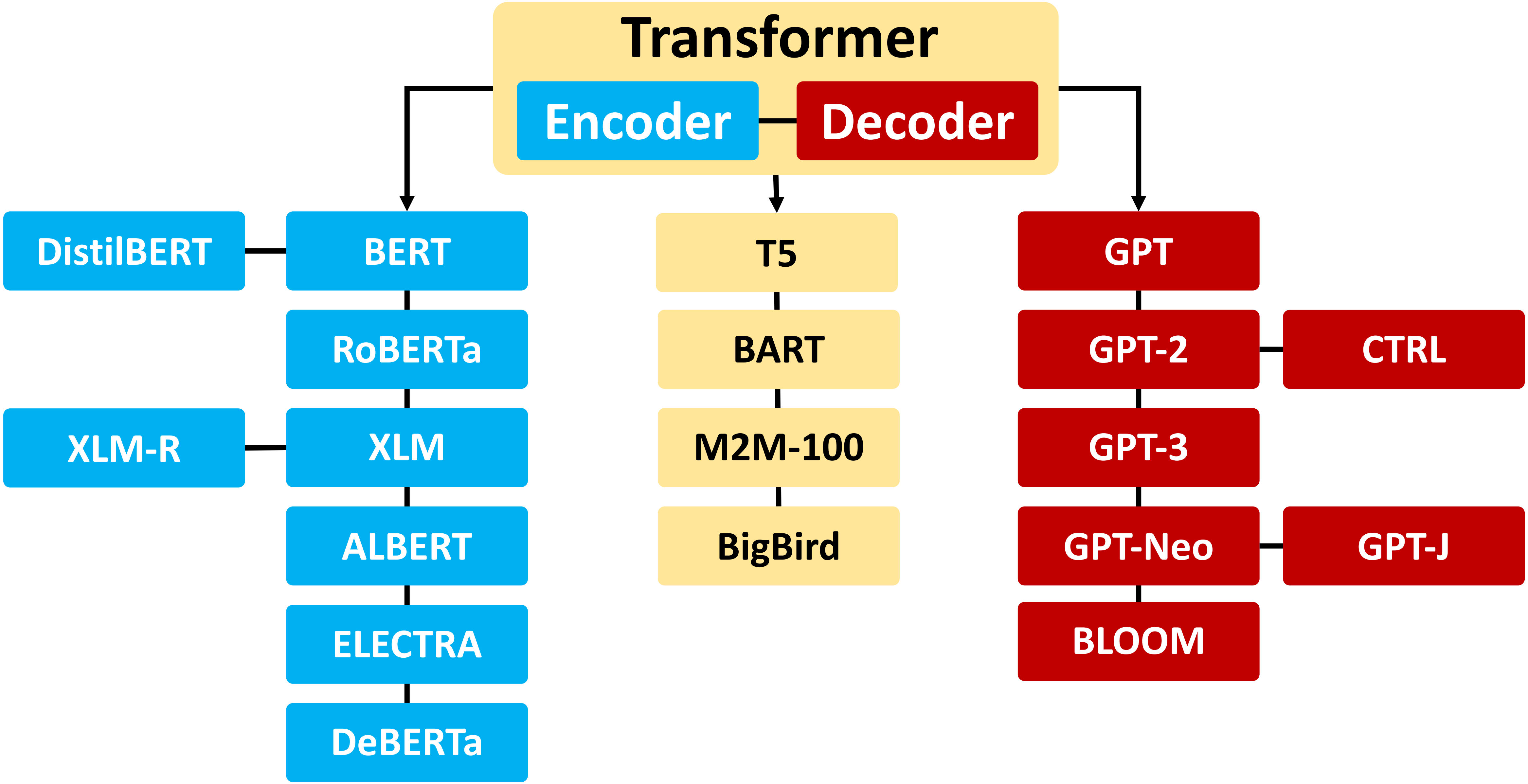
https://github.com/nlp-with-transformers/notebooks

The Transformers Timeline



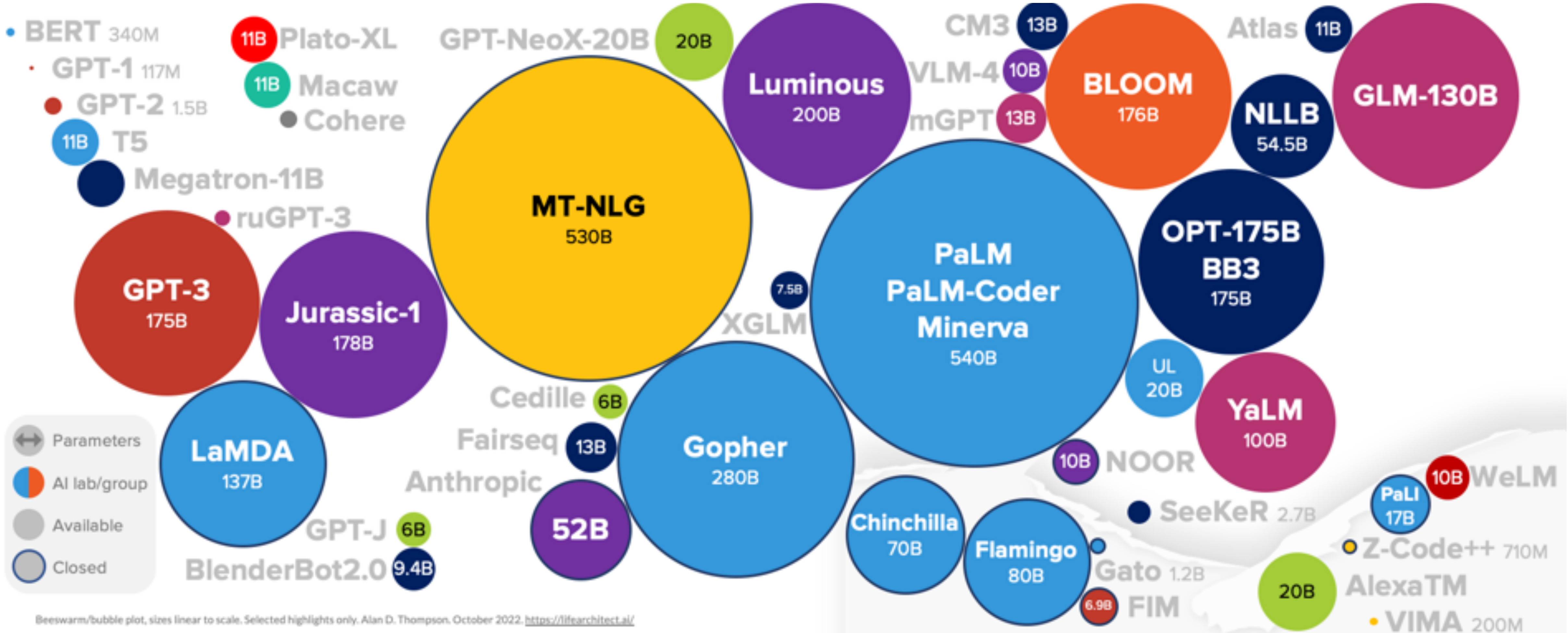
Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

Transformer 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.

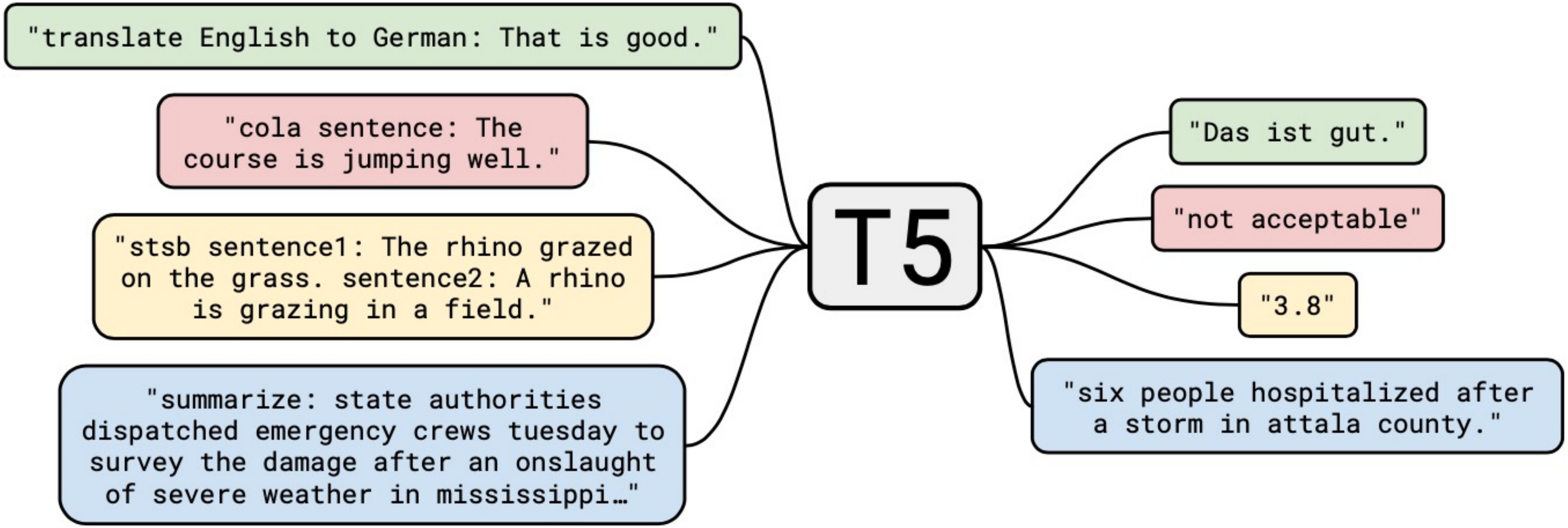
Language Models Sizes
(GPT-3, PaLM, BLOOM)



Source: https://lifearchitect.ai/models/

T5

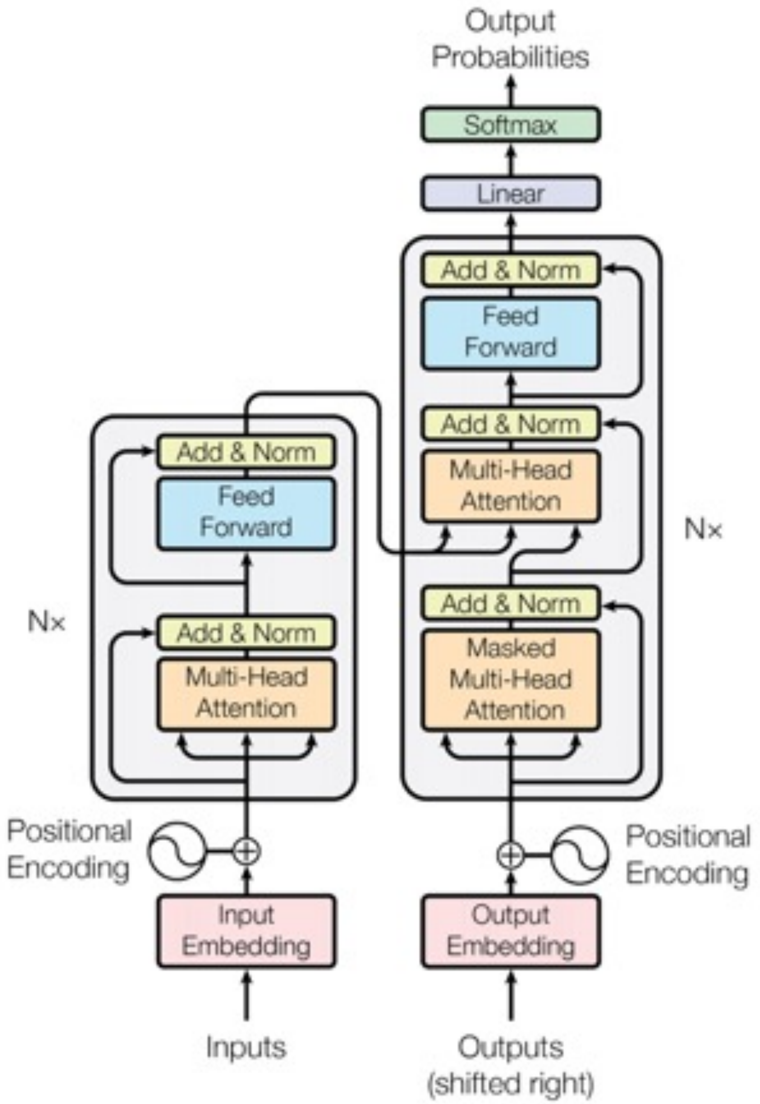
Text-to-Text Transfer Transformer



Source: JColin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu (2019). "Exploring the limits of transfer learning with a unified text-to-text transformer." arXiv preprint arXiv:1910.10683 (2019).

Transformer (Attention is All You Need)

(Vaswani et al., 2017)

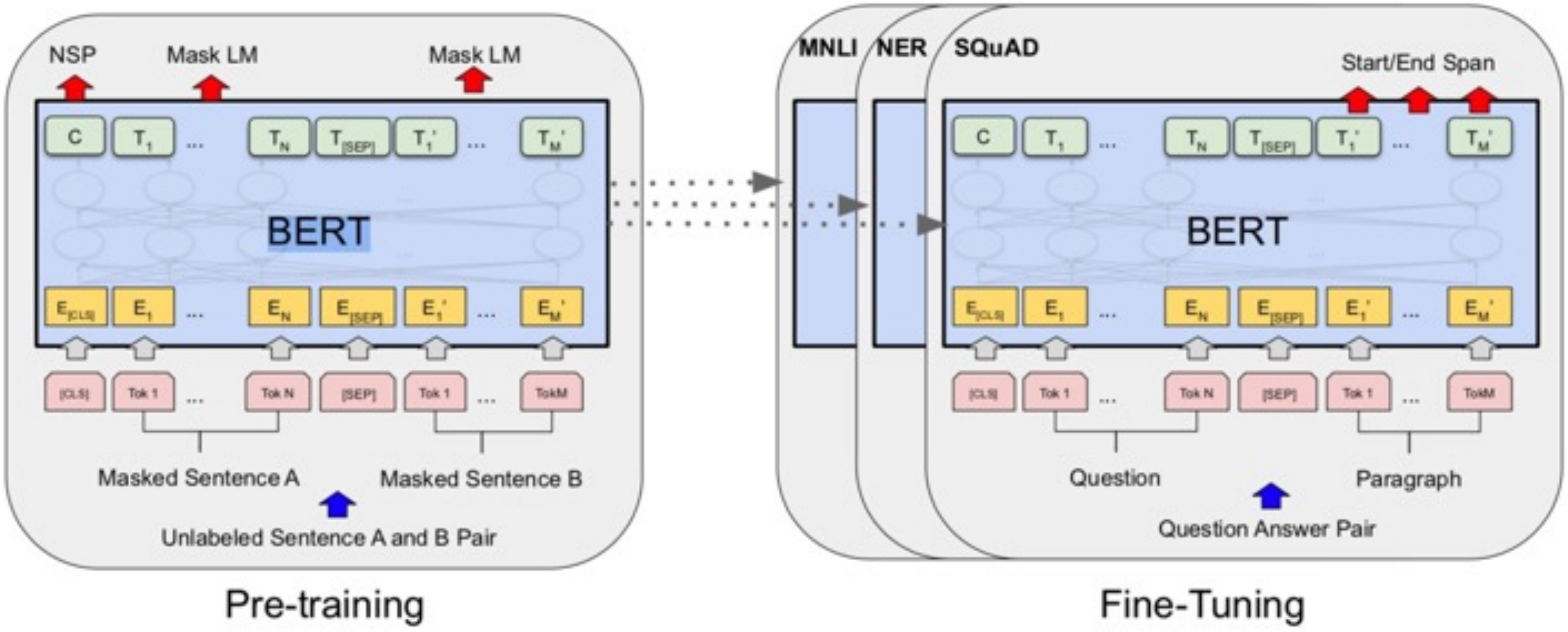


Source: Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In *Advances in neural information processing systems*, pp. 5998-6008. 2017.

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

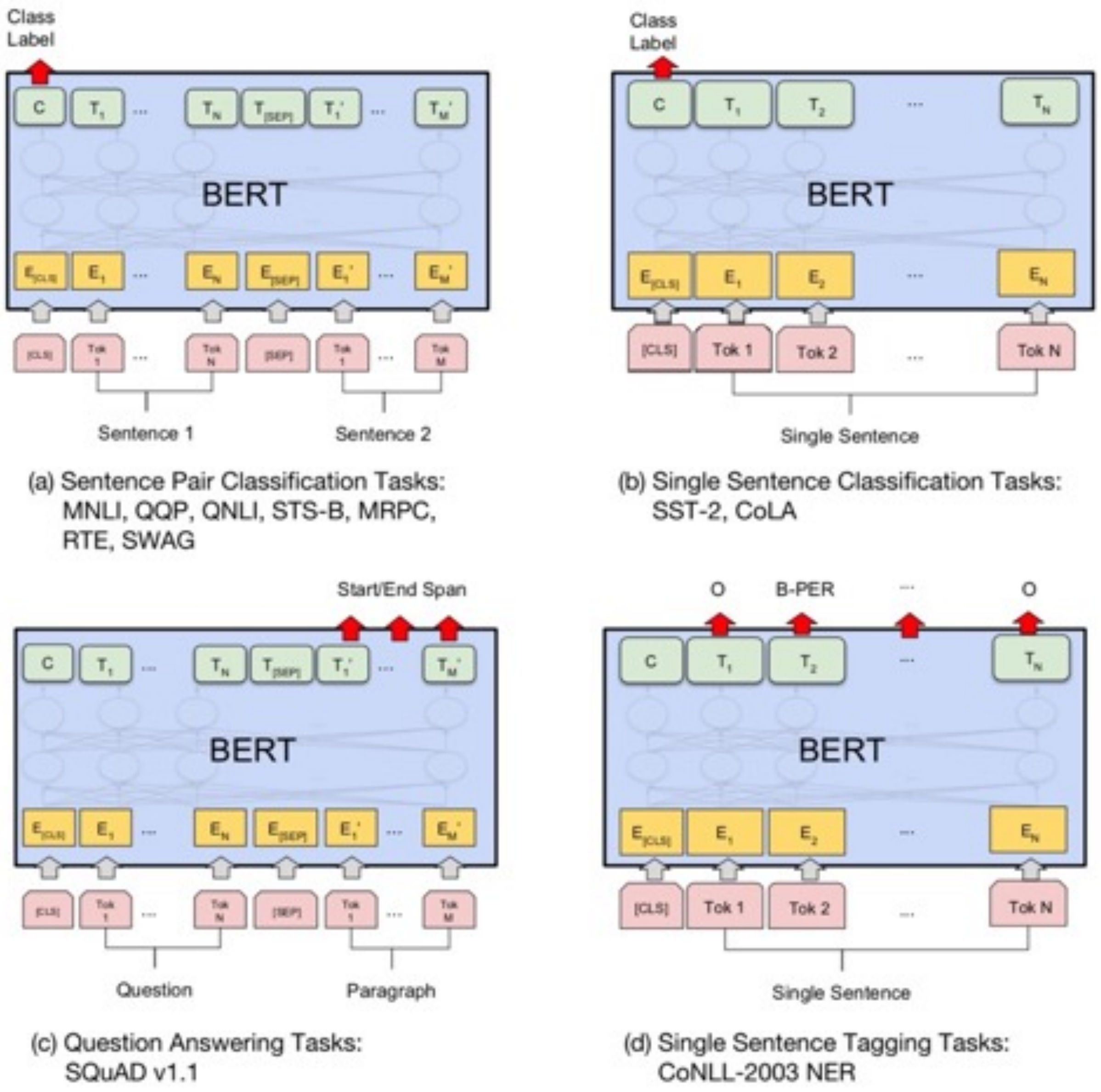
BERT (Bidirectional Encoder Representations from Transformers)

Overall pre-training and fine-tuning procedures for BERT



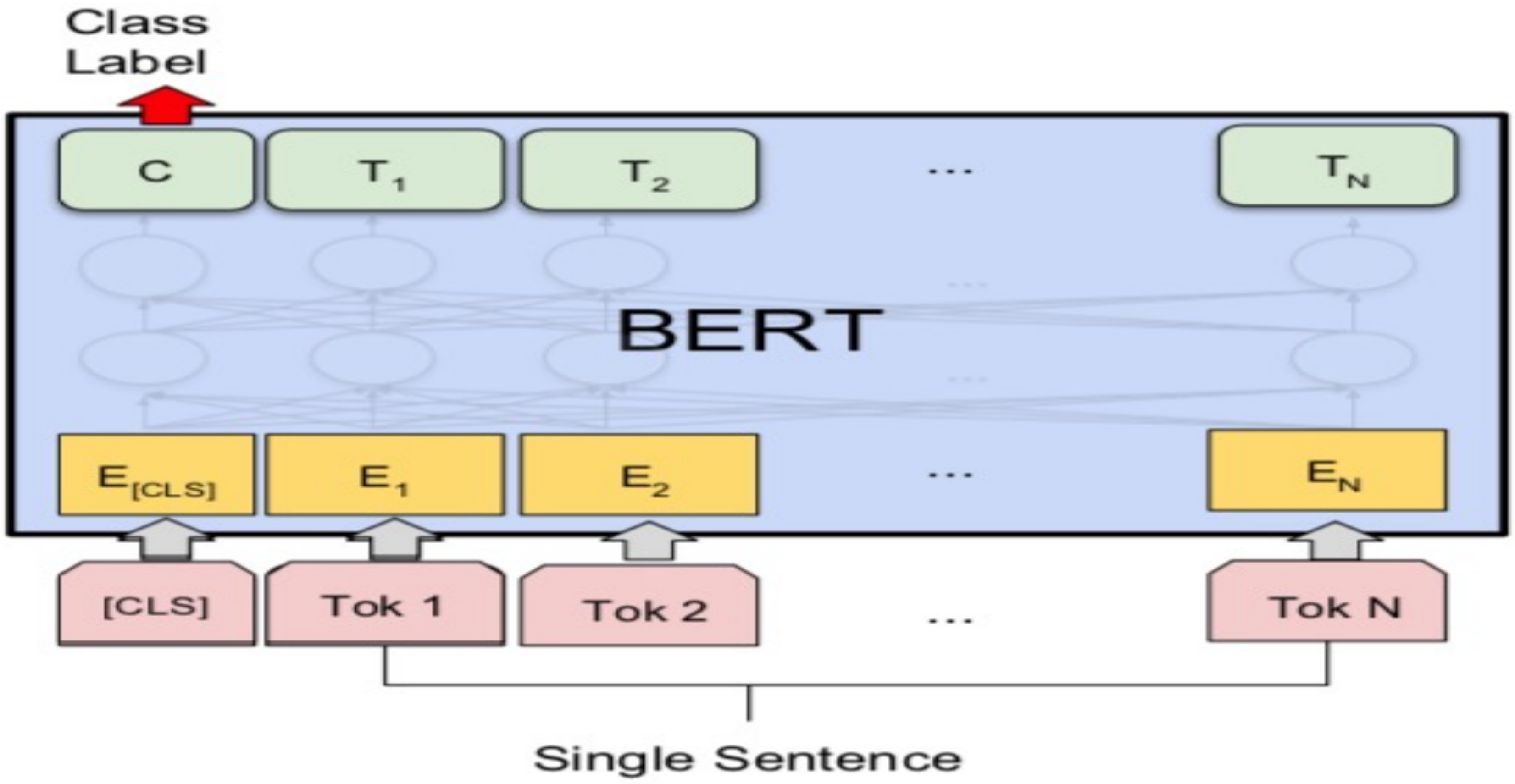
Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

Fine-tuning BERT on Different 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.

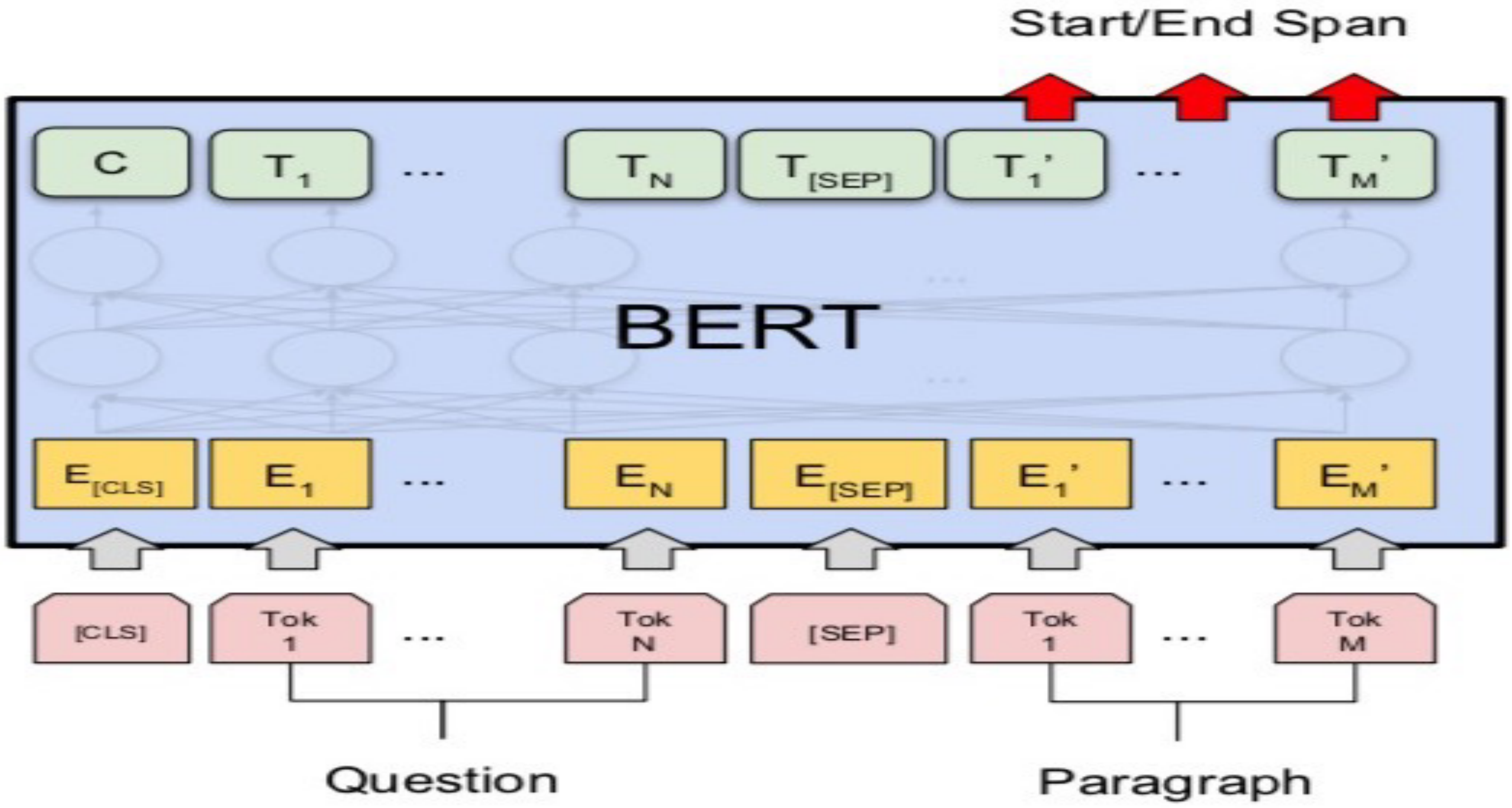
Sentiment Analysis:
Single Sentence Classification



(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

Fine-tuning BERT on Question Answering (QA)

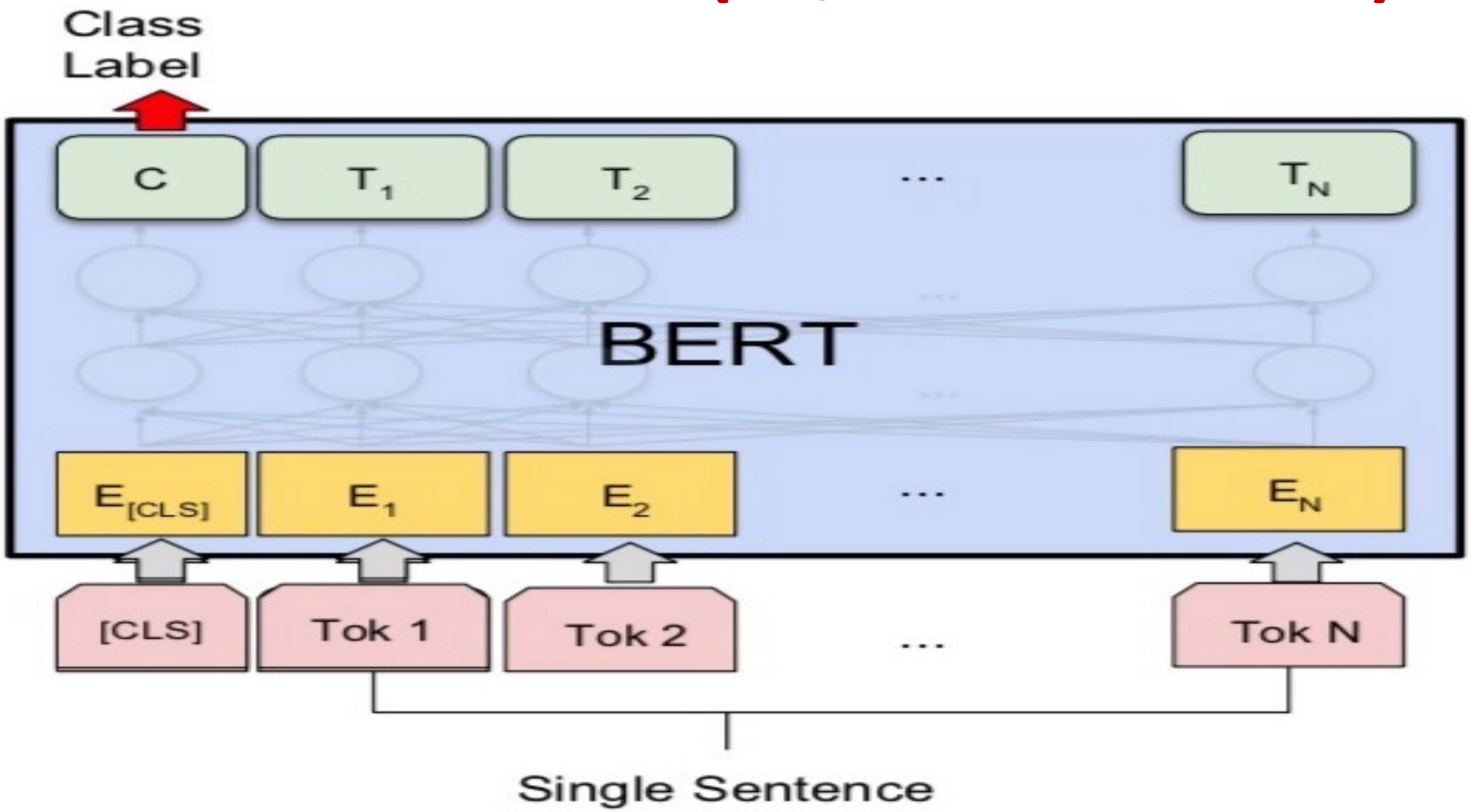


(c) Question Answering Tasks: SQuAD v1.1

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.

Fine-tuning BERT on Dialogue

Intent Detection (ID; Classification)

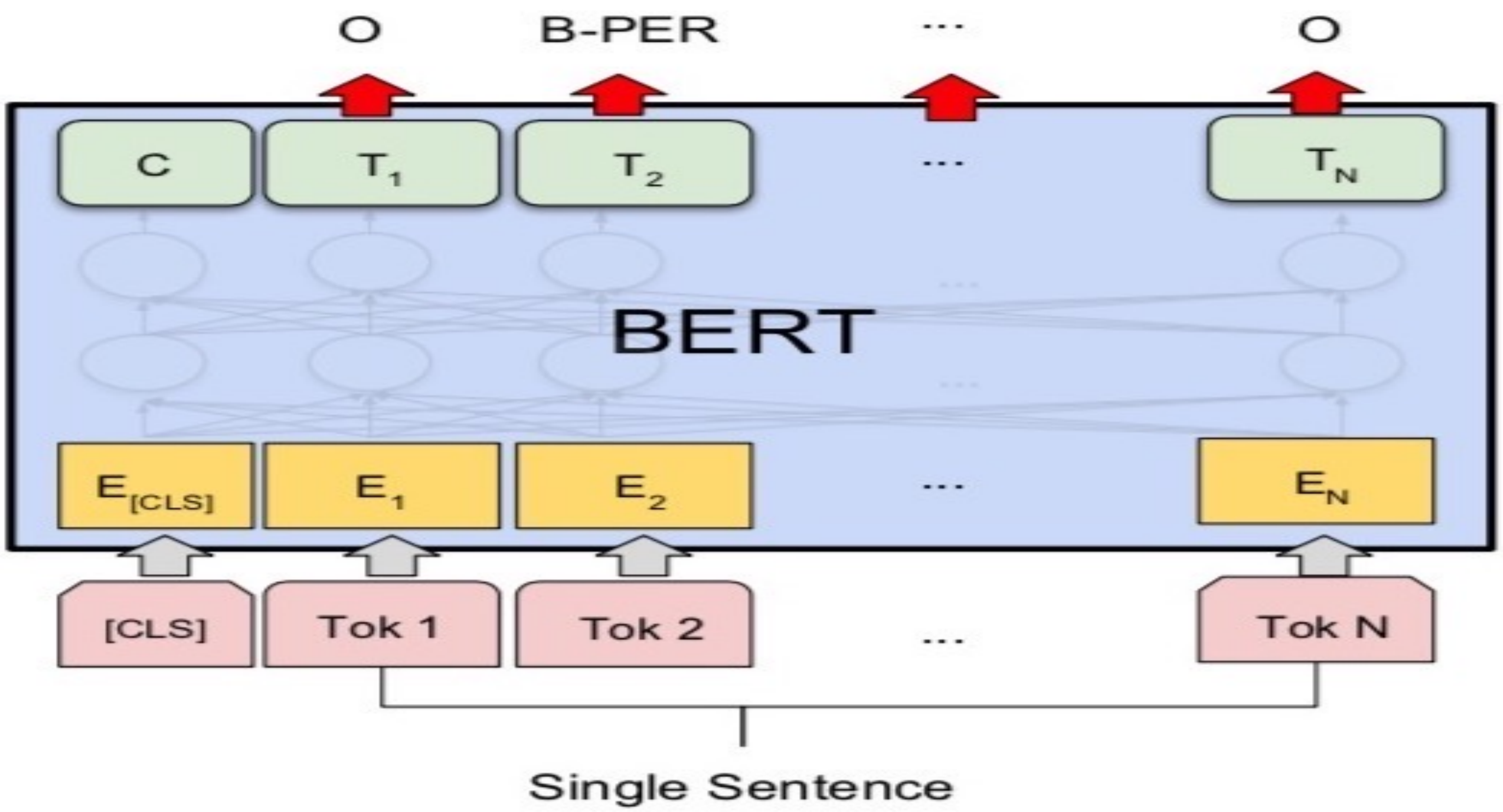


(b) Single Sentence Classification Tasks: SST-2, CoLA

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.

Fine-tuning BERT on Dialogue

Slot Filling (SF)

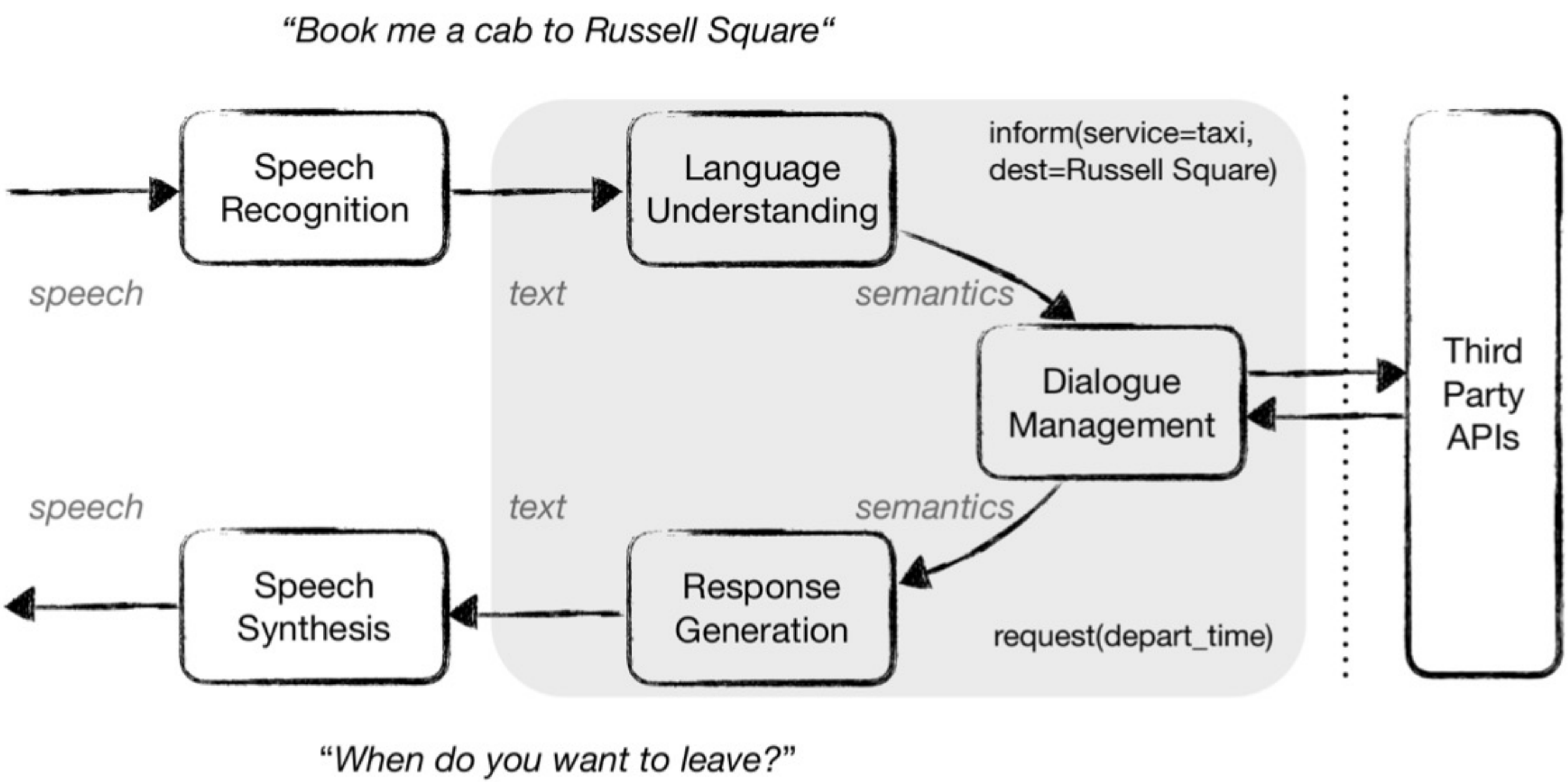


(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

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.

Task-Oriented Dialogue (ToD) System

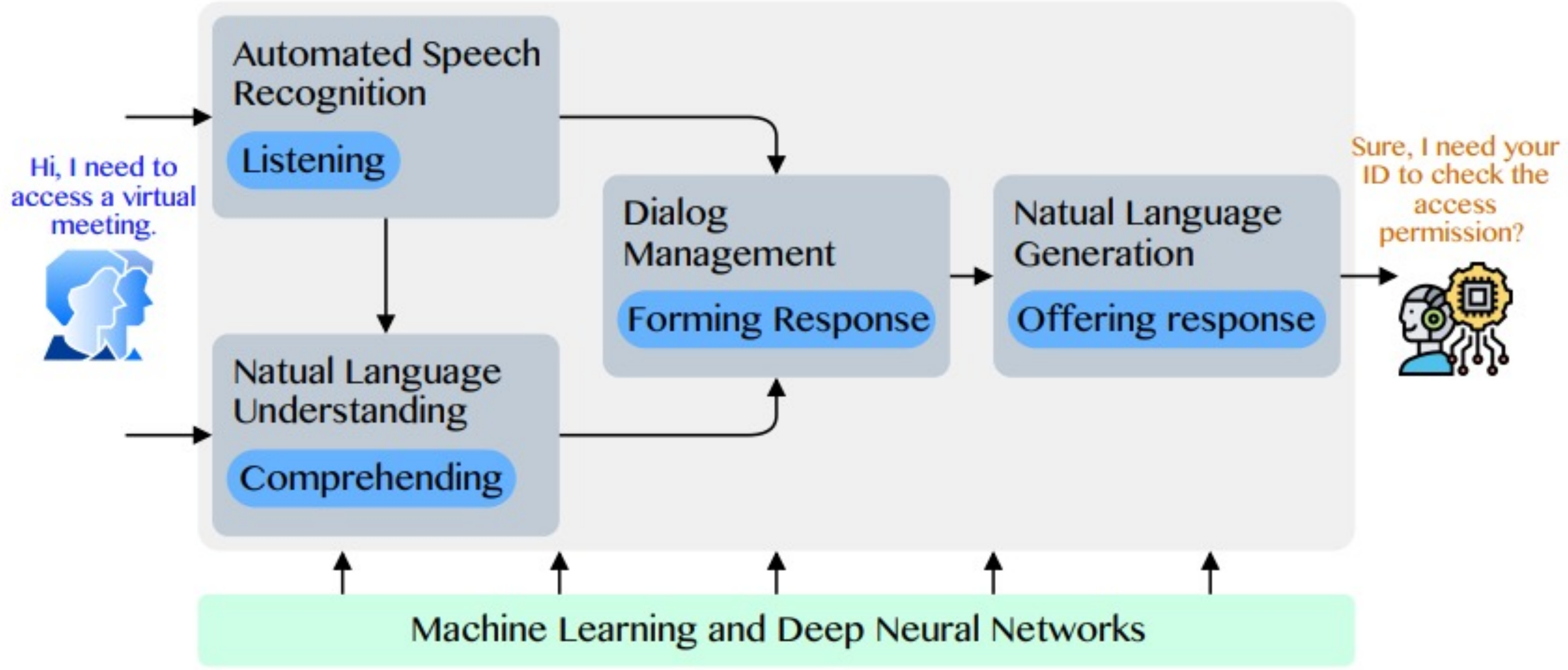
Speech, Text, NLP



Source: Razumovskaia, Evgeniia, Goran Glavas, Olga Majewska, Edoardo M. Ponti, Anna Korhonen, and Ivan Vulic.
"Crossing the conversational chasm: A primer on natural language processing for multilingual task-oriented dialogue systems." Journal of Artificial Intelligence Research 74 (2022): 1351-1402.

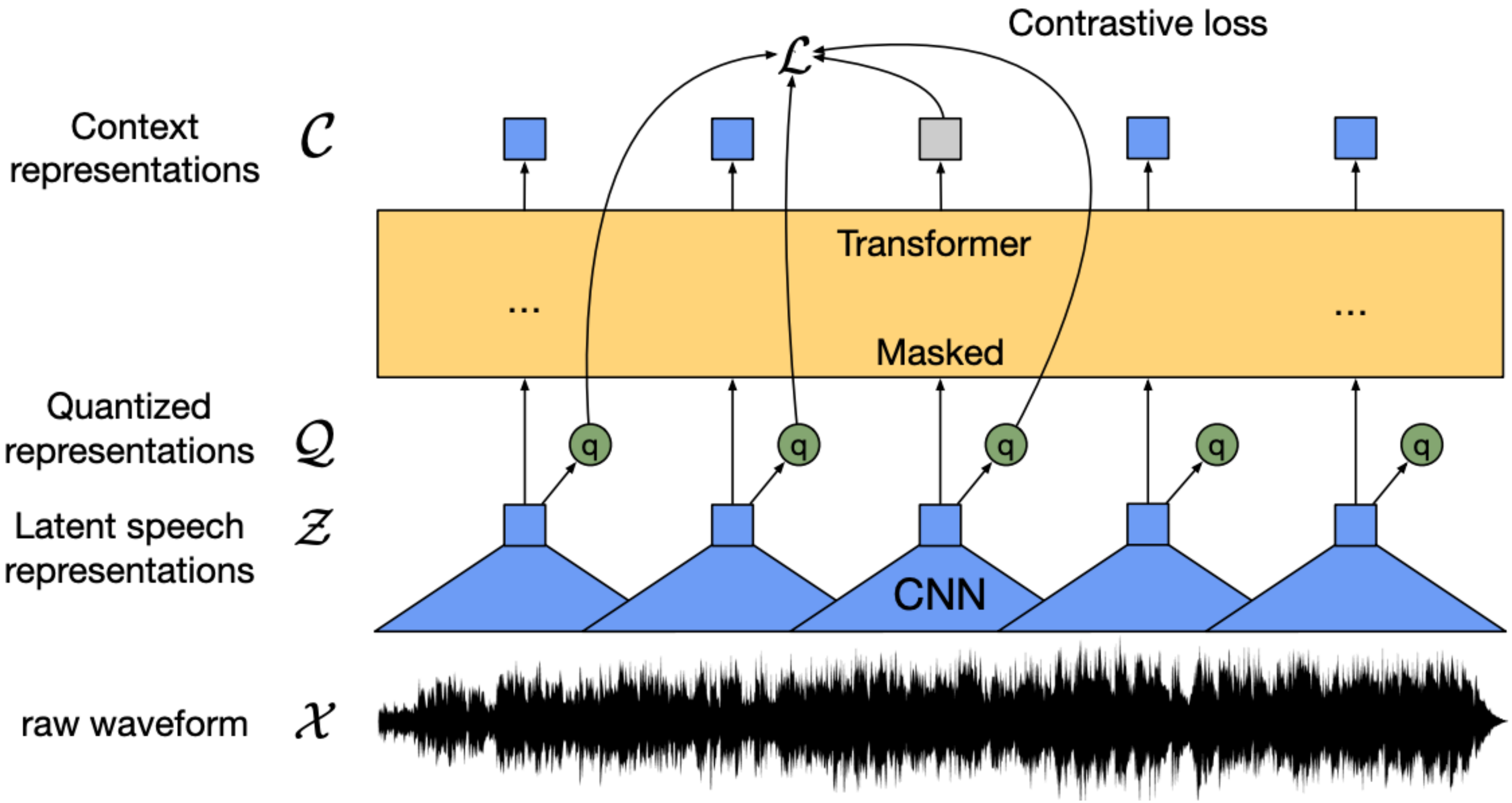
Conversational AI

to deliver contextual and personal experience to users



Source: Huynh-The, Thien, Quoc-Viet Pham, Xuan-Qui Pham, Thanh Thi Nguyen, Zhu Han, and Dong-Seong Kim (2022). "Artificial Intelligence for the Metaverse: A Survey." arXiv preprint arXiv:2202.10336.

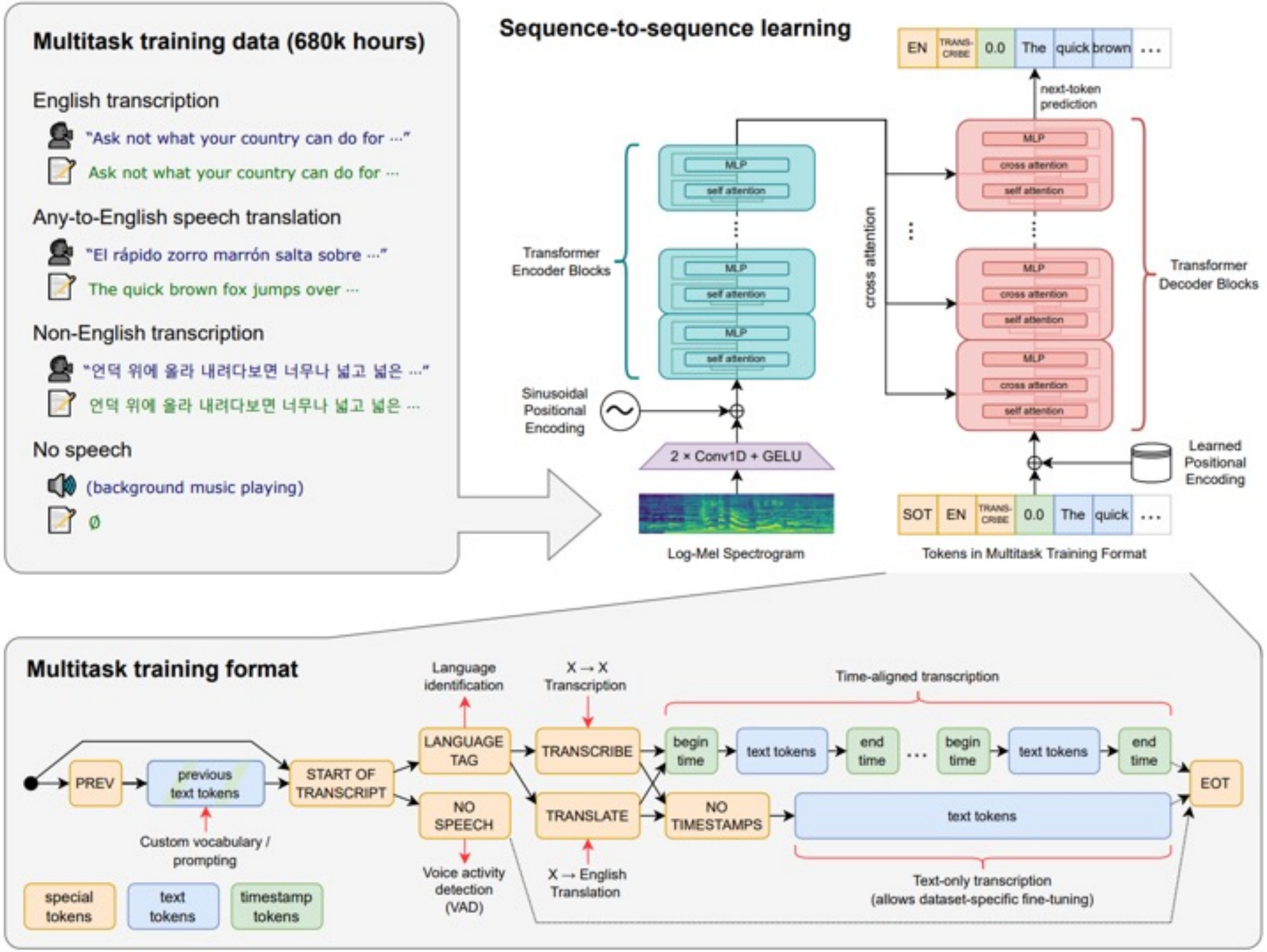
wav2vec 2.0:
A framework for self-supervised learning of speech representations



Source: Baevski, Alexei, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli.
"wav2vec 2.0: A framework for self-supervised learning of speech representations." Advances in Neural Information Processing Systems 33 (2020): 12449-12460.

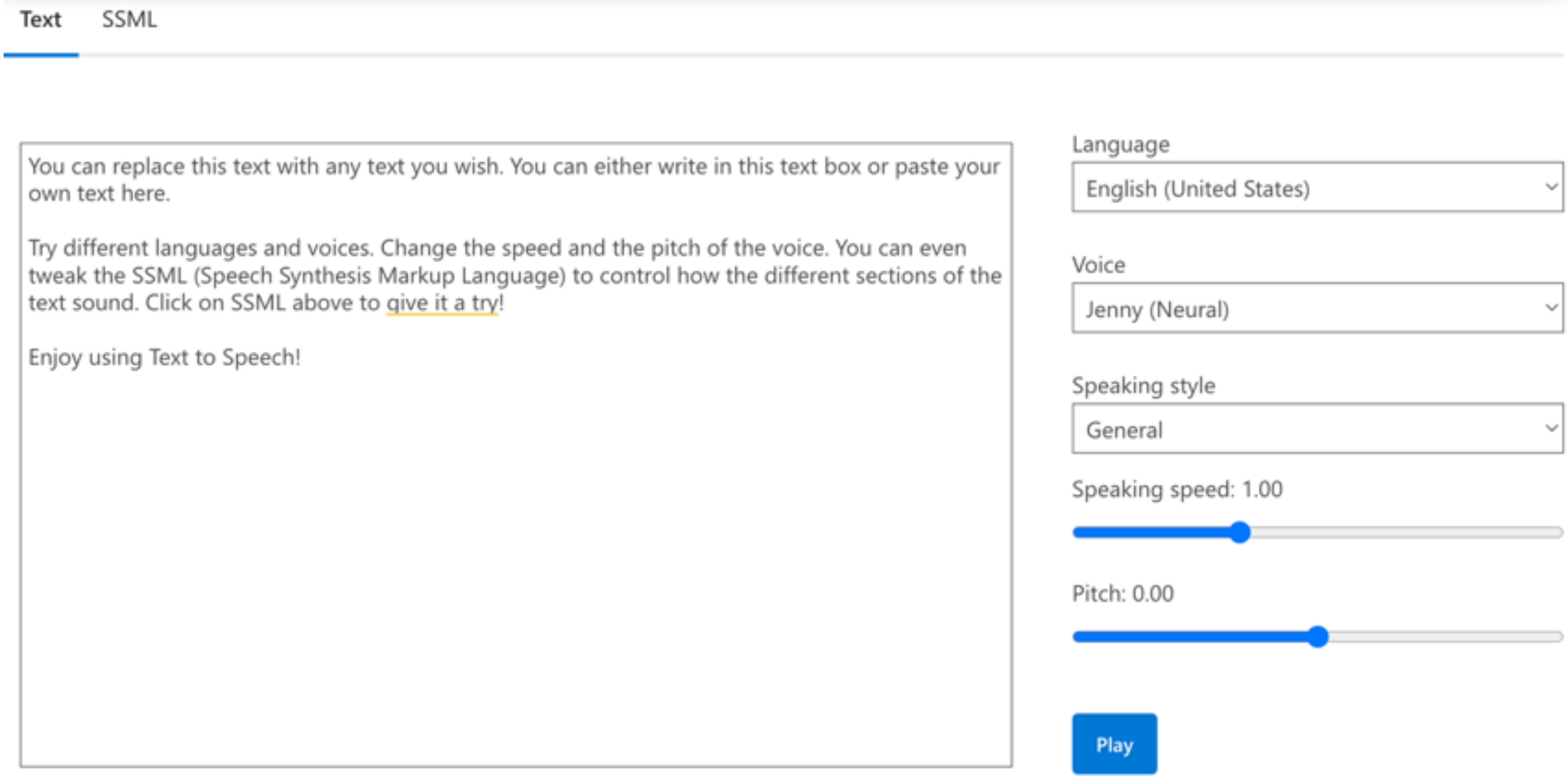
Whisper:

Robust Speech Recognition via Large-Scale Weak Supervision



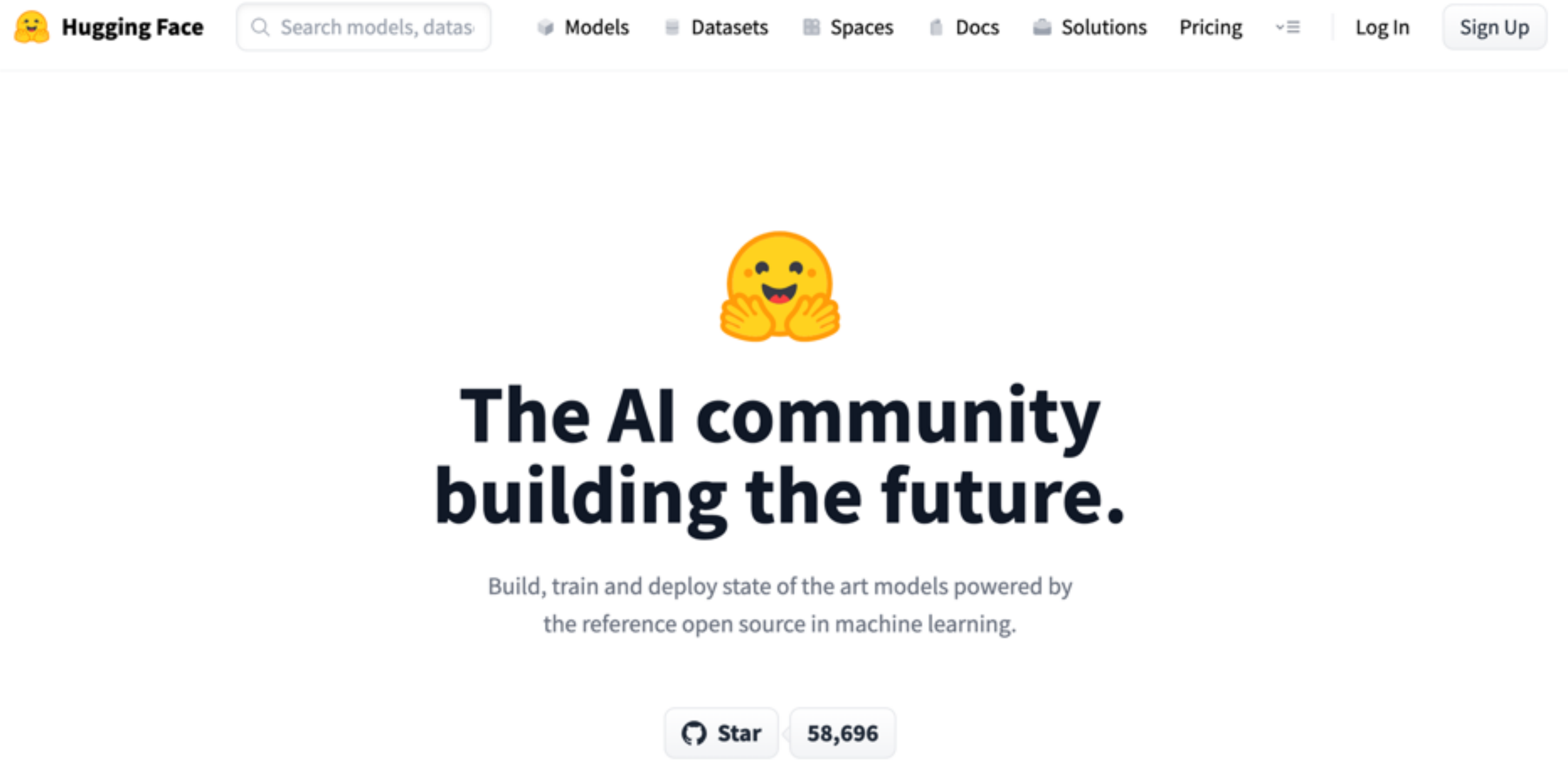
Source: Radford, Alec, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. Tech. Rep., Technical report, OpenAI, 2022.

Microsoft Azure
Text to Speech (TTS)



Source: https://azure.microsoft.com/en-gb/products/cognitive-services/text-to-speech/

Hugging Face



https://huggingface.co/

BLOOM

BigScience Large Open-science Open-access Multilingual Language Model

a BigScience initiative

BLOOM

176B params · 59 languages · Open-access

BigScience Large Open-science Open-access Multilingual Language Model

Version 1.3 / 6 July 2022

Current Checkpoint: **Training Iteration 95000**

Total seen tokens: **366B**

Downloads last month
12,875

Hosted inference API

Text Generation

Groups | Examples

I love bloom. Super simple, but so effective! I went through a similar process a couple of years ago when I

sampling | greedy

BLOOM prompting tips

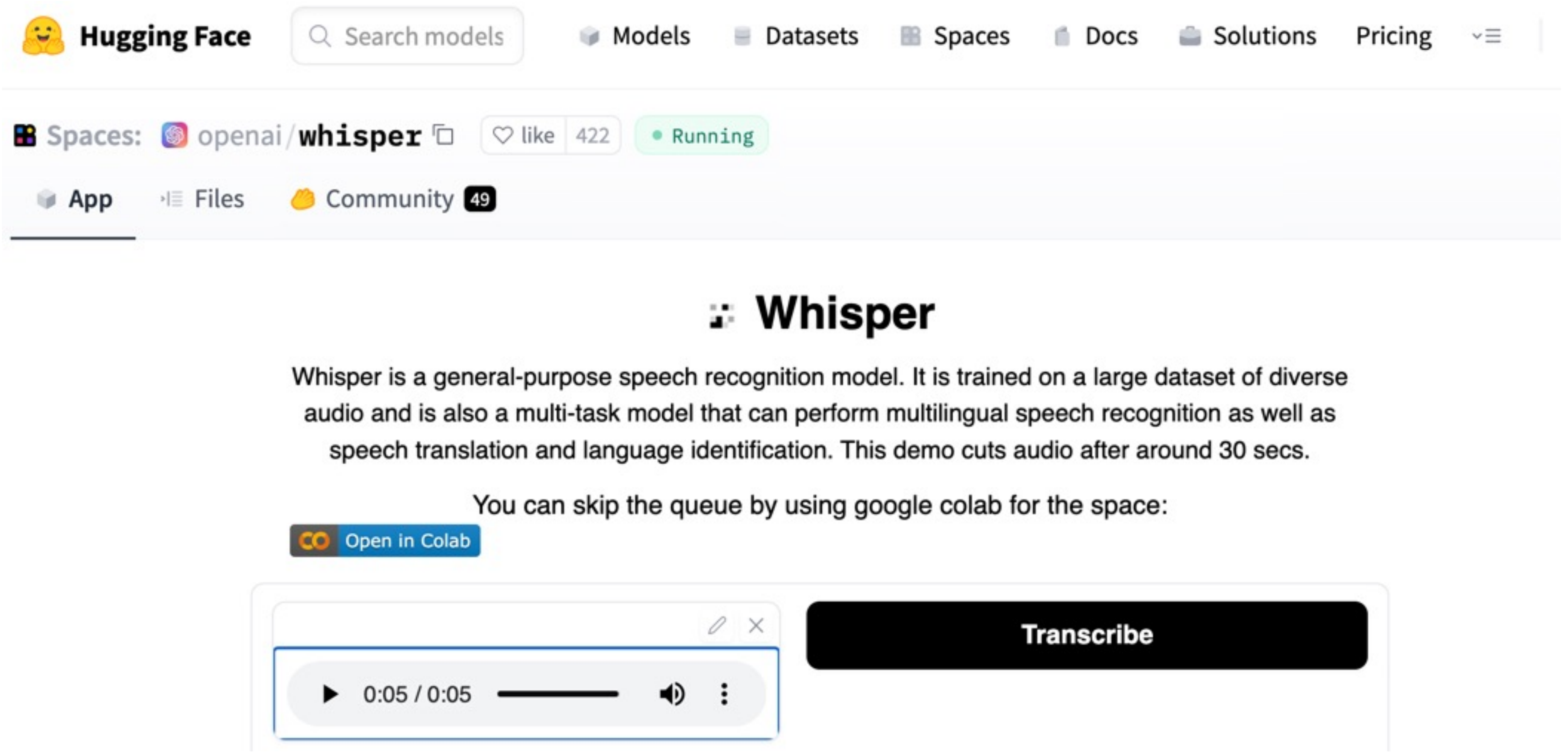
Switch to "greedy" for more accurate completion e.g. math/history/translations (but which may be repetitive/less inventive)

Compute | ⌘+Enter

1.3

Source: https://huggingface.co/bigscience/bloom

OpenAI Whisper



Source: https://huggingface.co/spaces/openai/whisper

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

```
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

Named Entity Recognition

```
ner_tagger = pipeline("ner", aggregation_strategy="simple")
outputs = ner_tagger(text)
pd.DataFrame(outputs)
```

	entity_group	score	word	start	end
0	ORG	0.879010	Amazon	5	11
1	MISC	0.990859	Optimus Prime	36	49
2	LOC	0.999755	Germany	90	97
3	MISC	0.556570	Mega	208	212
4	PER	0.590256	##tron	212	216
5	ORG	0.669692	Decept	253	259
6	MISC	0.498349	##icons	259	264
7	MISC	0.775362	Megatron	350	358
8	MISC	0.987854	Optimus Prime	367	380
9	PER	0.812096	Bumblebee	502	511

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

Question Answering

```
reader = pipeline("question-answering")
question = "What does the customer want?"
outputs = reader(question=question, context=text)
pd.DataFrame([outputs])
```

	score	start	end	answer
0	0.631292	335	358	an exchange of Megatron

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

Summarization

```
summarizer = pipeline("summarization")
outputs = summarizer(text, max_length=45, clean_up_tokenization_spaces=True)
print(outputs[0]['summary_text'])
```

```
Bumblebee 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.
```

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

Translation

```
translator = pipeline("translation_en_to_de",
                      model="Helsinki-NLP/opus-mt-en-de")
outputs = translator(text, clean_up_tokenization_spaces=True, min_length=100)
print(outputs[0]['translation_text'])
```

Sehr geehrter Amazon, letzte Woche habe ich eine Optimus Prime Action Figur aus Ihrem Online-Shop in Deutschland bestellt. Leider, als ich das Paket öffnete, entdeckte ich zu meinem Entsetzen, dass ich stattdessen eine Action Figur von Megatron geschickt worden war! Als lebenslanger Feind der Decepticons, Ich hoffe, Sie können mein Dilemma verstehen. Um das Problem zu lösen, Ich fordere einen Austausch von Megatron für die Optimus Prime Figur habe ich bestellt. Anbei sind Kopien meiner Aufzeichnungen über diesen Kauf. Ich erwarte, bald von Ihnen zu hören. Aufrichtig, 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 Generation

```
from transformers import set_seed
set_seed(42) # Set the seed to get reproducible results
```

```
generator = pipeline("text-generation")
response = "Dear Bumblebee, I am sorry to hear that your order was mixed up."
prompt = text + "\n\nCustomer service response:\n" + response
outputs = generator(prompt, max_length=200)
print(outputs[0]['generated_text'])
```

```
Customer service response:
Dear Bumblebee, I am sorry to hear that your order was mixed up. The
order was completely mislabeled, which is very common in our online
store, but I can appreciate it because it was my understanding from this
site and our customer service of the previous day that your order was
not made correct in our mind and that we are in a process of resolving
this matter. We can assure you that your order
```

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 Generation

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.

Customer service response:
Dear Bumblebee, I am sorry to hear that your order was mixed up. The order was completely mislabeled, which is very common in our online store, but I can appreciate it because it was my understanding from this site and our customer service of the previous day that your order was not made correct in our mind and that we are in a process of resolving this matter. We can assure you that your order

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

Named Entity Recognition (NER)

```
from transformers import pipeline
import pandas as pd
classifier = pipeline("ner")
text = "My name is Michael and I live in Berkeley, California."
outputs = classifier(text)
pd.DataFrame(outputs)
```

	entity	score	index	word	start	end
0	I-PER	0.998874	4	Michael	11	18
1	I-LOC	0.997050	9	Berkeley	33	41
2	I-LOC	0.999170	11	California	43	53

https://tinyurl.com/aintpupython101

Question Answering

```
!pip install transformers
from transformers import pipeline
qamodel = pipeline("question-answering")
question = "Where do I live?"
context = "My name is Michael and I live in Taipei."
qamodel(question = question, context = context)
```

```
{'answer': 'Taipei', 'end': 39, 'score': 0.9730741381645203, 'start': 33}
```

https://tinyurl.com/aintpupython101

Question Answering

```
from transformers import pipeline
qamodel = pipeline("question-answering", model ='deepset/roberta-base-squad2')
question = "Where do I live?"
context = "My name is Michael and I live in Taipei."
output = qamodel(question = question, context = context)
print(output['answer’])
```

```
Taipei
```

Question Answering

```
from transformers import pipeline
qamodel = pipeline("question-answering", model ='deepset/roberta-base-squad2')
question = "What causes precipitation to fall?"
context = """In meteorology, precipitation is any product of
the condensation of atmospheric water vapor that falls under
gravity. The main forms of precipitation include drizzle,
rain, sleet, snow, graupel and hail... Precipitation forms as
smaller droplets coalesce via collision with other rain drops
or ice crystals within a cloud. Short, intense periods of
rain in scattered locations are called "showers"."""
output = qamodel(question = question, context = context)
print(output['answer'])
```

```
gravity
```

https://tinyurl.com/aintpupython101

Text Generation

```
!pip install transformers
from transformers import pipeline
generator = pipeline('text-generation', model = 'gpt2')
generator("Hello, I'm a language model", max_length = 30, num_return_sequences=3)
```

```
[{'generated_text': "Hello, I'm a language model. It's like looking at it,
where is each word of the sentence? That's what I mean. Like"},
{'generated_text': "Hello, I'm a language modeler. I'm using this for two
purposes: I'm having a lot fewer bugs and faster performance. If I"},
{'generated_text': 'Hello, I\'m a language model, and I was born to
code."\n\nNow, I am thinking about this from a different perspective with
a'}]
```

https://tinyurl.com/aintpupython101

Text Generation

```
from transformers import pipeline
generator = pipeline('text-generation', model = 'gpt2')
outputs = generator("Once upon a time", max_length = 30)
print(outputs[0]['generated_text'])
```

```
Once upon a time, every person who ever saw
Jesus, knew that He was Christ. And even
though he might not have known Him, He was
```

Text Generation

```
from transformers import pipeline
generator = pipeline('text-generation', model = 'gpt2')
outputs = generator("Once upon a time", max_length = 100)
print(outputs[0]['generated_text'])
```

```
Once upon a time we should be able to speak to people who
have lost children, so we try to take those that have lost
the children to our institutions — but the first time is very
hard for us because of our institutions. To me, it's
important to acknowledge that in an institution of faith and
love they are not children. And that there are many people
who are still hurting the child and there are many in need of
help, if not a system. So I'm very curious
```

https://tinyurl.com/aintpupython101

Text2Text Generation

```
from transformers import pipeline
text2text_generator = pipeline("text2text-generation", model = 't5-base')
outputs = text2text_generator("translate from English to French: I am a student")
print(outputs[0]['generated_text'])
```

I am a student

Je suis un étudiant

https://tinyurl.com/aintpupython101

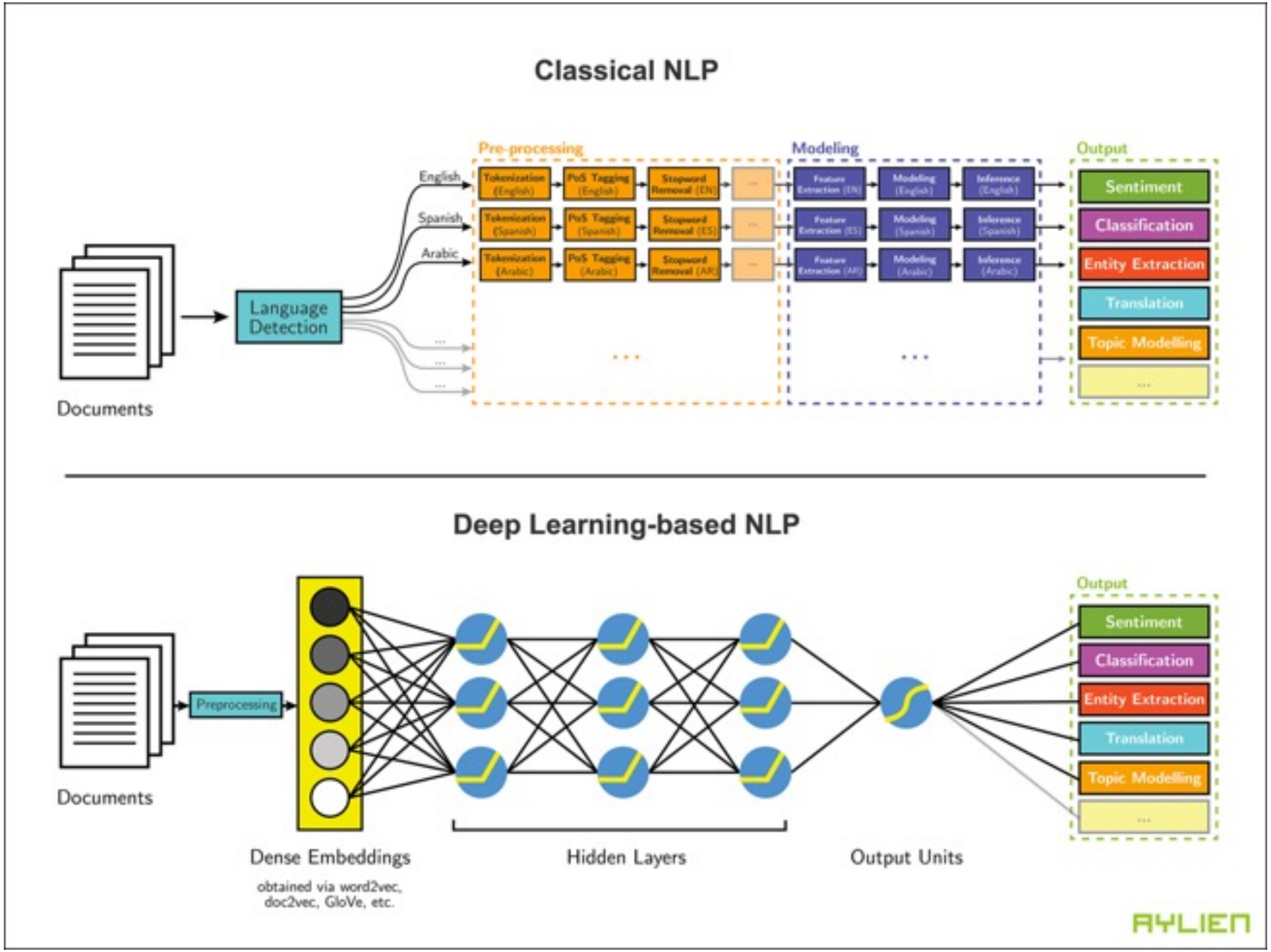
Text2Text Generation

```
from transformers import pipeline
text2text_generator = pipeline("text2text-generation")
text2text_generator("question: What is 42 ? context: 42 is the answer to life, the universe and everything")
```

```
[{'generated_text': 'the answer to life, the universe and everything'}]
```

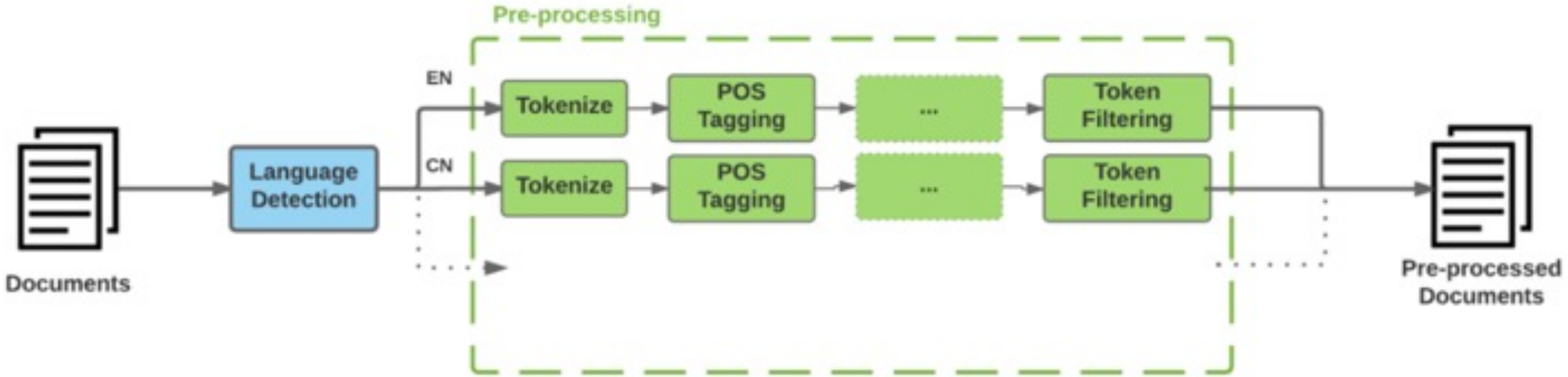
https://tinyurl.com/aintpupython101

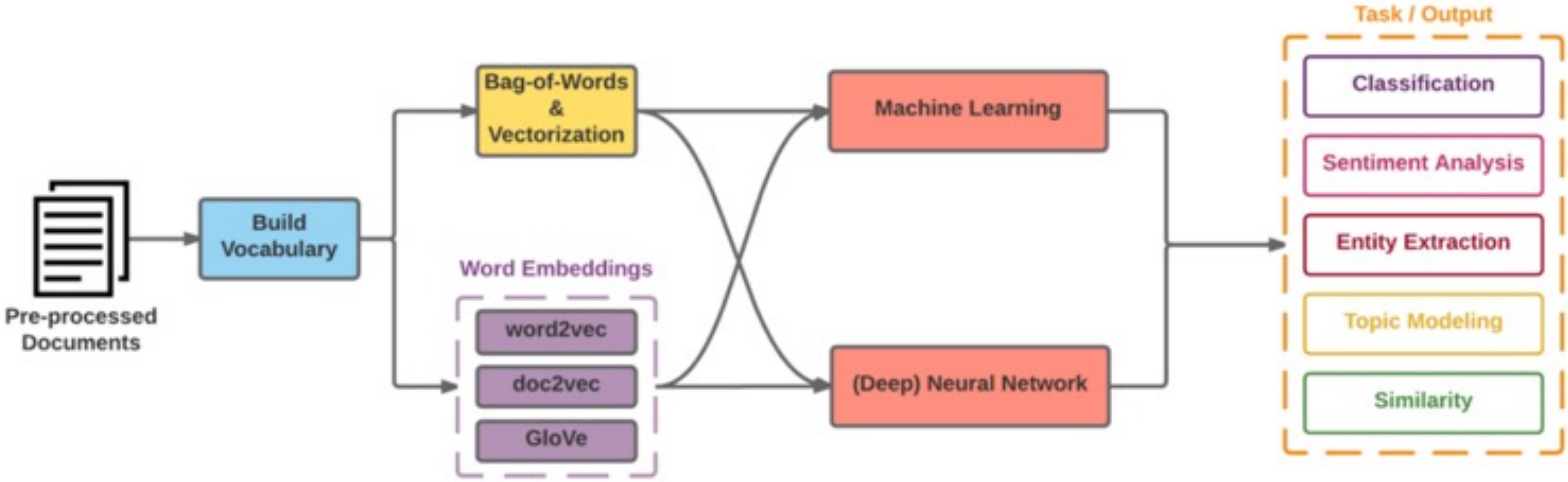
NLP



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

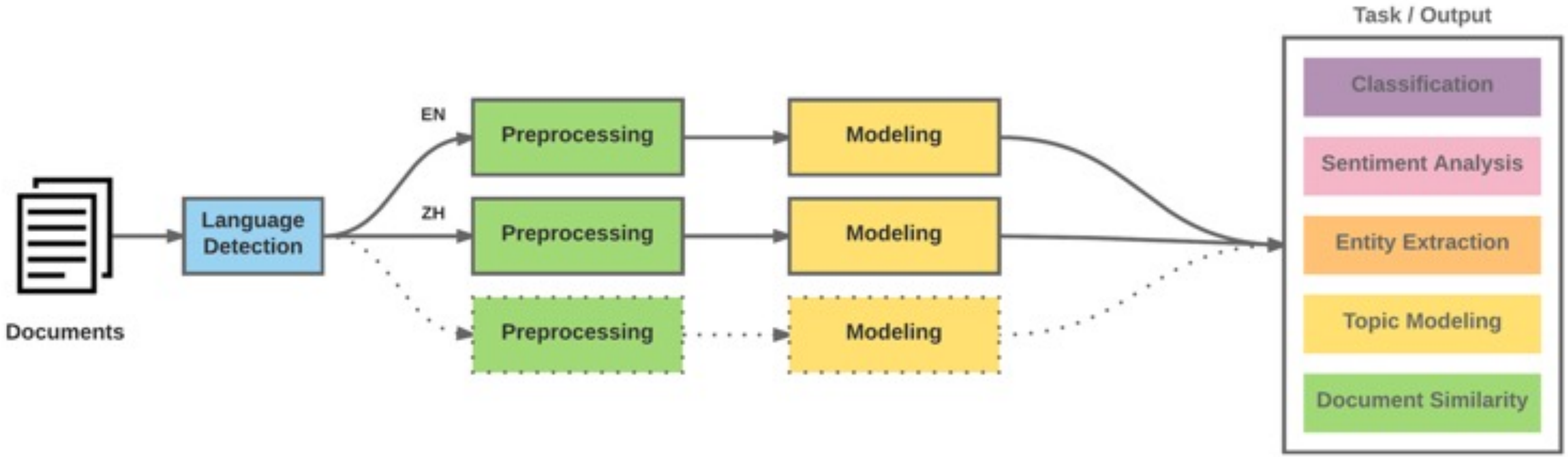
Modern NLP Pipeline





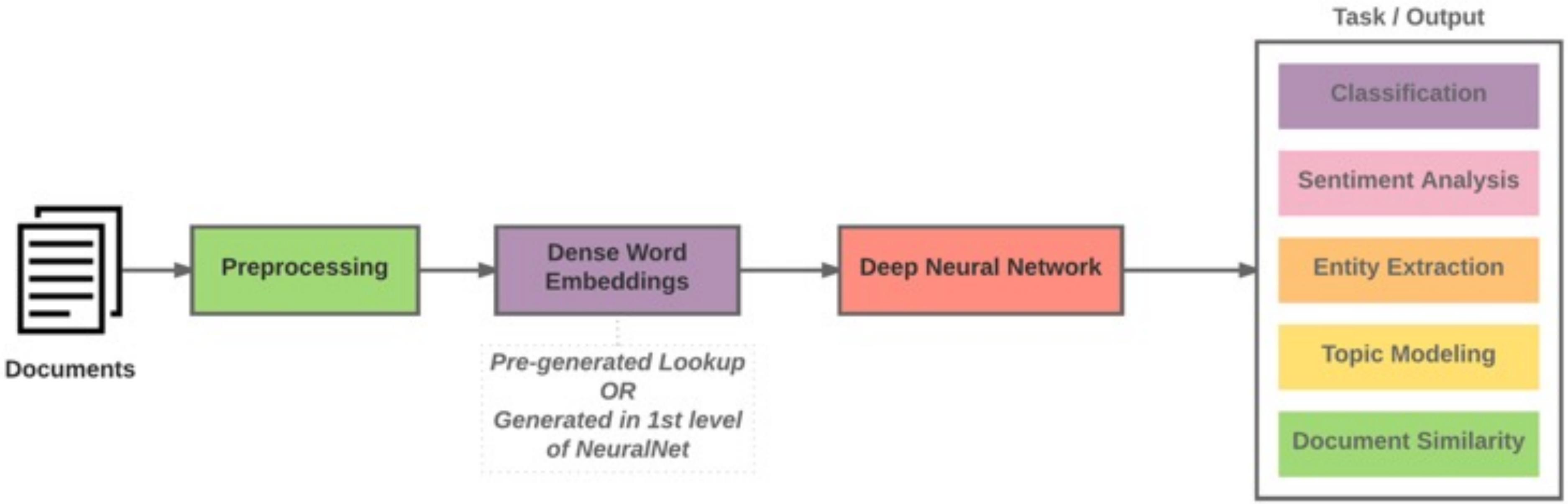
Source: https://github.com/fortiema/talks/blob/master/opendata2016sh/pragmatic-nlp-opendata2016sh.pdf

Modern NLP Pipeline



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

Deep Learning NLP



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

Natural Language Processing (NLP) and Text Mining

- Raw text
- Sentence Segmentation
- Tokenization
- Part-of-Speech (POS)
- Stop word removal
- Stemming / Lemmatization
- Dependency Parser
- String Metrics & Matching

word's stem
am → am
having → hav

word's lemma
am → be
having → have

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

Outline

- Word Embeddings
- Recurrent Neural Networks for NLP
- Sequence-to-Sequence Models
- The Transformer Architecture
- Pretraining and Transfer Learning
- State of the art (SOTA)

One-hot encoding

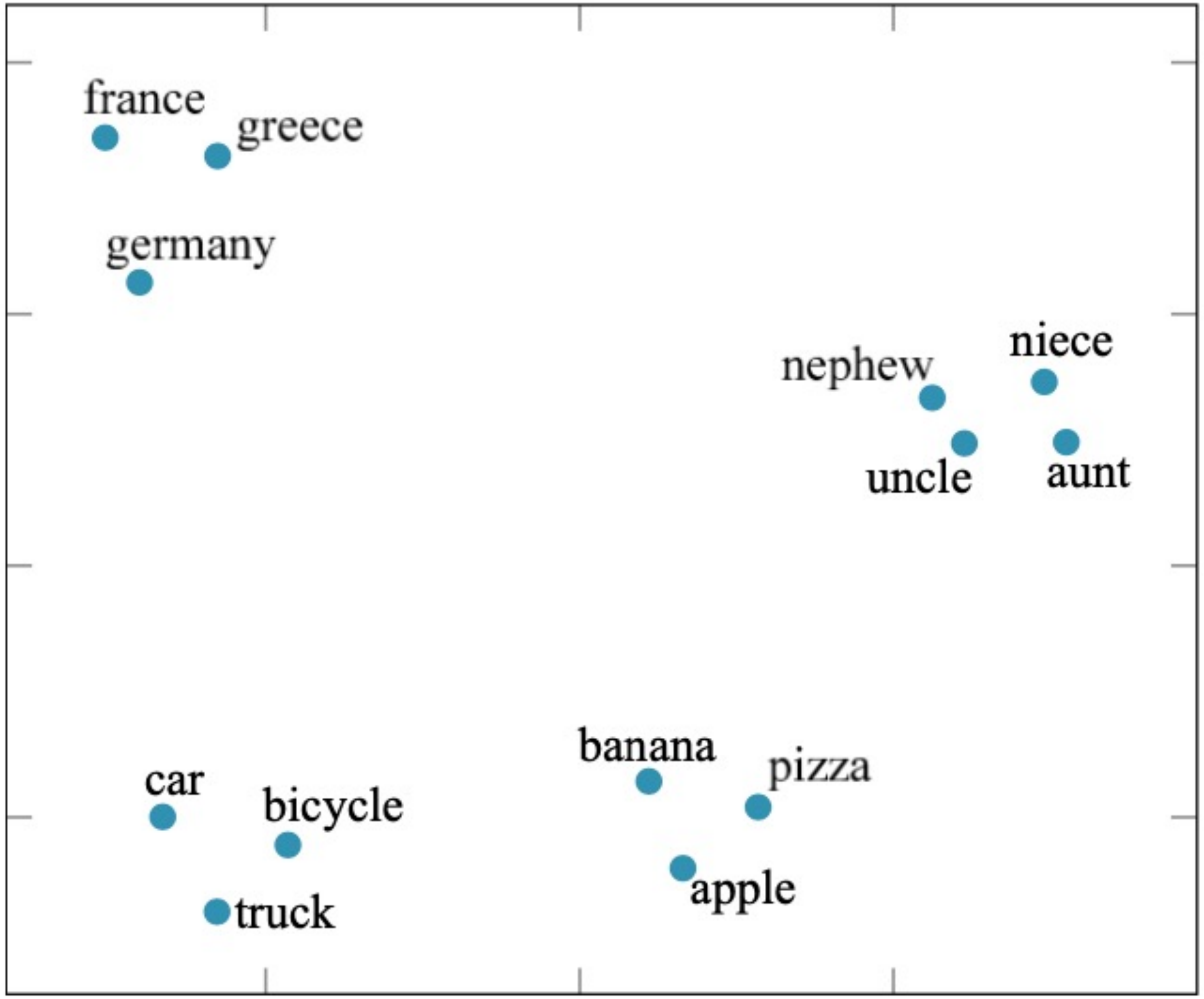
```
'The mouse ran up the clock’ =

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

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

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

Word embedding

GloVe (trained on 6 billion words of text)

100-dimensional word vectors are projected down onto two dimensions



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

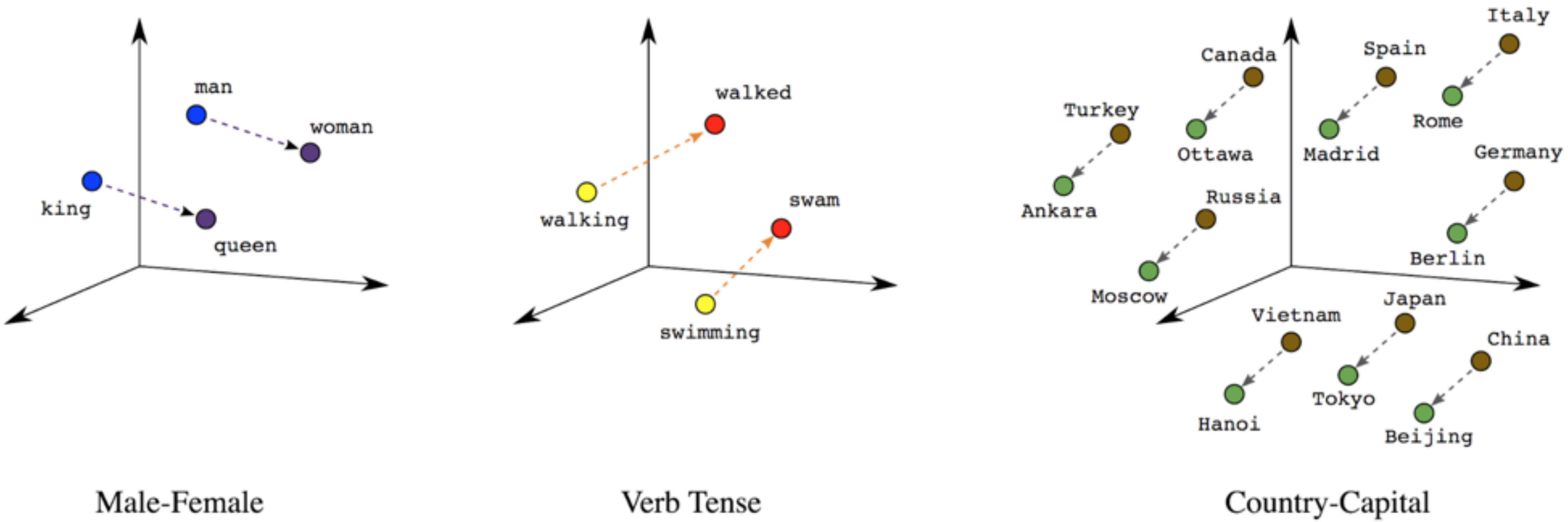
Word Embedding model

answer the question "A is to B as C is to [what]?"

A	**B**	**C**	$\mathbf{D = C + (B - A)}$	**Relationship**
Athens	Greece	Oslo	Norway	*Capital*
Astana	Kazakhstan	Harare	Zimbabwe	*Capital*
Angola	kwanza	Iran	rial	*Currency*
copper	Cu	gold	Au	*Atomic Symbol*
Microsoft	Windows	Google	Android	*Operating System*
New York	New York Times	Baltimore	Baltimore Sun	*Newspaper*
Berlusconi	Silvio	Obama	Barack	*First name*
Switzerland	Swiss	Cambodia	Cambodian	*Nationality*
Einstein	scientist	Picasso	painter	*Occupation*
brother	sister	grandson	granddaughter	*Family Relation*
Chicago	Illinois	Stockton	California	*State*
possibly	impossibly	ethical	unethical	*Negative*
mouse	mice	dollar	dollars	*Plural*
easy	easiest	lucky	luckiest	*Superlative*
walking	walked	swimming	swam	*Past tense*

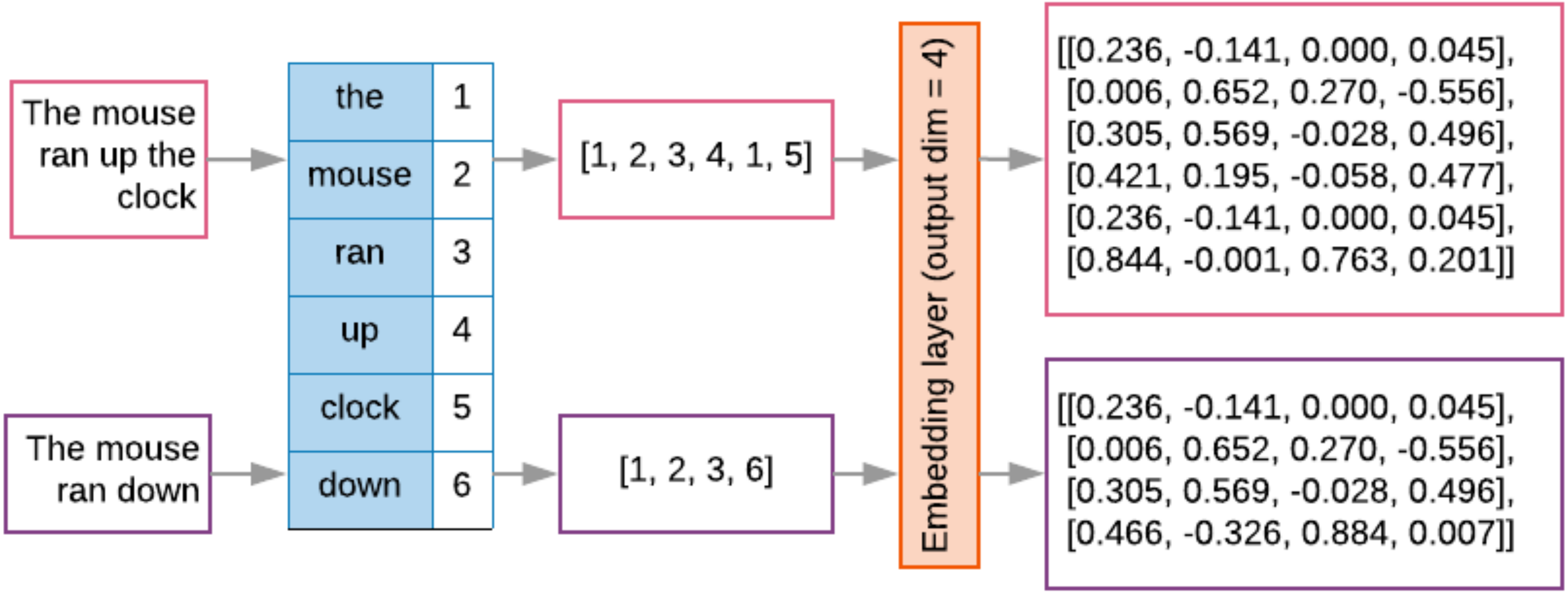
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Word embeddings



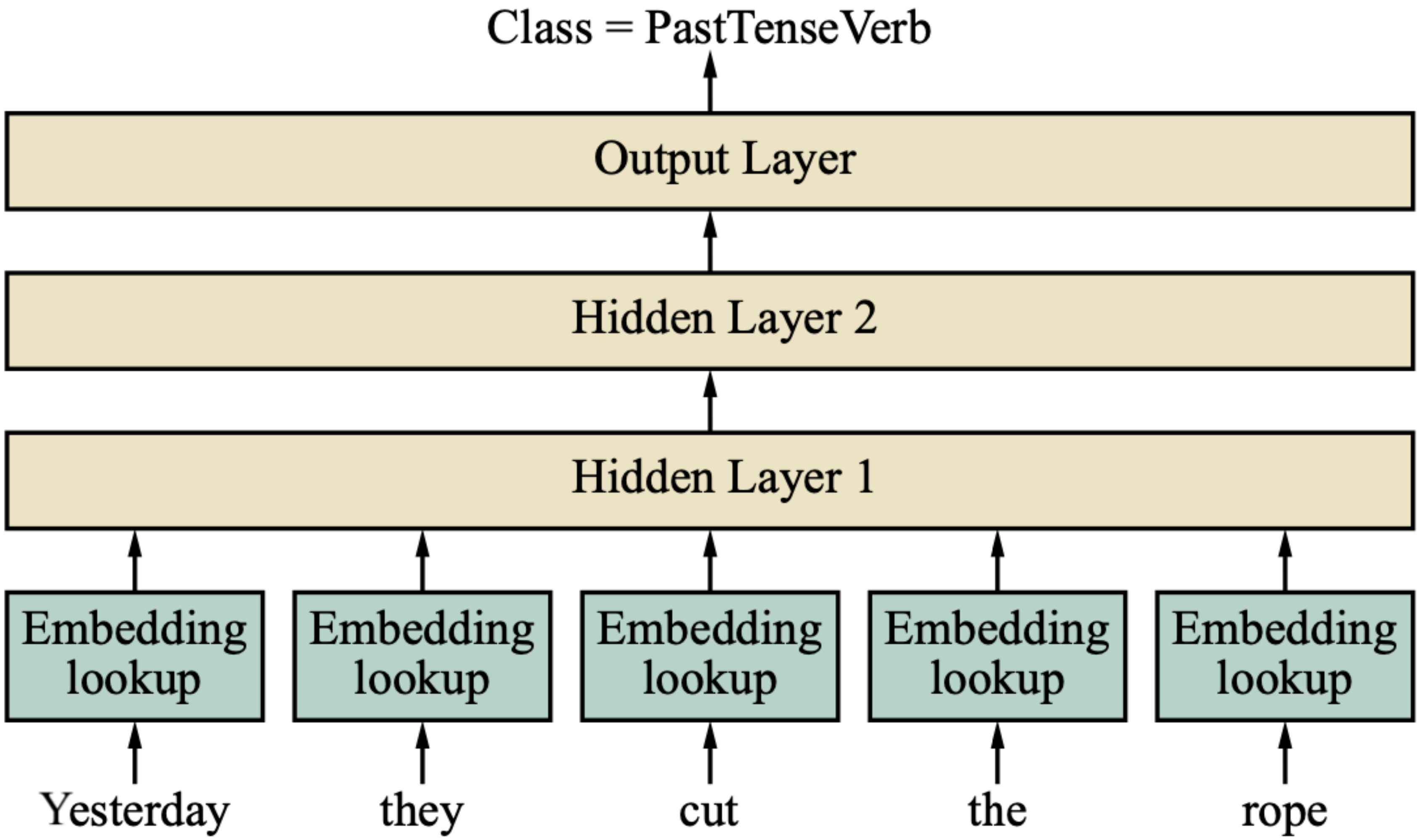
Source: https://developers.google.com/machine-learning/guides/text-classification/step-3

Word embeddings



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

Feedforward part-of-speech (POS) tagging model



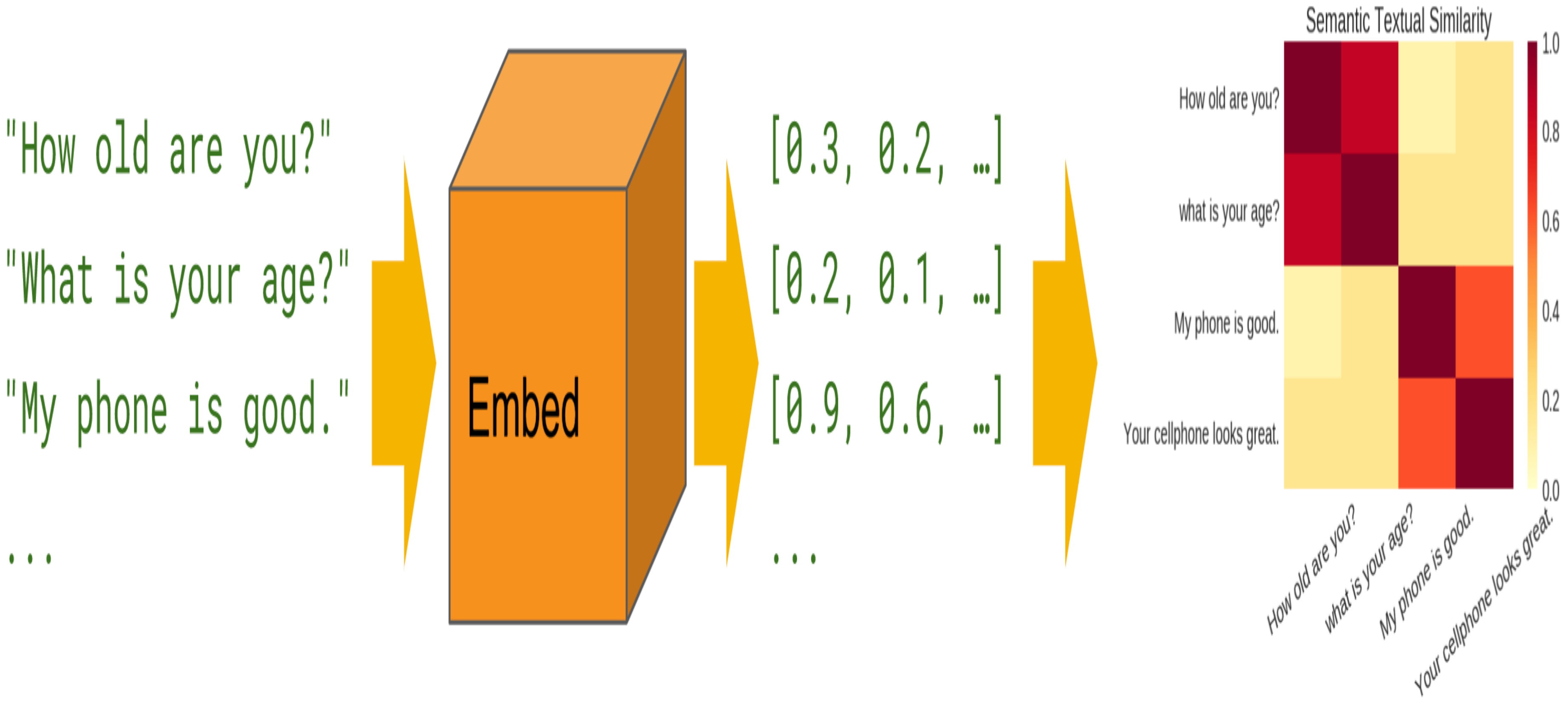
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Universal Sentence Encoder (USE)

- The Universal Sentence Encoder encodes text into high-dimensional vectors that can be used for
  text classification,
  semantic similarity,
  clustering and
  other natural language tasks.
- The universal-sentence-encoder model is trained with a deep averaging network (DAN) encoder.

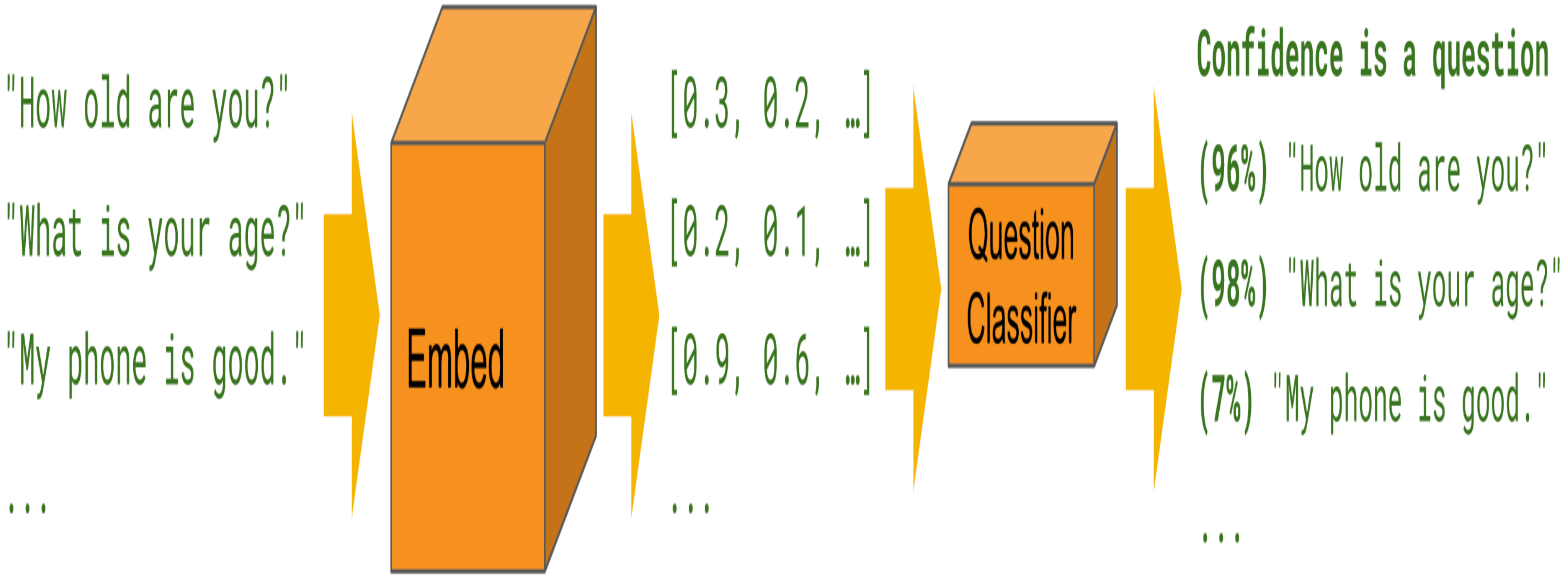
Source: https://tfhub.dev/google/universal-sentence-encoder/4

Universal Sentence Encoder (USE)
Semantic Similarity



Source: https://tfhub.dev/google/universal-sentence-encoder/4

Universal Sentence Encoder (USE) Classification



Source: https://tfhub.dev/google/universal-sentence-encoder/4

Universal Sentence Encoder (USE)

```
import tensorflow_hub as hub

embed = hub.Module("https://tfhub.dev/google/"
    "universal-sentence-encoder/1")

embedding = embed([
    "The quick brown fox jumps over the lazy dog."])
```

Source: Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, Ray Kurzweil. Universal Sentence Encoder. arXiv:1803.11175, 2018.

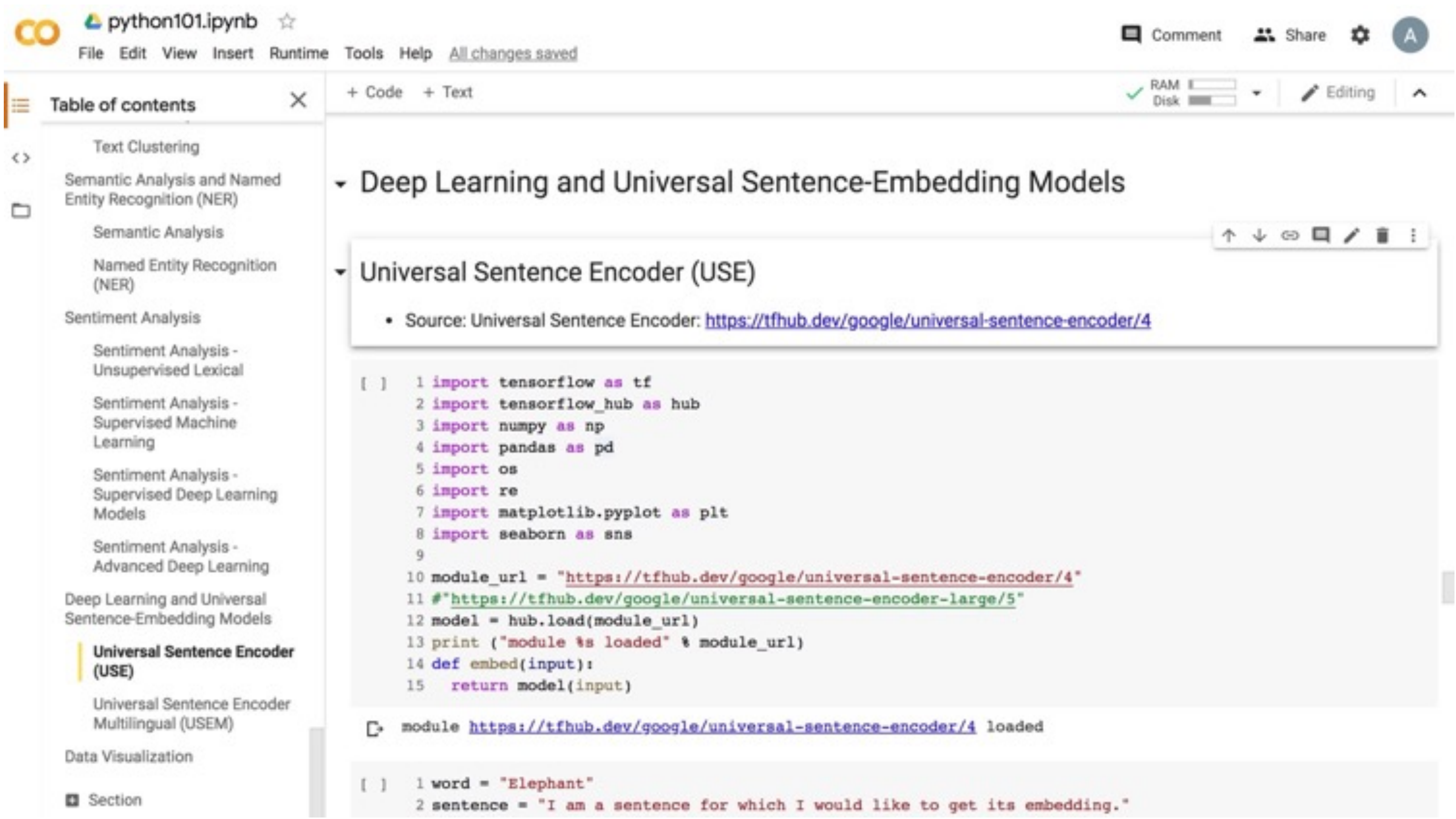
Multilingual Universal Sentence Encoder (MUSE)

```
import tensorflow_hub as hub

module = hub.Module("https://tfhub.dev/google/"
    "universal-sentence-encoder-multilingual/1")

multilingual_embeddings = module([
  "Hola Mundo!", "Bonjour le monde!", "Ciao mondo!"
  "Hello World!", "Hallo Welt!", "Hallo Wereld!",
  "你好世界!", "Привет, мир!", "!مرحبا بالعالم"])
```

Source: Yinfei Yang, Daniel Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, Gustavo Hernandez Abrego , Steve Yuan, Chris Tar, Yun-hsuan Sung, Ray Kurzweil. Multilingual Universal Sentence Encoder for Semantic Retrieval. July 2019

Python in Google Colab (Python101)

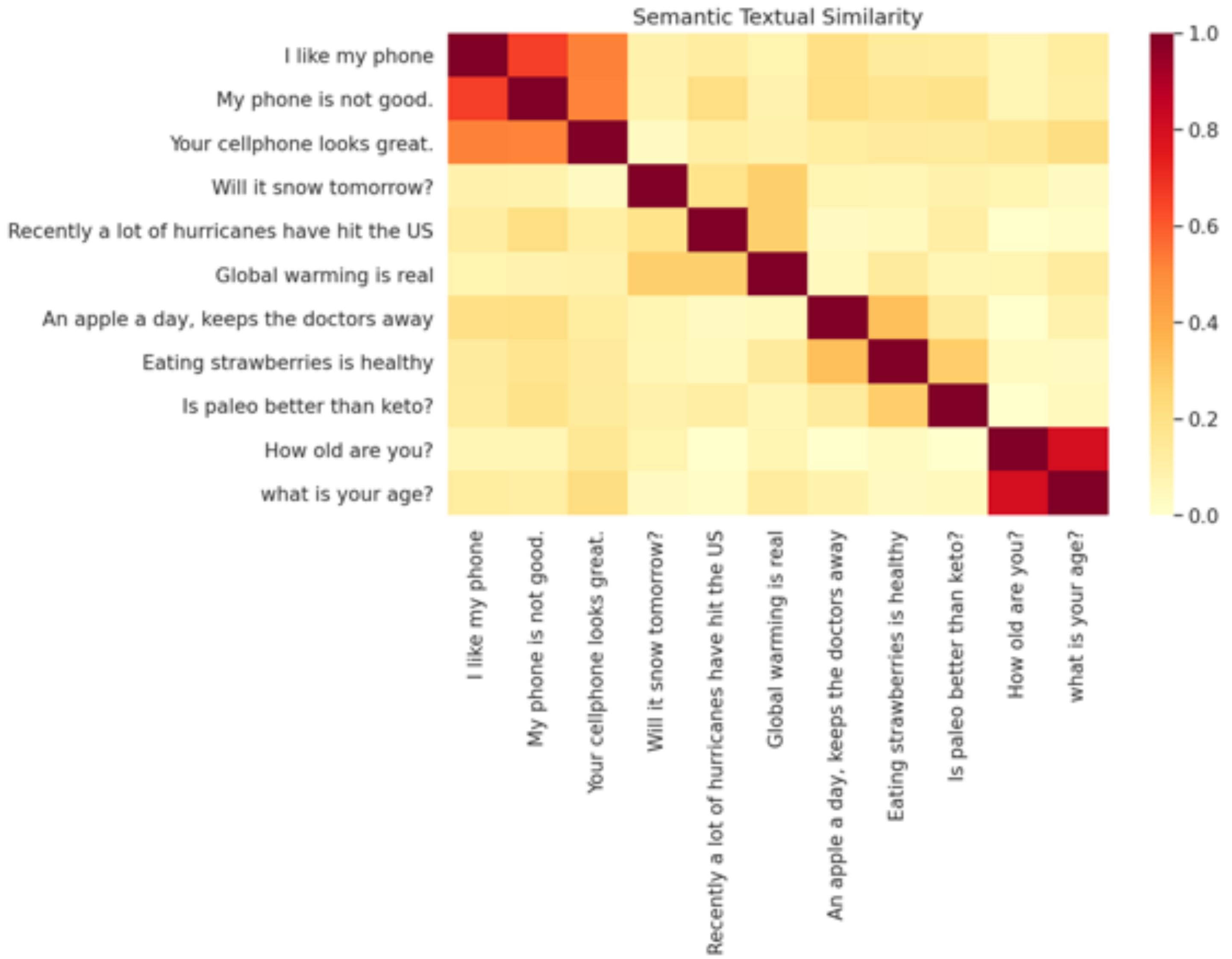
https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

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

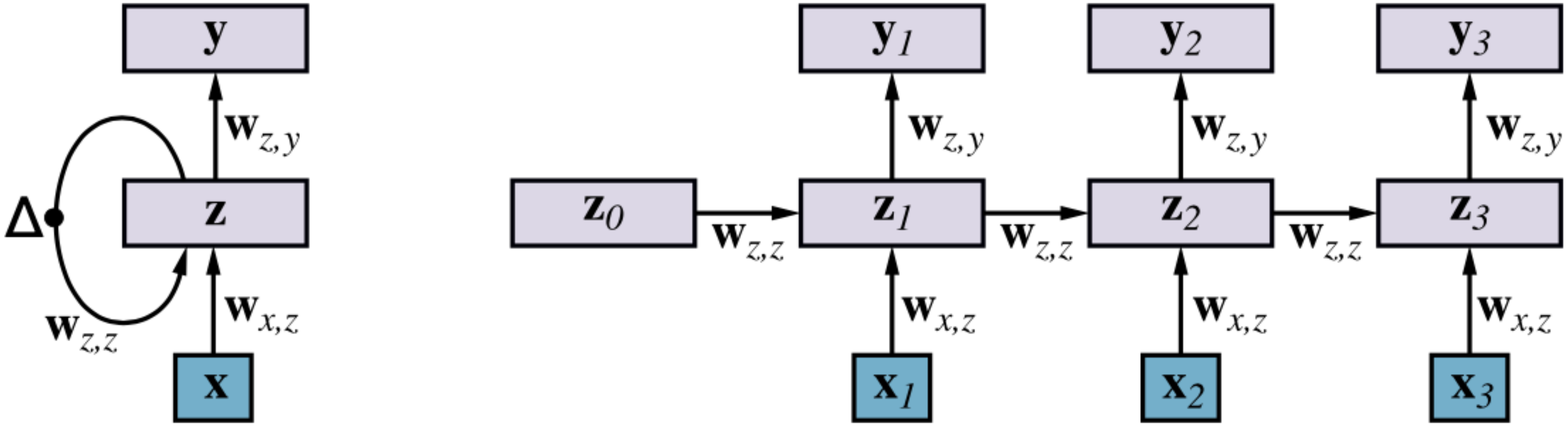


https://tinyurl.com/aintpupython101

Outline

- Word Embeddings
- Recurrent Neural Networks for NLP
- Sequence-to-Sequence Models
- The Transformer Architecture
- Pretraining and Transfer Learning
- State of the art (SOTA)

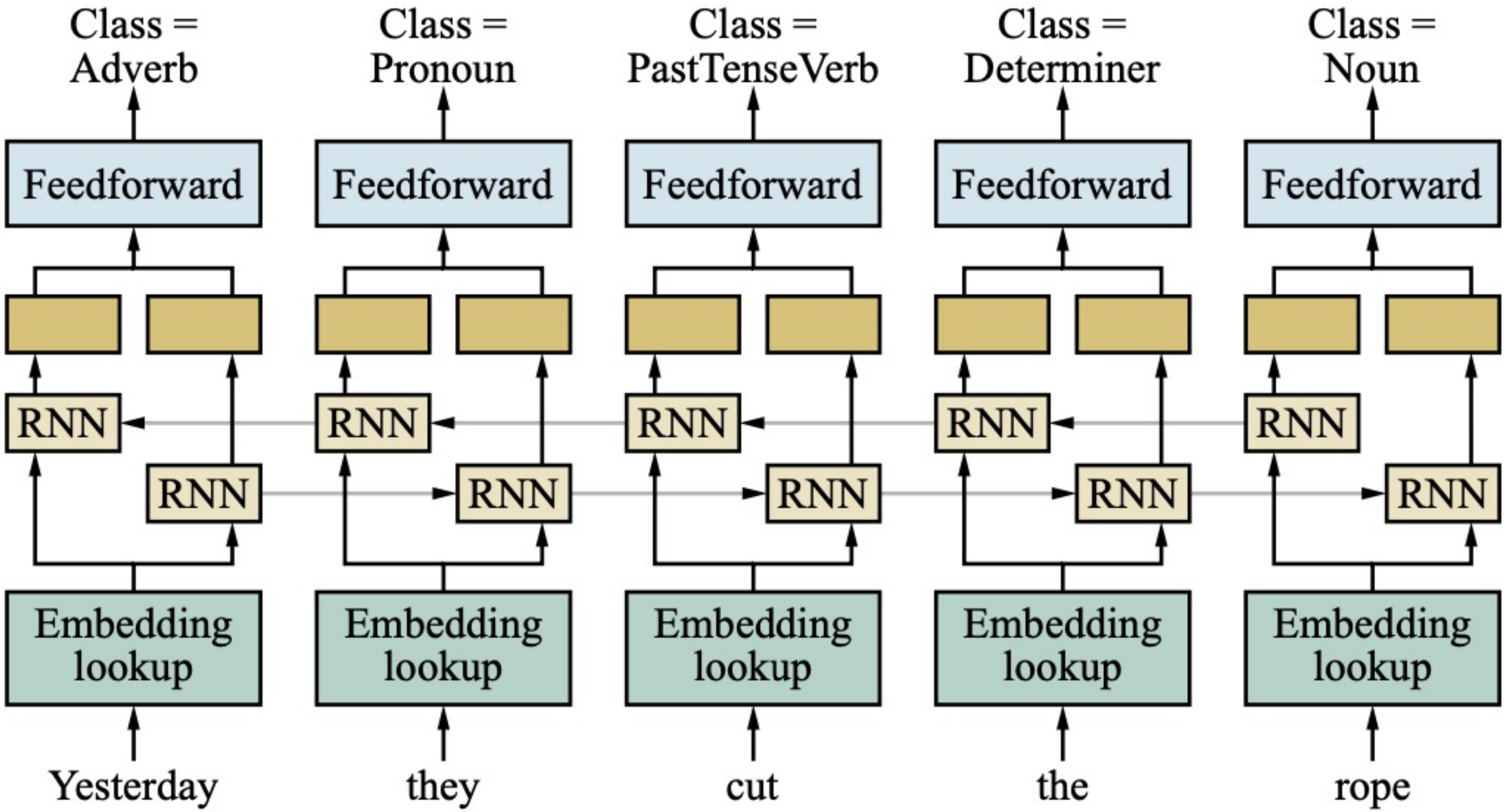
RNN



(a) (b)

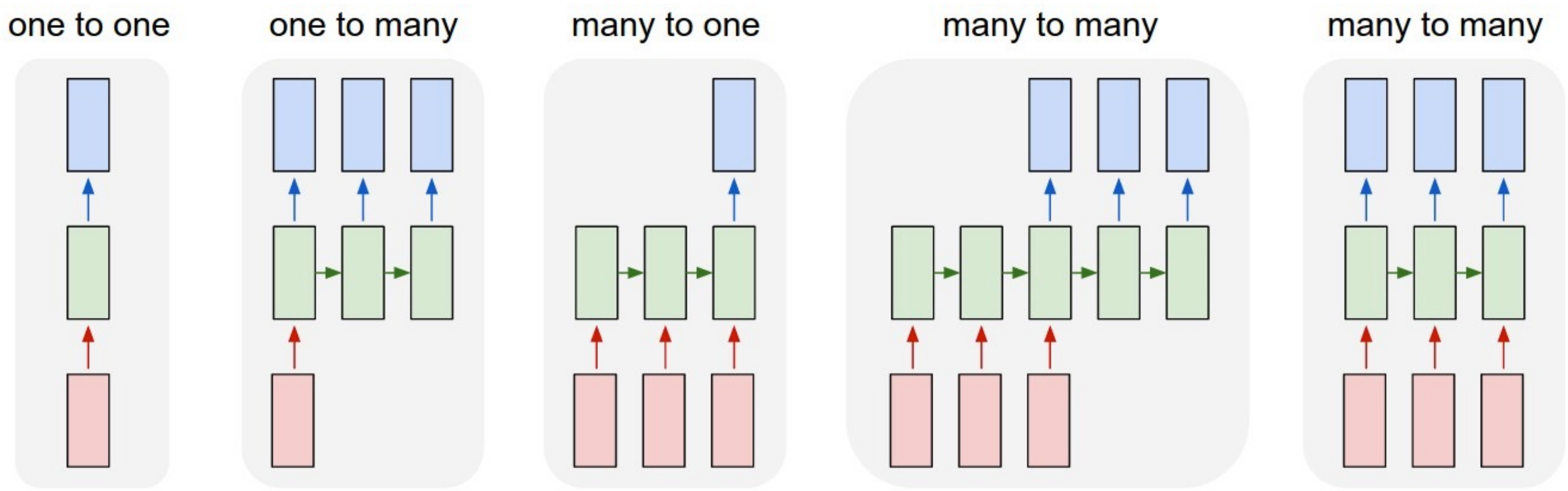
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Bidirectional RNN network for POS tagging



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

LSTM Recurrent Neural Network



Traditional Neural Network

Music Generation

Sentiment Classification

Name Entity Recognition

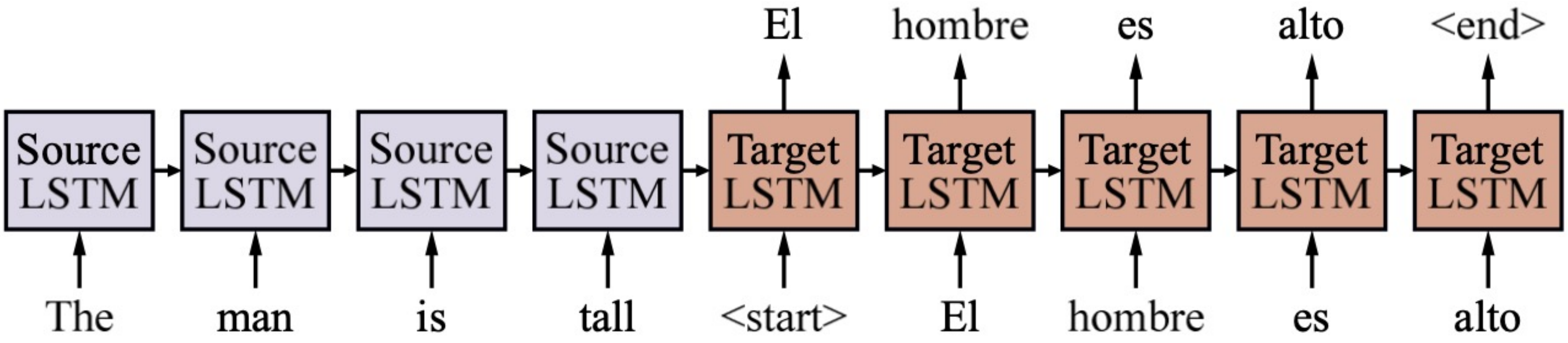
Machine Translation

Source: https://github.com/Vict0rSch/deep_learning/tree/master/keras/recurrent

Outline

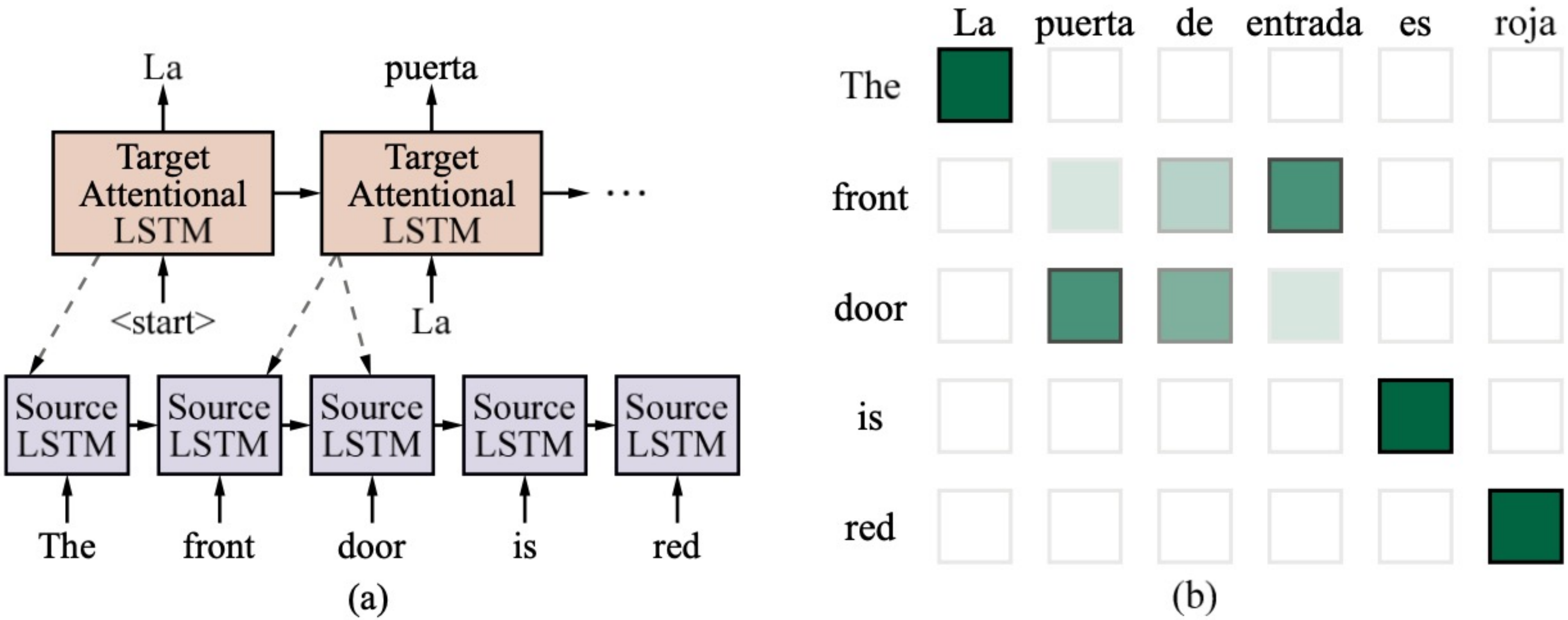
- Word Embeddings
- Recurrent Neural Networks for NLP
- Sequence-to-Sequence Models
- The Transformer Architecture
- Pretraining and Transfer Learning
- State of the art (SOTA)

Sequence-to-Sequence model



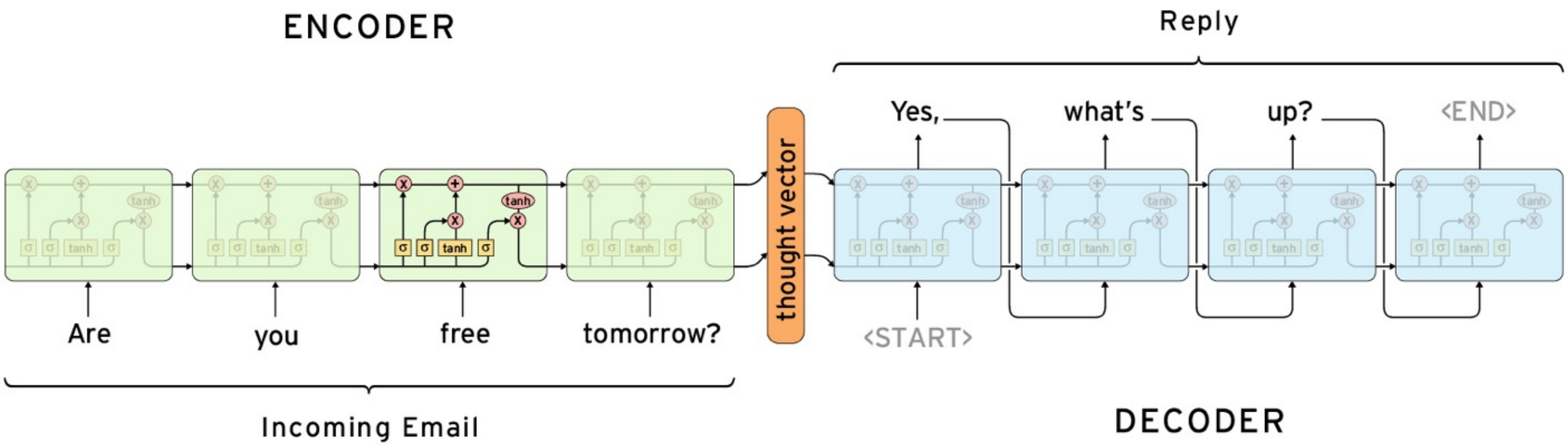
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Attentional Sequence-to-Sequence model for English-to-Spanish translation



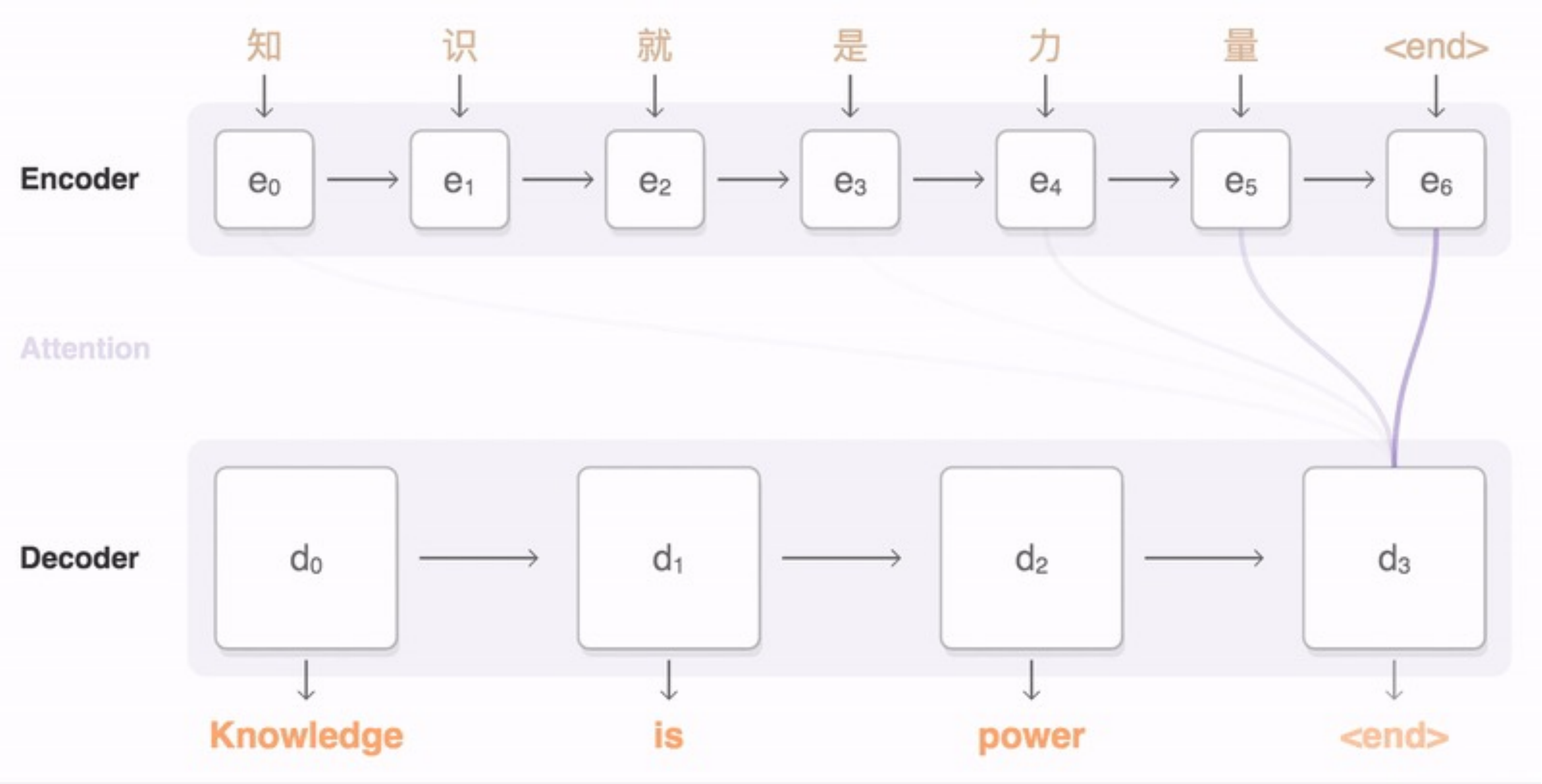
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

The Sequence to Sequence model (seq2seq)



Source: http://suriyadeepan.github.io/2016-12-31-practical-seq2seq/

Sequence to Sequence (Seq2Seq)



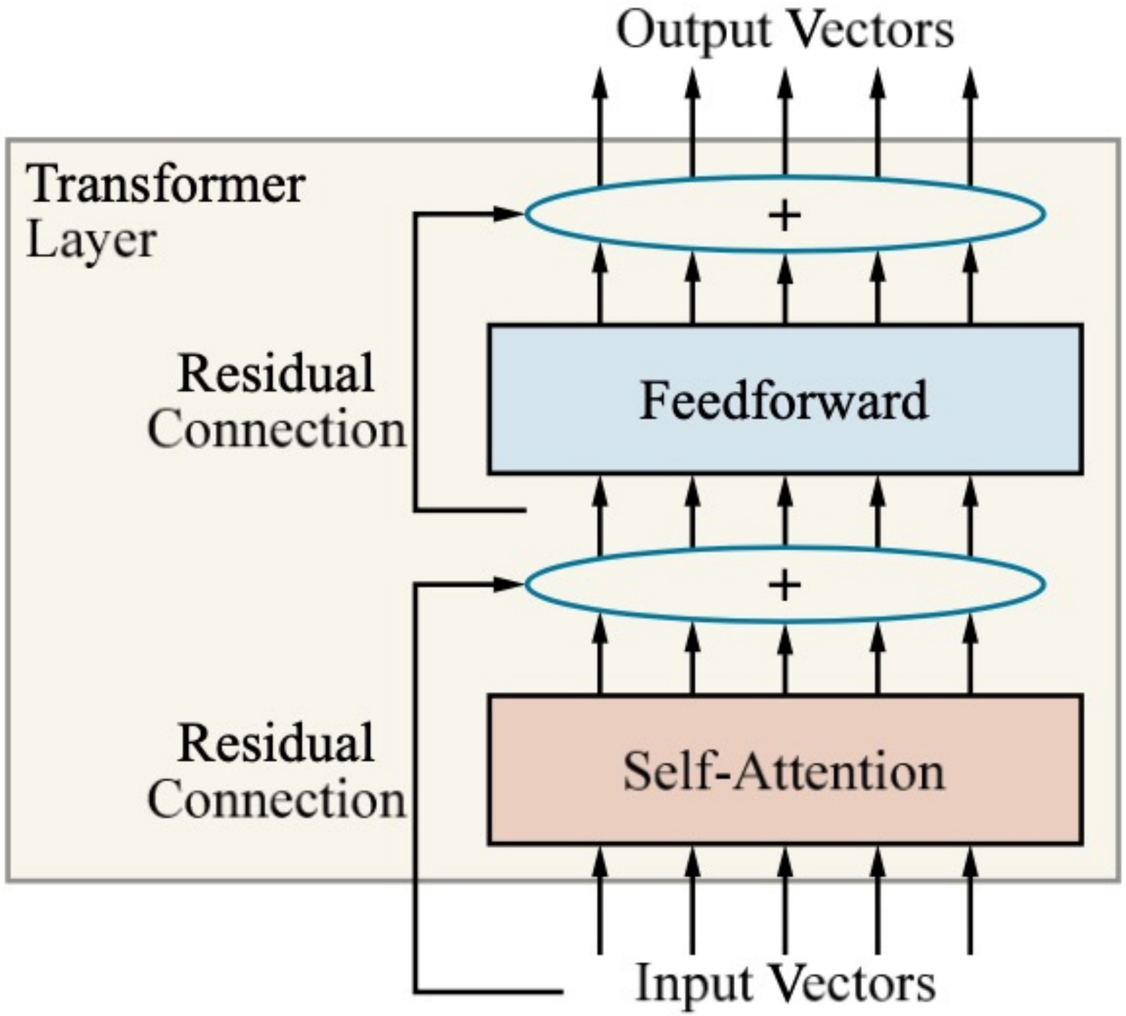
Source: https://google.github.io/seq2seq/

Outline

- Word Embeddings
- Recurrent Neural Networks for NLP
- Sequence-to-Sequence Models
- The Transformer Architecture
- Pretraining and Transfer Learning
- State of the art (SOTA)

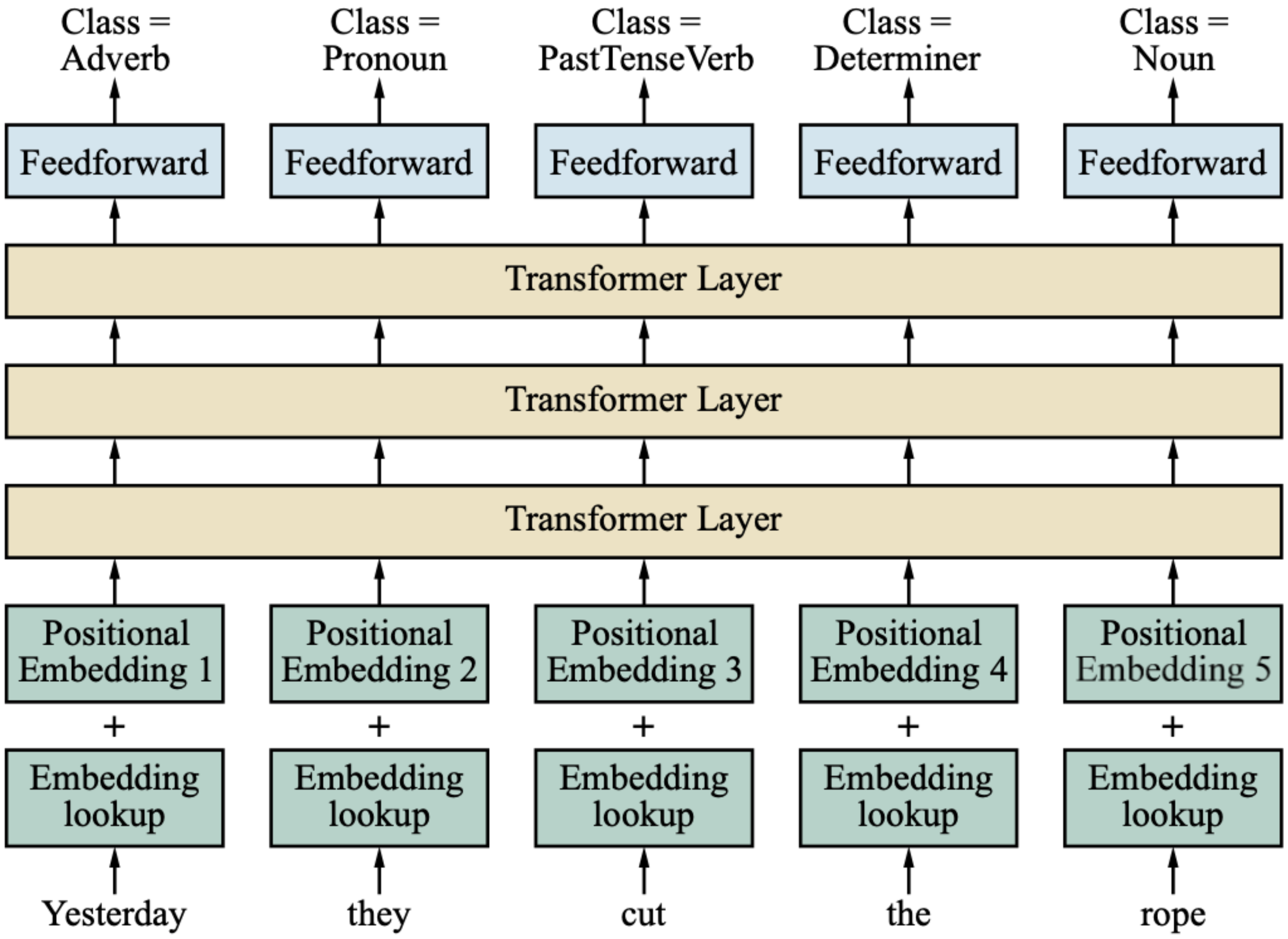
Single-layer Transformer

consists of self-attention,
a feedforward network, and residual connection



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

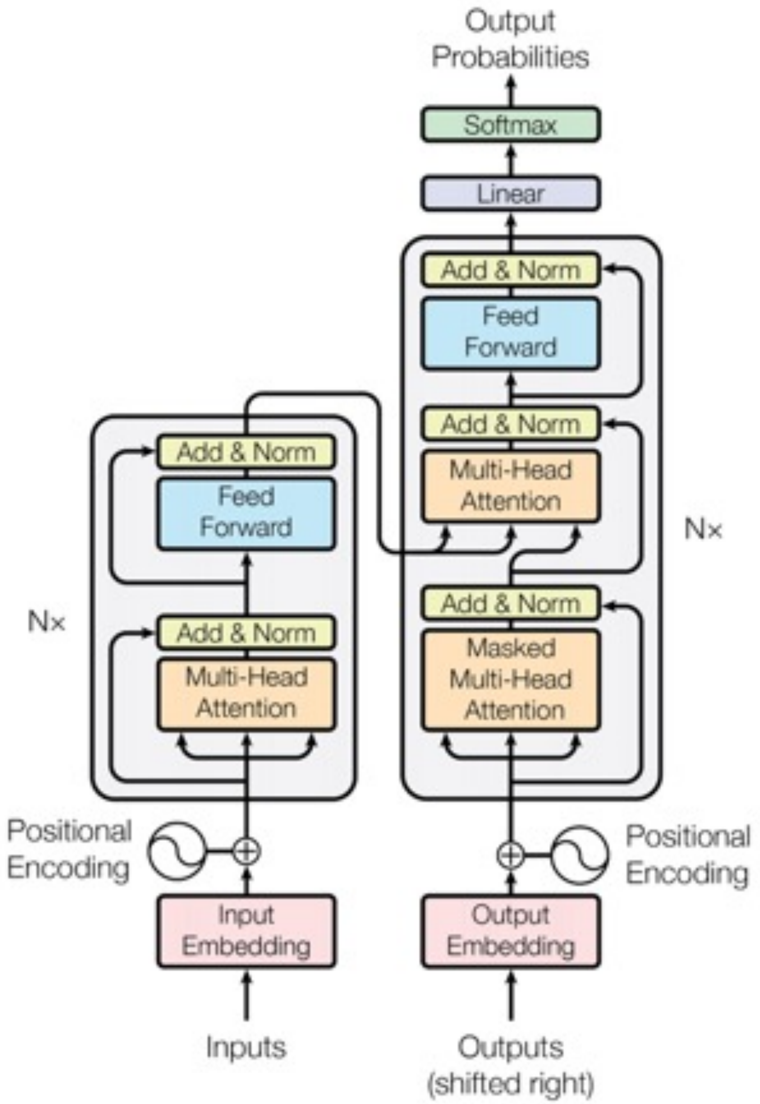
Transformer Architecture for POS Tagging



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Transformer (Attention is All You Need)

(Vaswani et al., 2017)

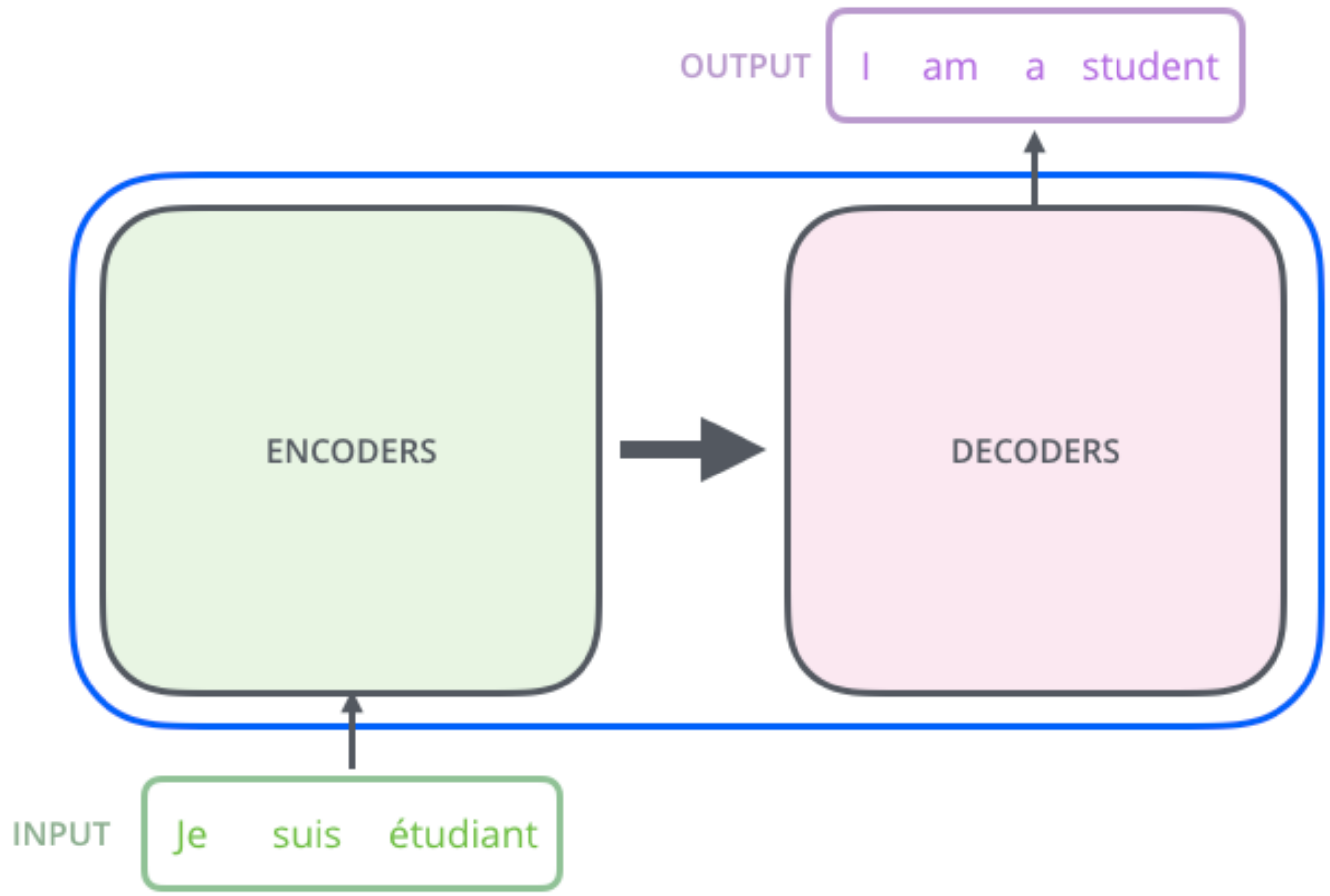


Source: Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In *Advances in neural information processing systems*, pp. 5998-6008. 2017.

Transformer

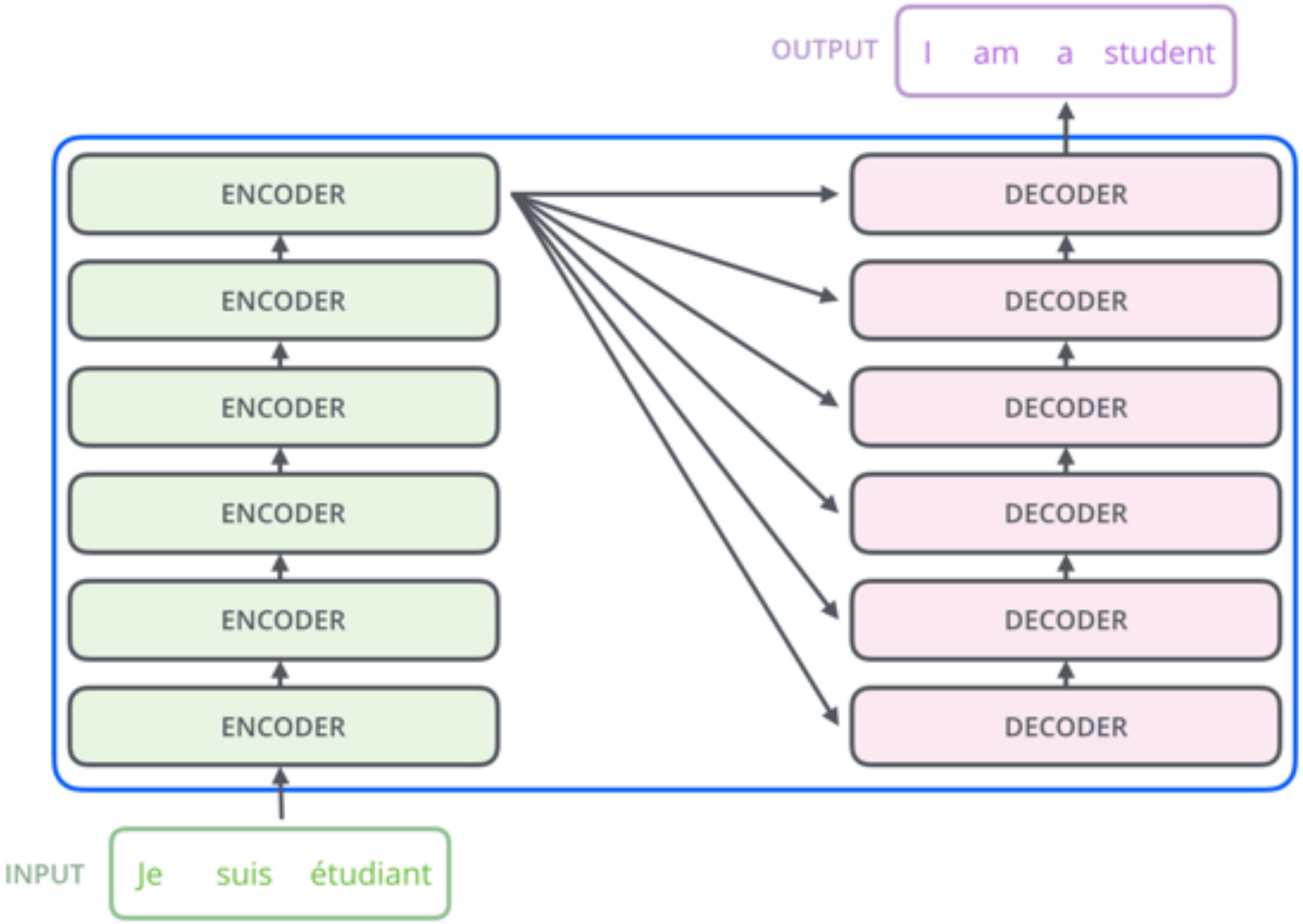
Source: Jay Alammar (2019), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Transformer Encoder Decoder



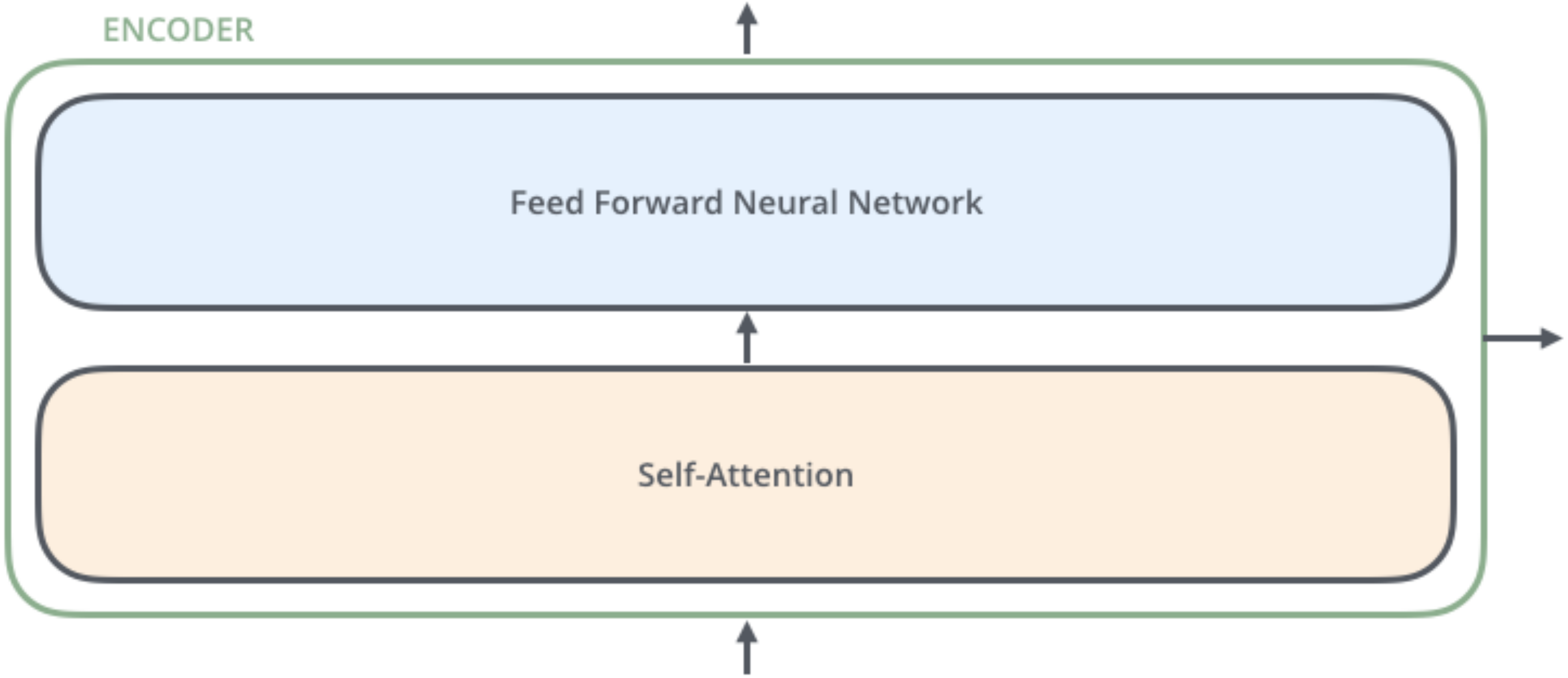
Source: Jay Alammar (2019), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Transformer
Encoder Decoder Stack



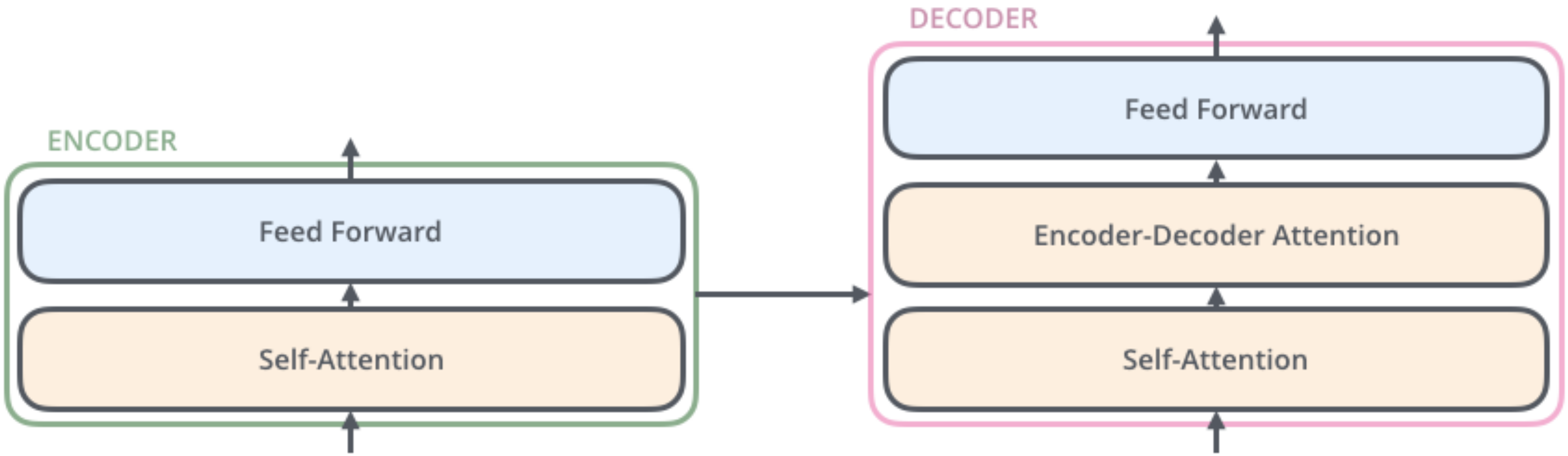
Source: Jay Alammar (2019), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Transformer
Encoder Self-Attention



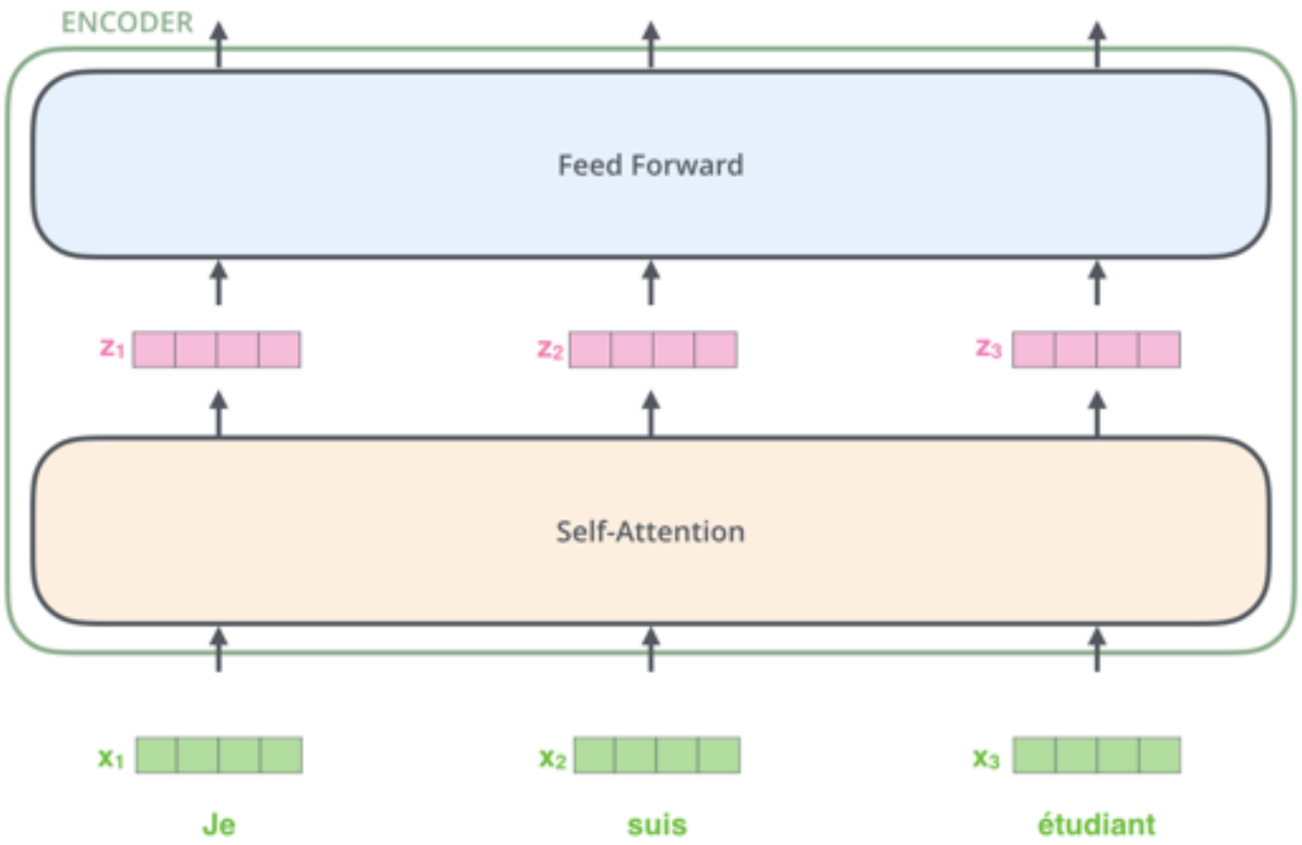
Source: Jay Alammar (2019), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Transformer Decoder



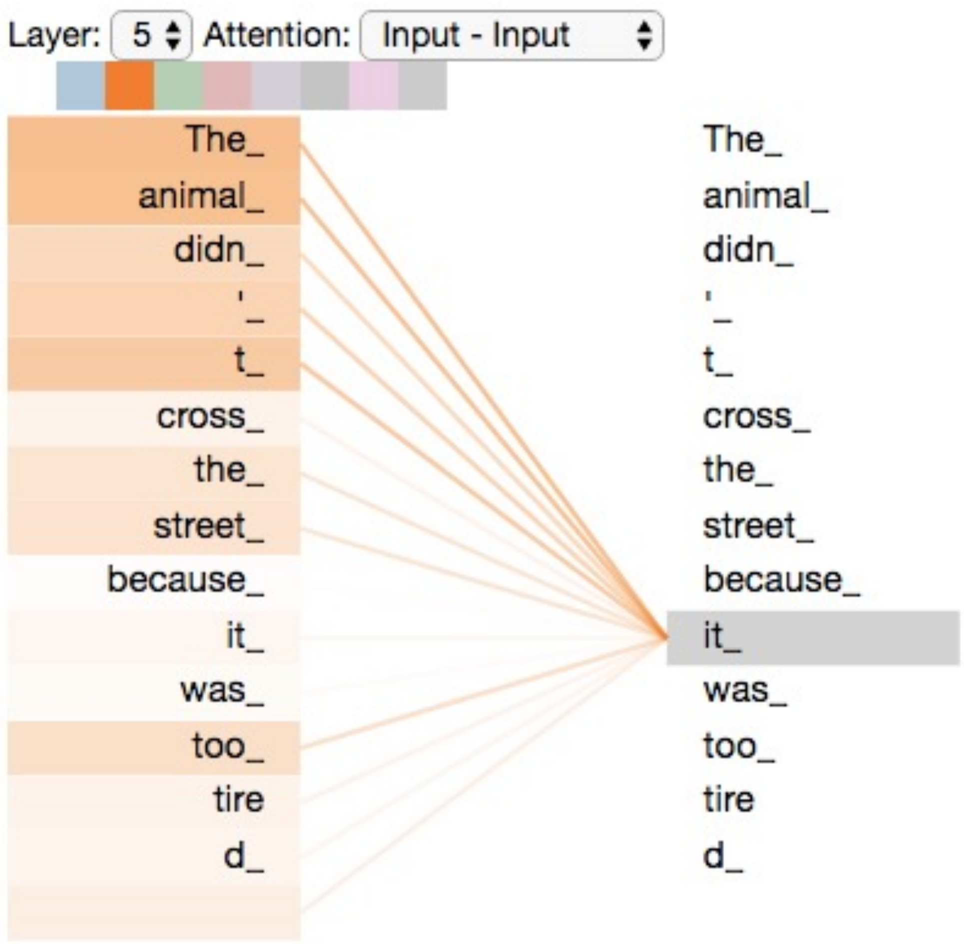
Source: Jay Alammar (2019), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Transformer Encoder with Tensors Word Embeddings



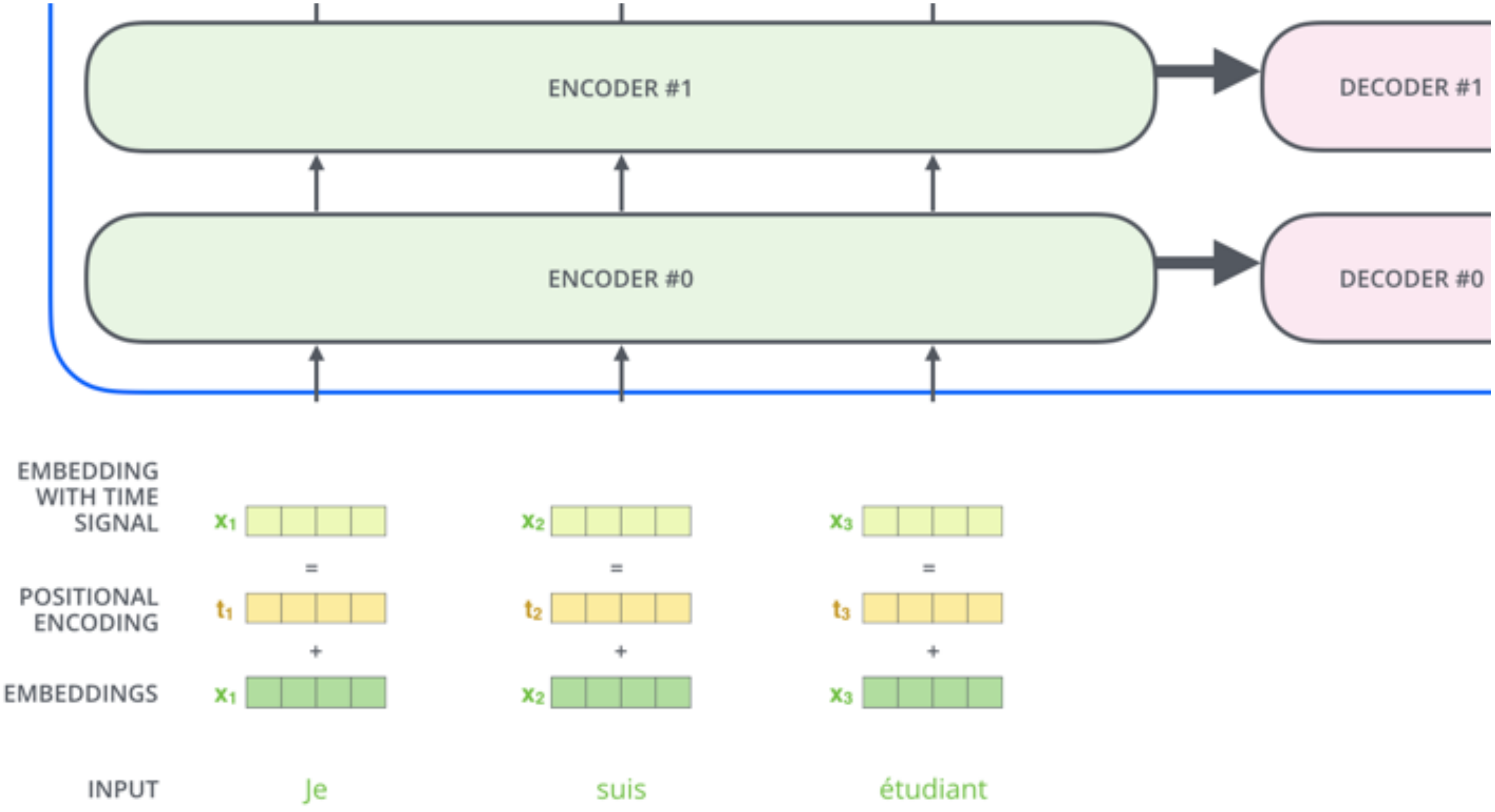
Source: Jay Alammar (2019), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Transformer Self-Attention Visualization



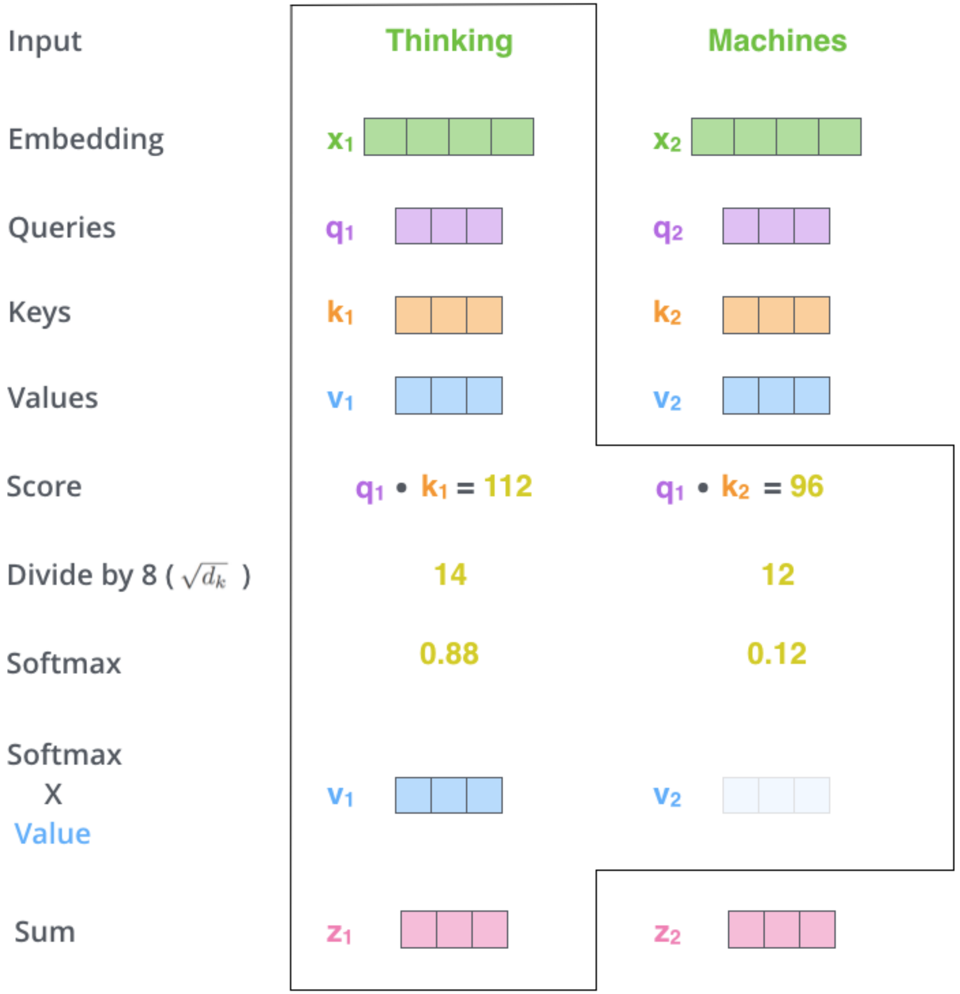
Source: Jay Alammar (2019), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Transformer
Positional Encoding Vectors



Source: Jay Alammar (2019), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Transformer
Self-Attention Softmax Output



Source: Jay Alammar (2019), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Outline

- Word Embeddings
- Recurrent Neural Networks for NLP
- Sequence-to-Sequence Models
- The Transformer Architecture
- Pretraining and Transfer Learning
- State of the art (SOTA)

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

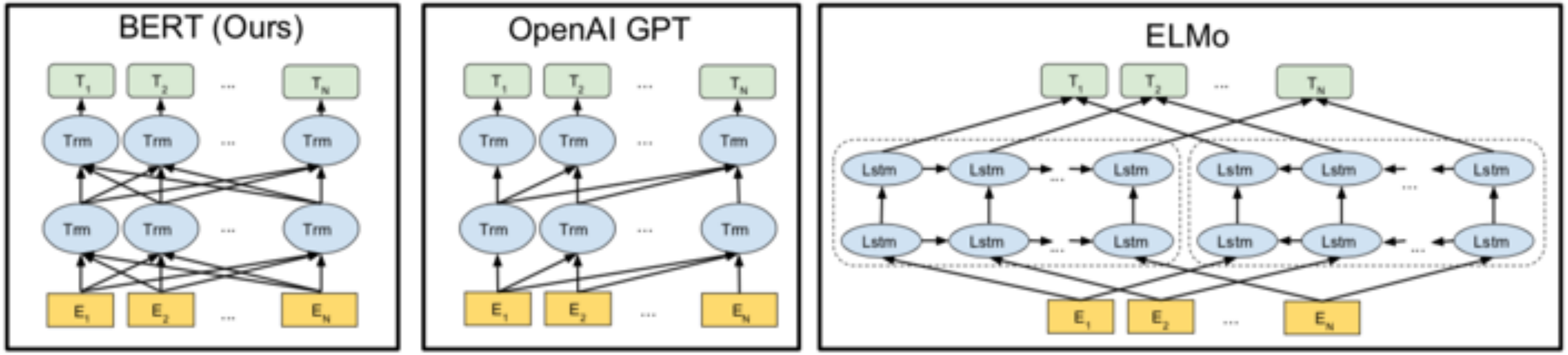
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

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

Bidirectional Encoder Representations from Transformers



Pre-training model architectures

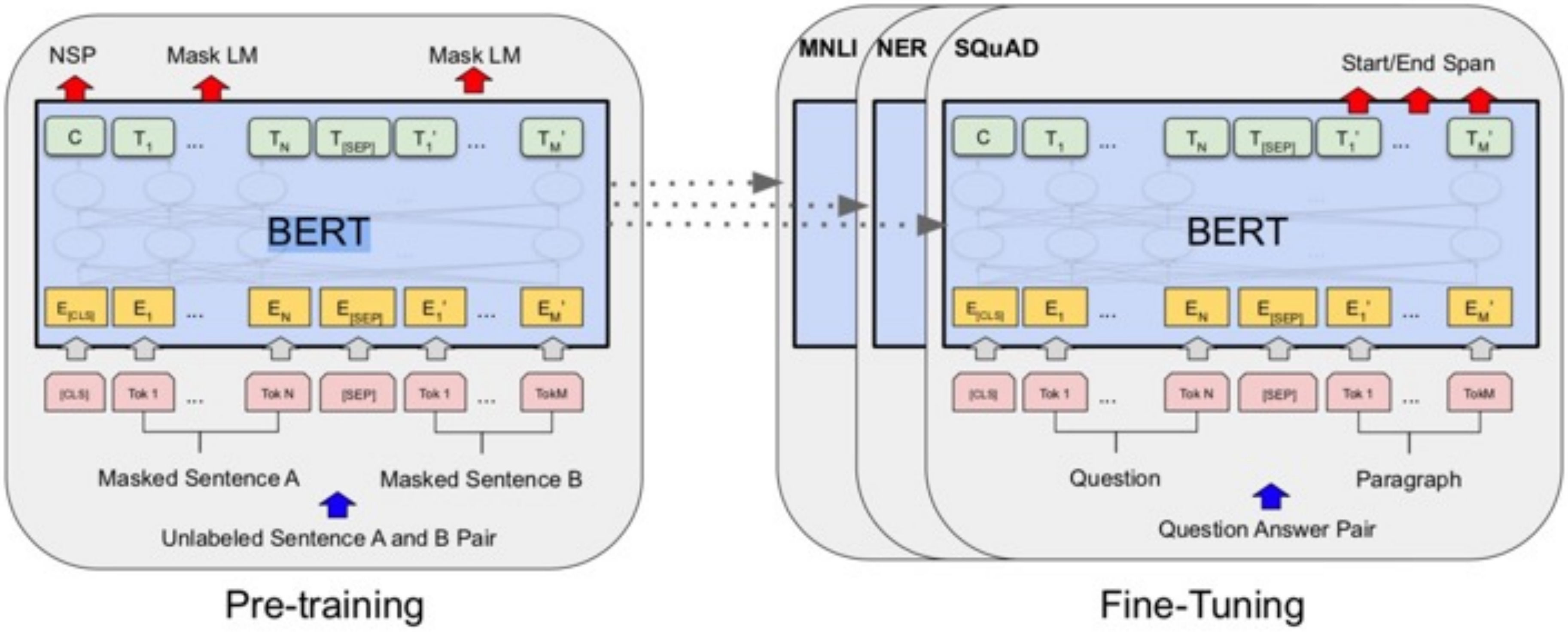
BERT uses a bidirectional Transformer.
OpenAI GPT uses a left-to-right Transformer.
ELMo uses the concatenation of independently trained left-to-right and right- to-left LSTM to generate features for downstream tasks.
Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

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: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT
(Bidirectional Encoder Representations from Transformers)

Overall pre-training and fine-tuning procedures for BERT

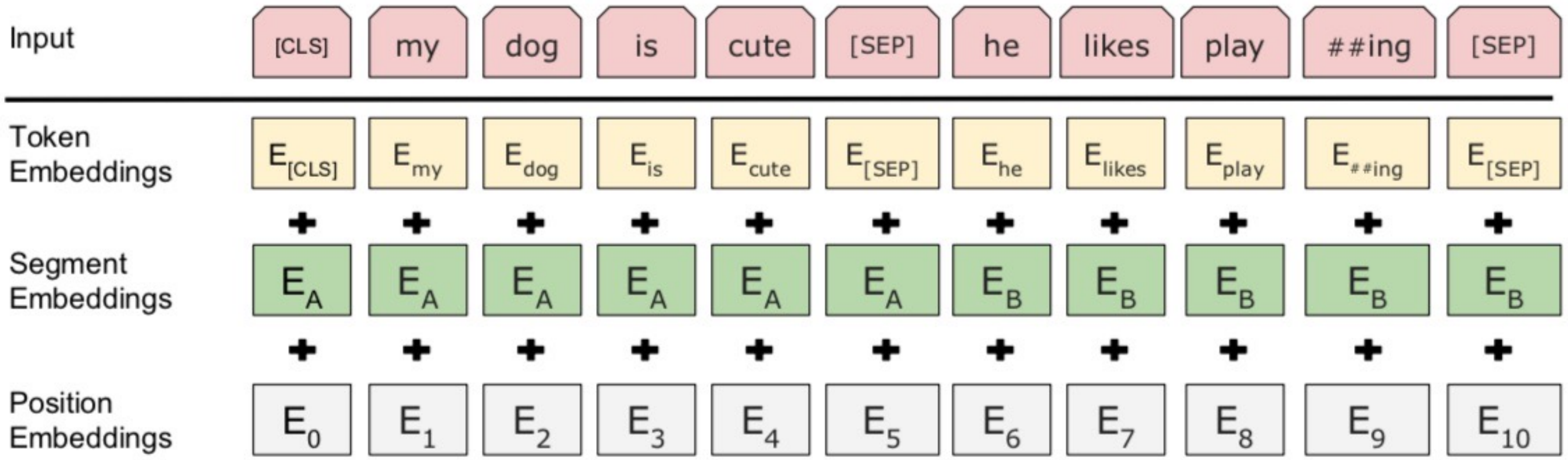


Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT (Bidirectional Encoder Representations from Transformers)

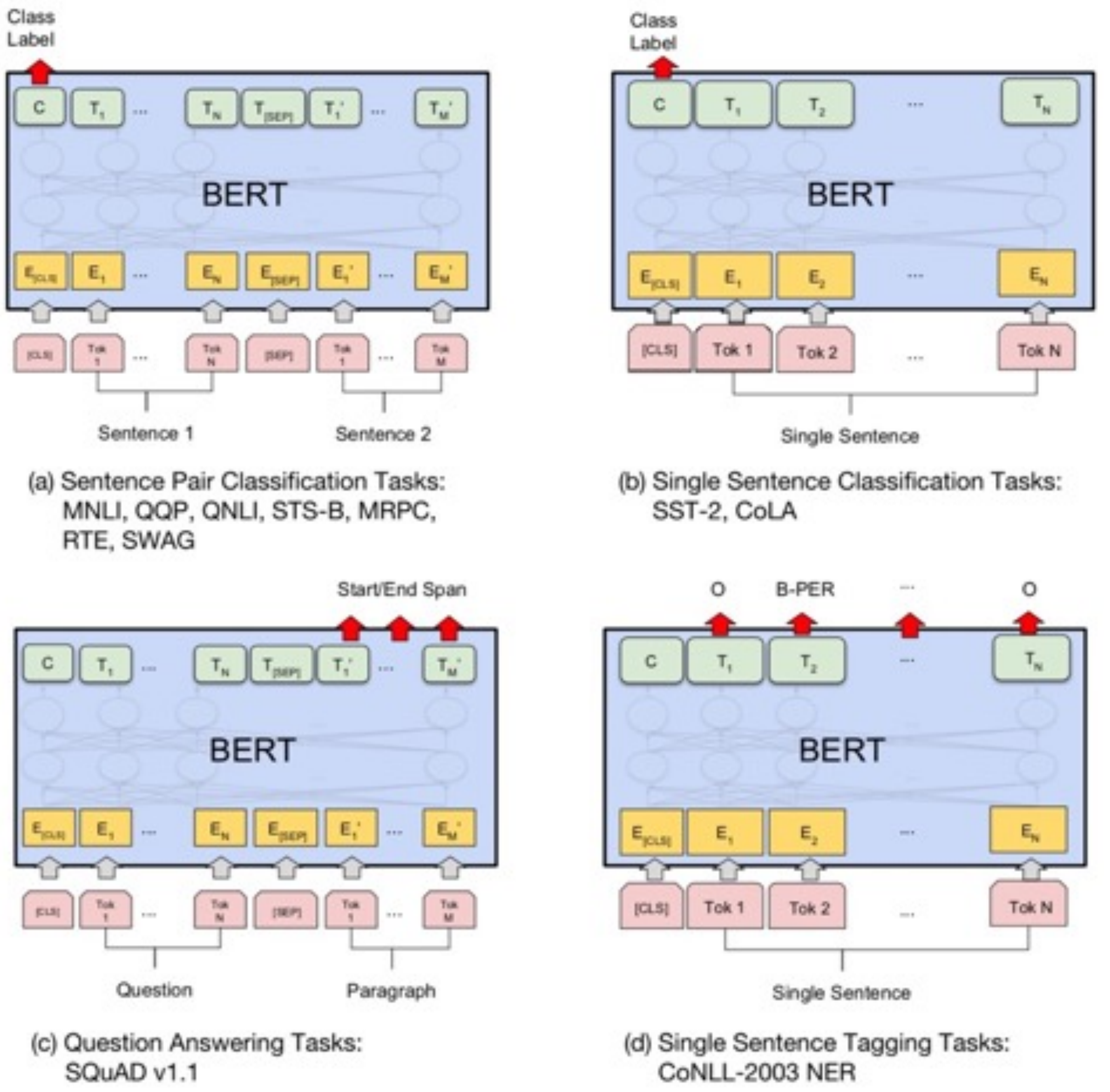
BERT input representation



The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

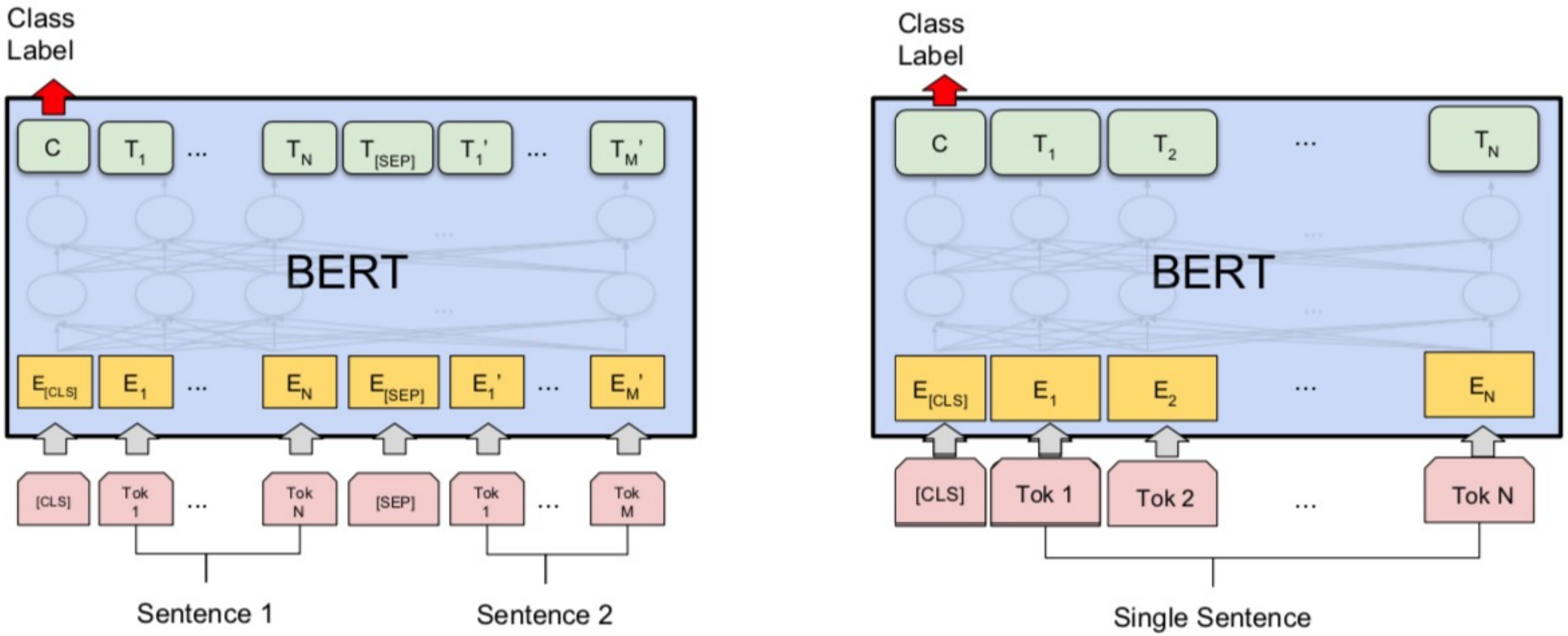
Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

Fine-tuning BERT on Different 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.

BERT Sequence-level tasks

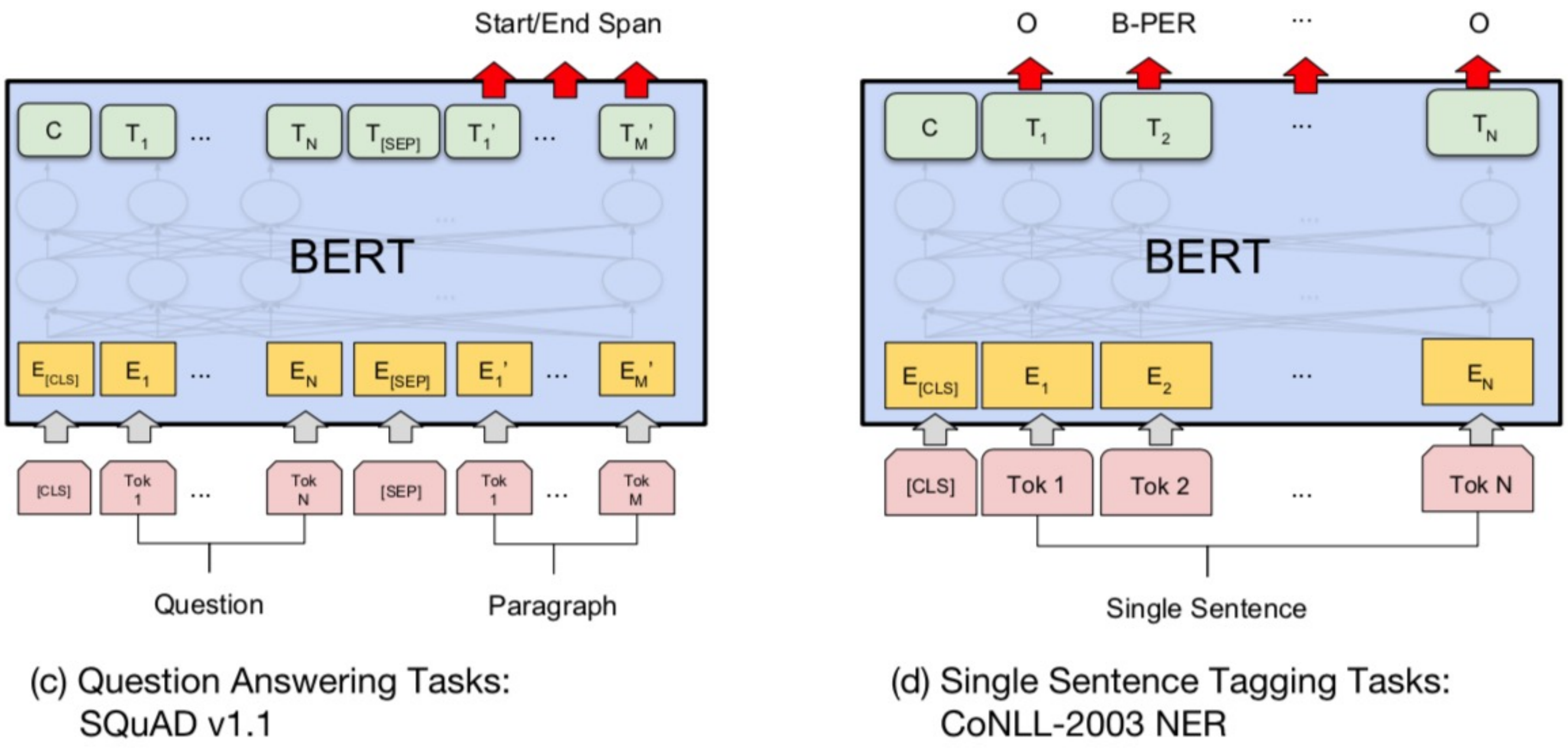


(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

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

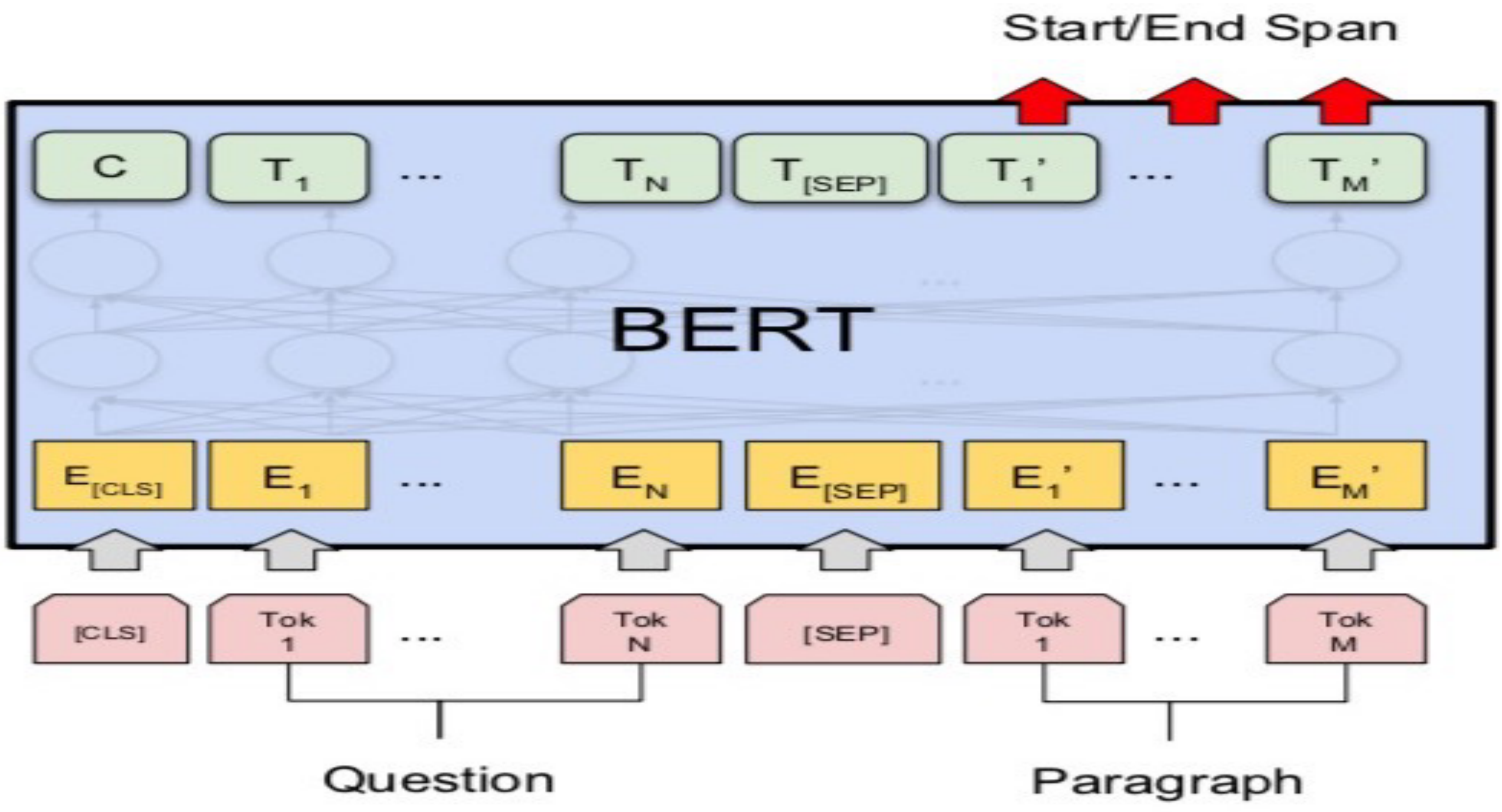
Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

BERT Token-level tasks



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

Fine-tuning BERT on Question Answering (QA)

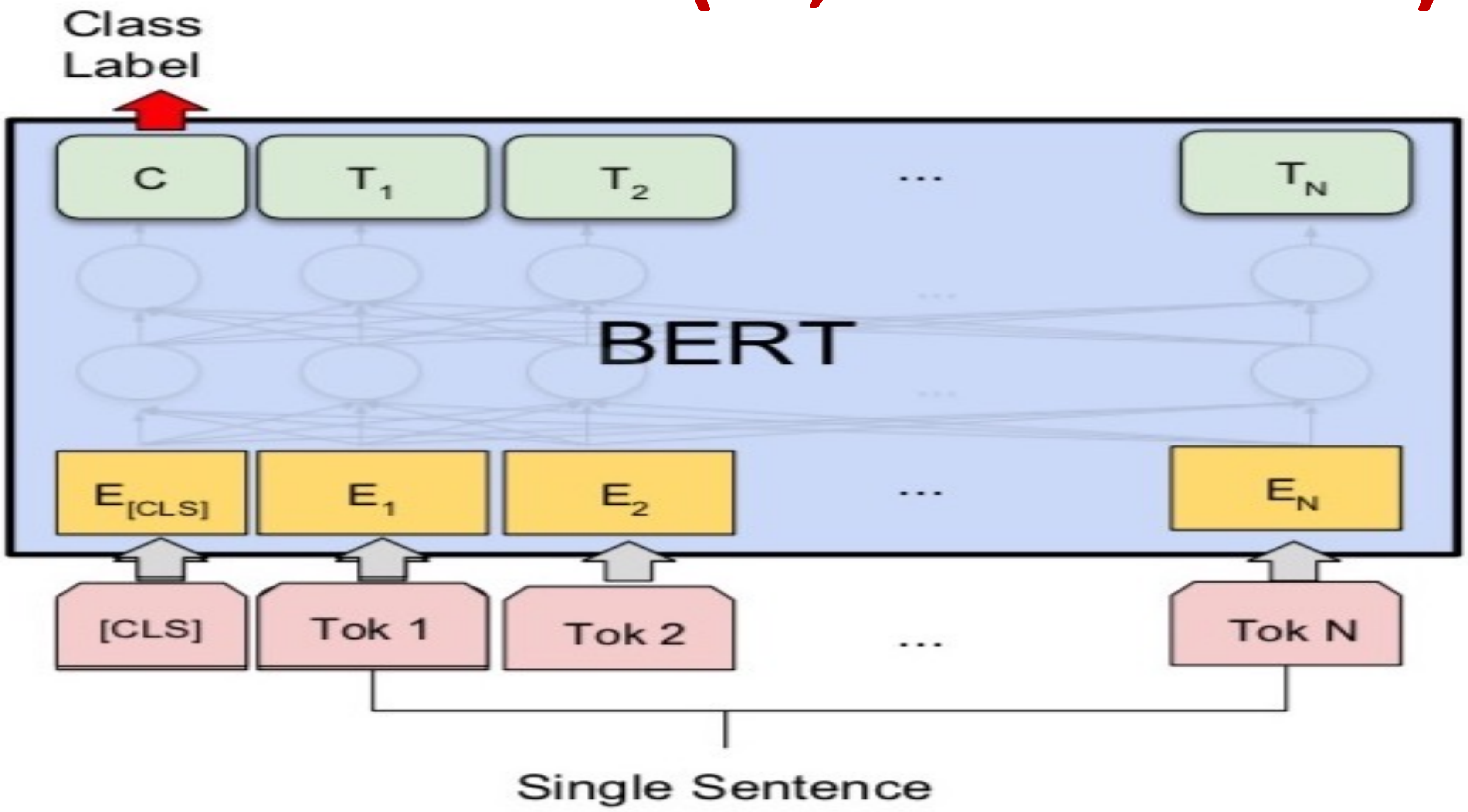


(c) Question Answering Tasks: SQuAD v1.1

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.

Fine-tuning BERT on Dialogue

Intent Detection (ID; Classification)

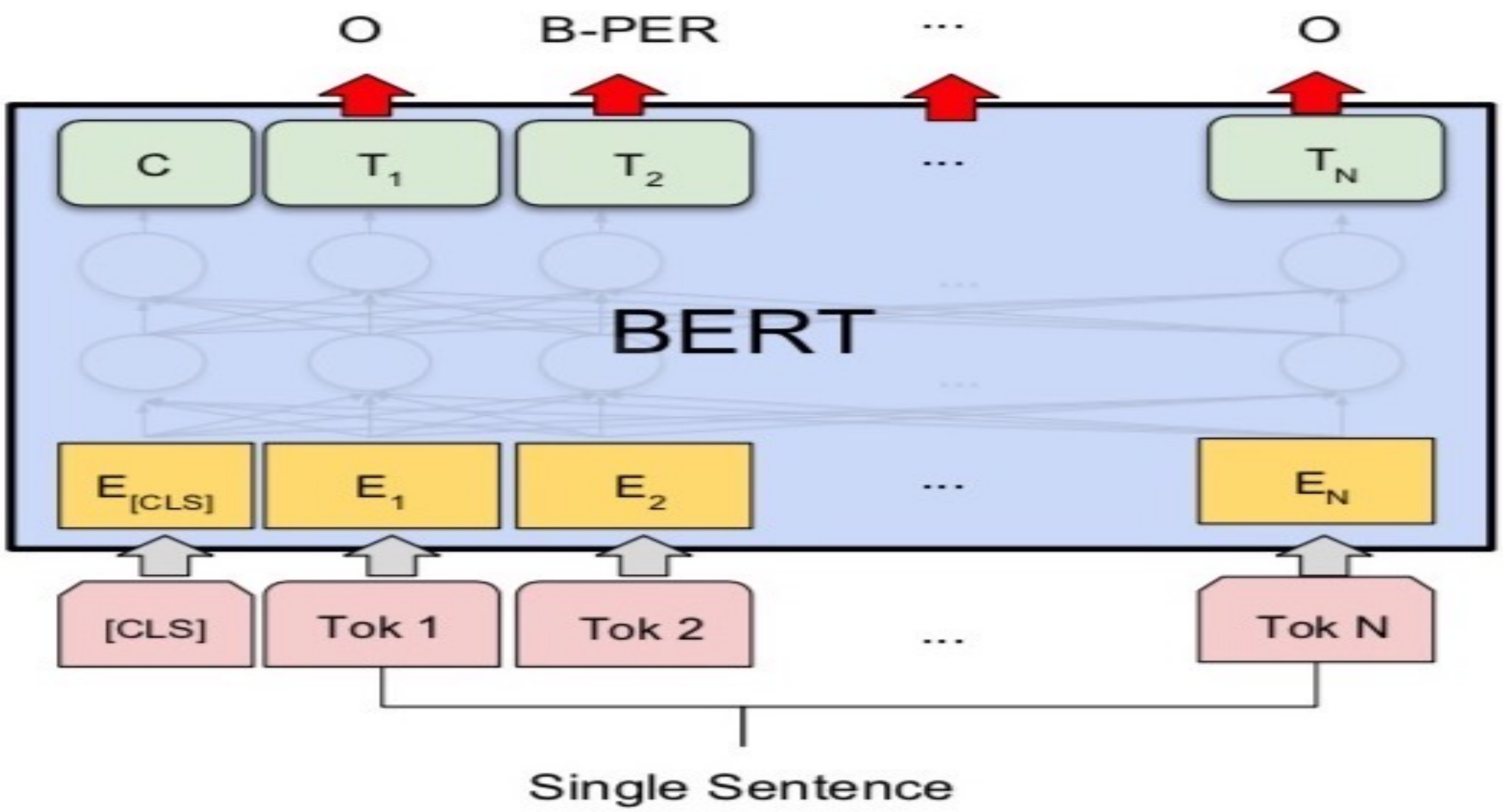


(b) Single Sentence Classification Tasks: SST-2, CoLA

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.

Fine-tuning BERT on Dialogue

Slot Filling (SF)



(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.

General Language Understanding Evaluation (GLUE) benchmark

GLUE Test results

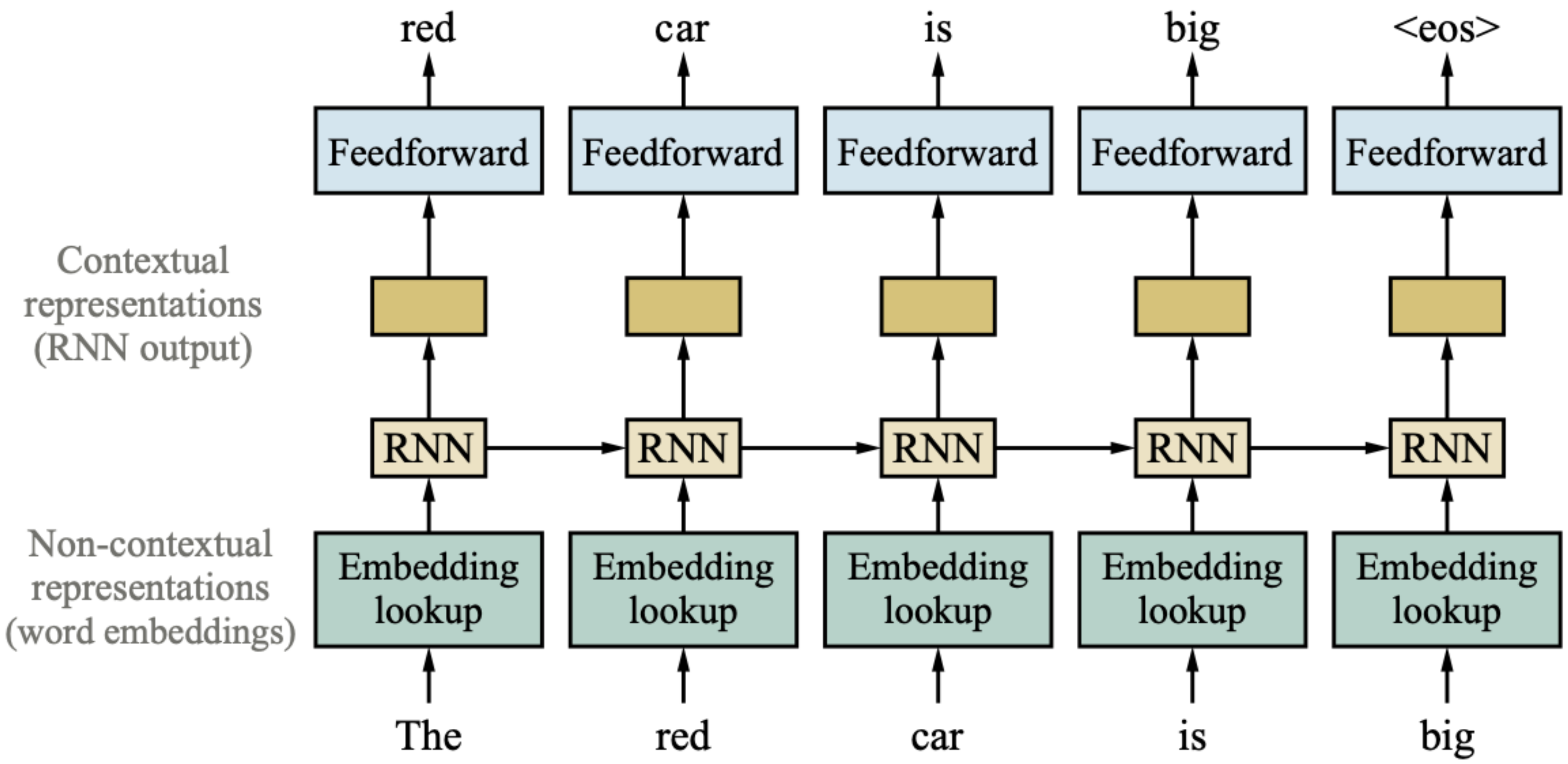
System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2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

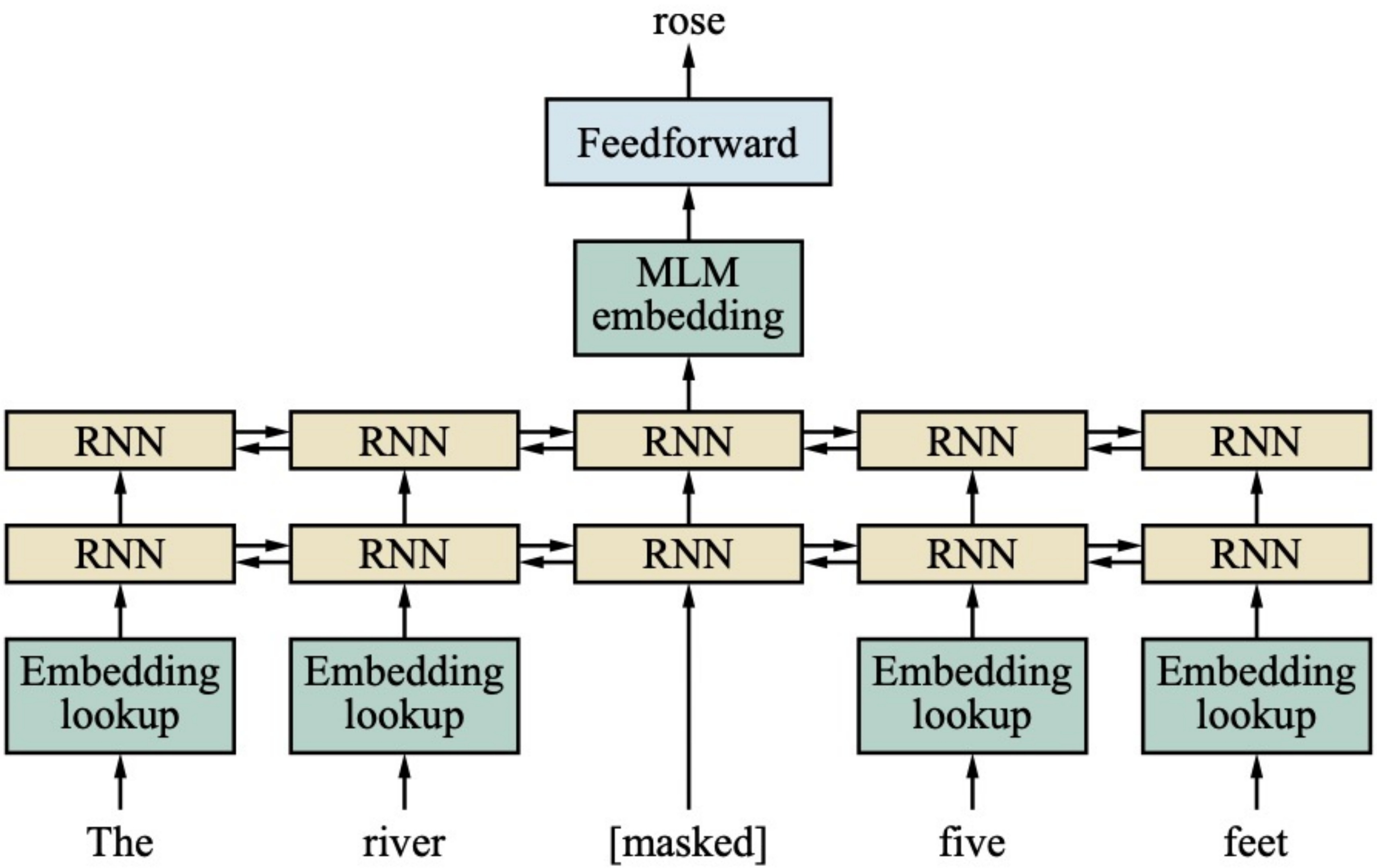
Training Contextual Representations

using a left-to-right Language Model



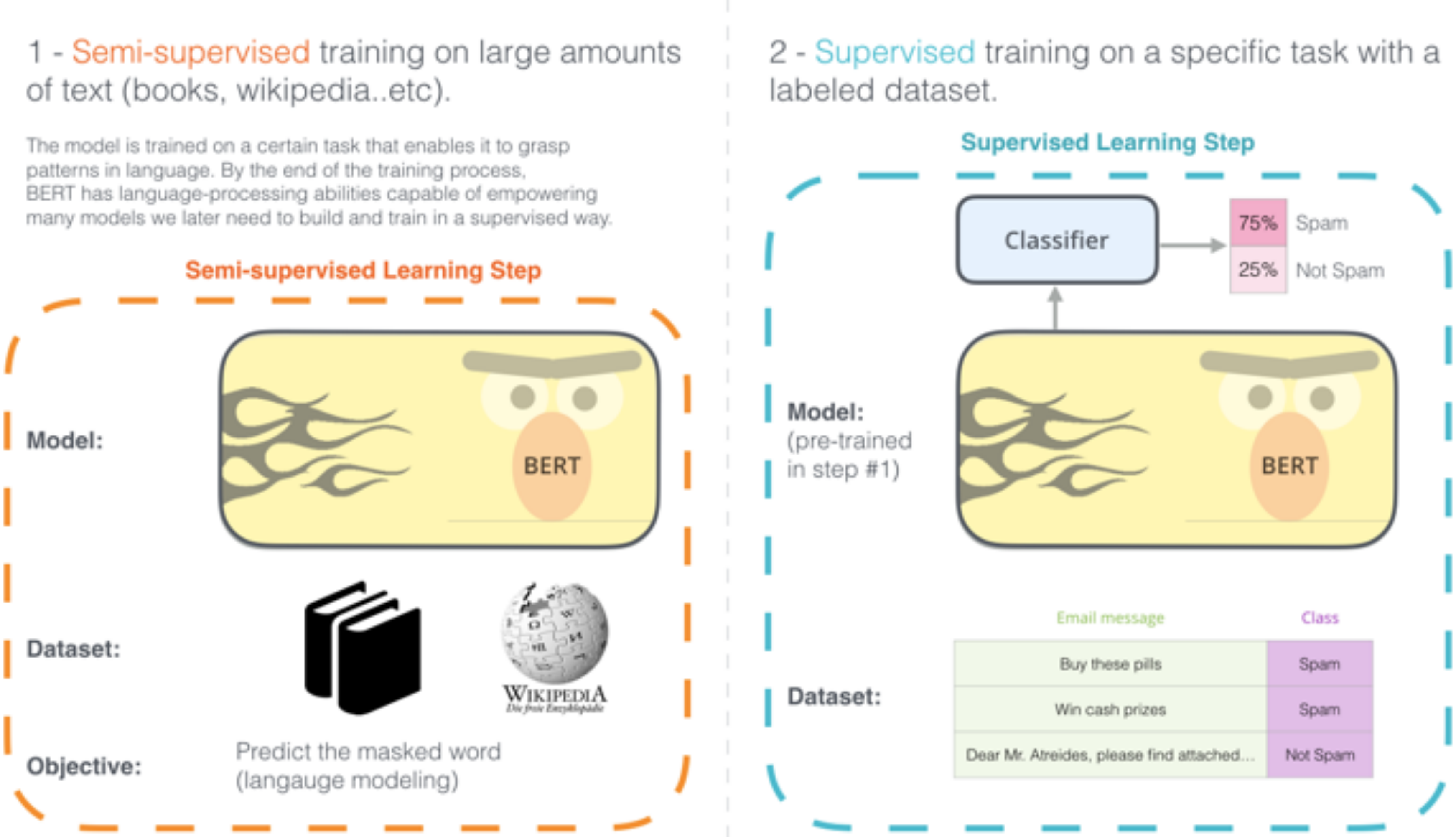
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Masked Language Modeling:
Pretrain a Bidirectional Model



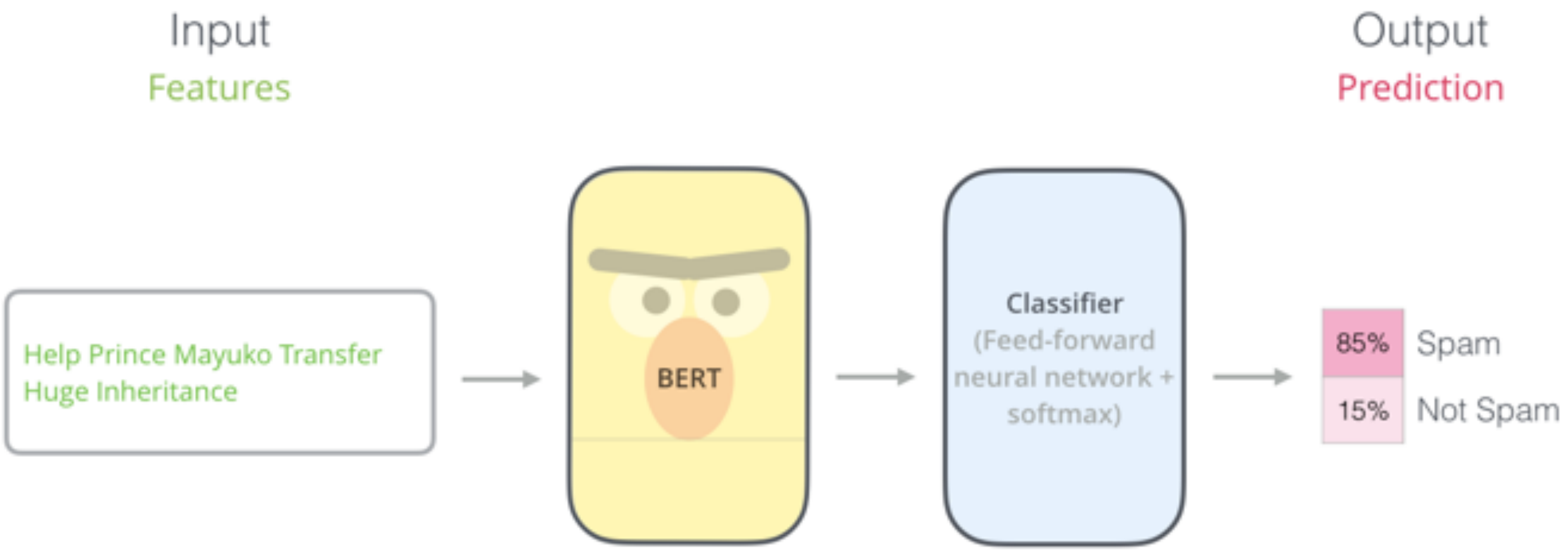
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Illustrated BERT



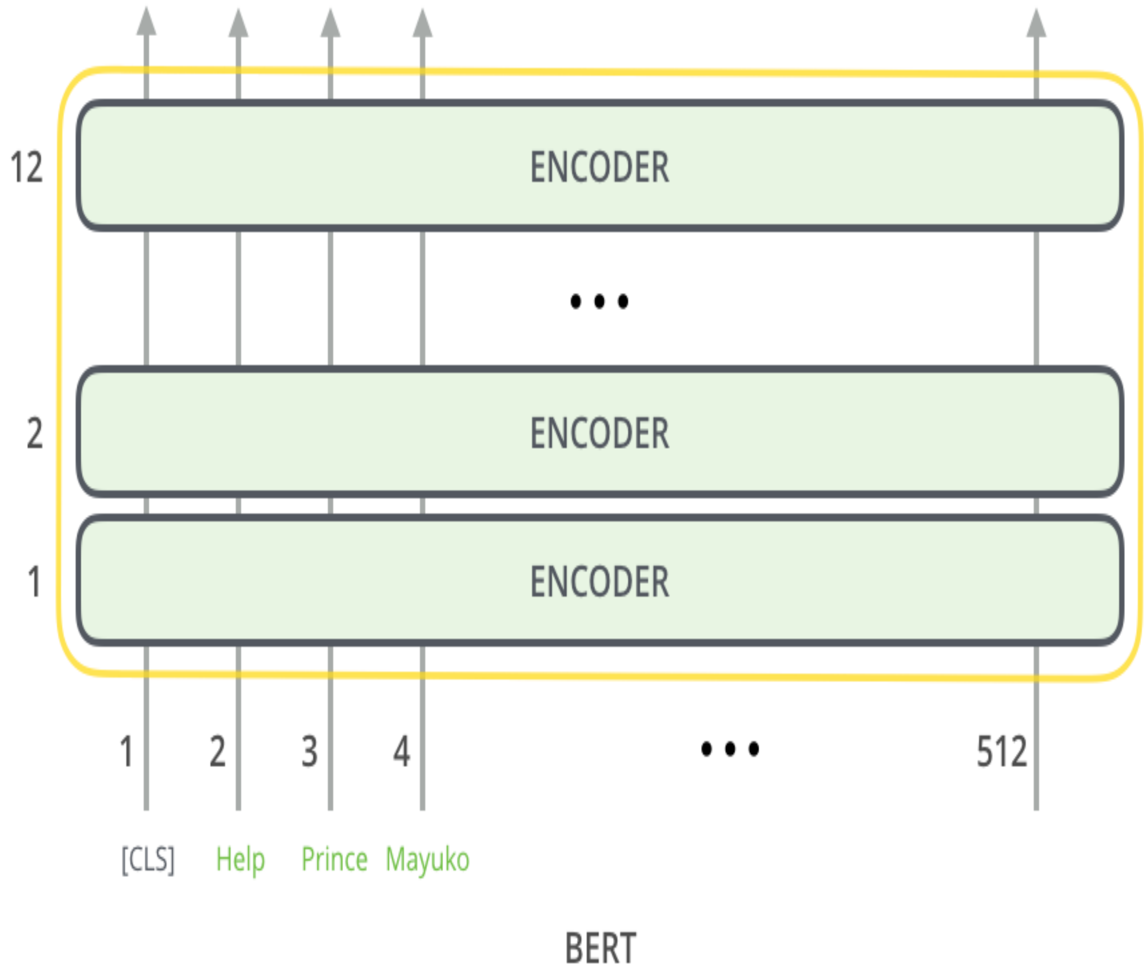
Source: Jay Alammar (2019), The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning), http://jalammar.github.io/illustrated-bert/

BERT Classification Input Output



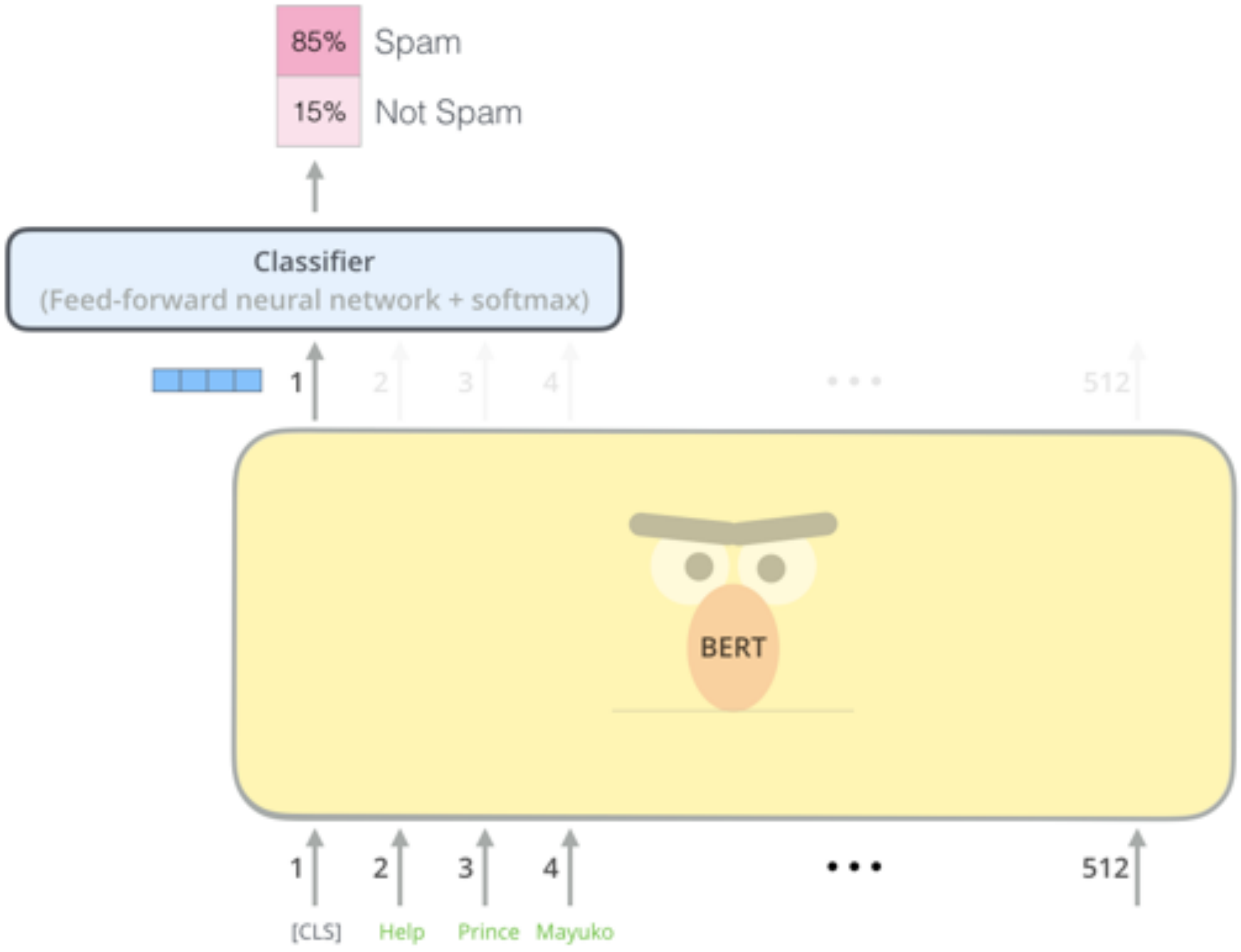
Source: Jay Alammar (2019), The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning),
http://jalammar.github.io/illustrated-bert/

BERT Encoder Input



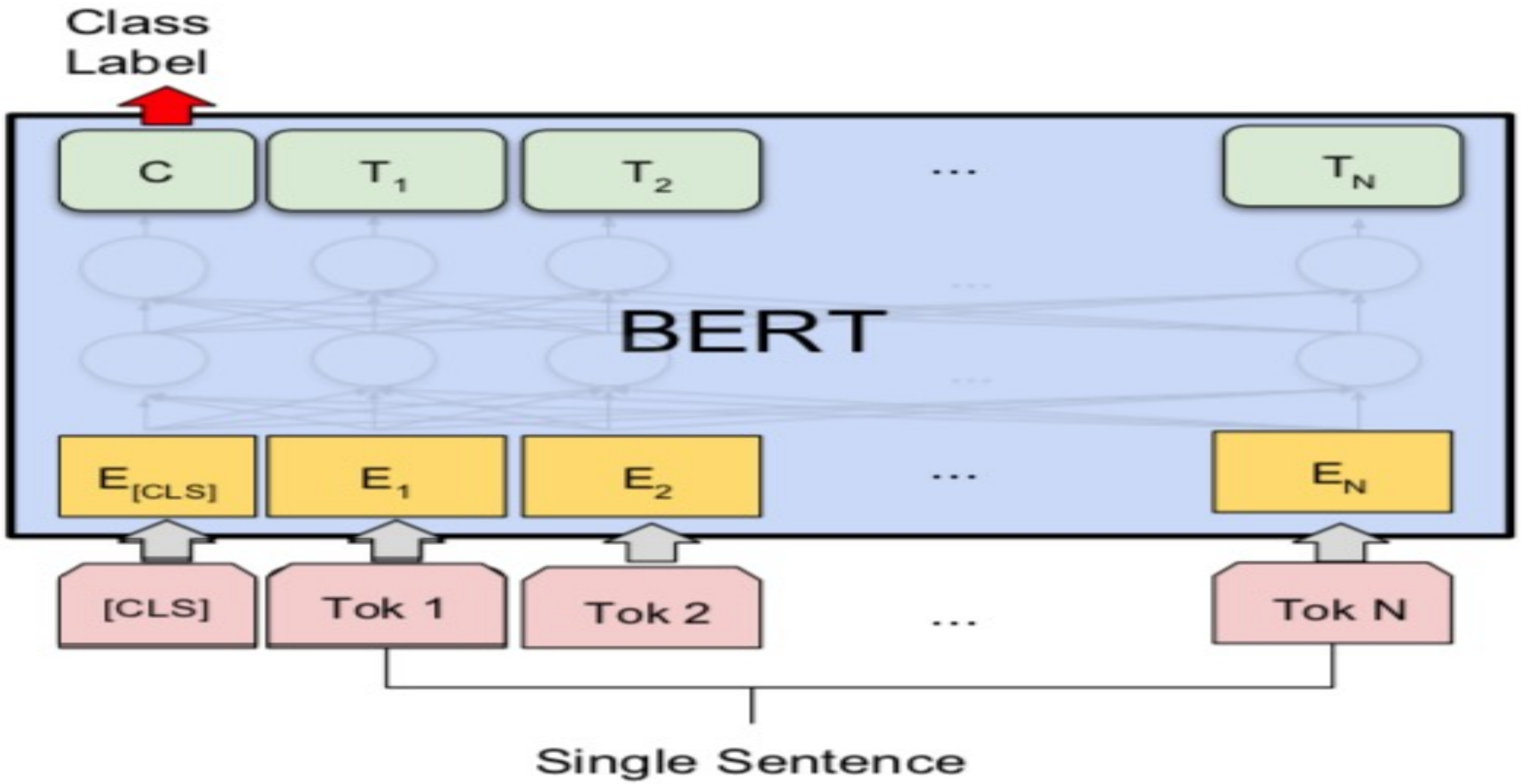
Source: Jay Alammar (2019), The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning), http://jalammar.github.io/illustrated-bert/

BERT Classifier



Source: Jay Alammar (2019), The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning), http://jalammar.github.io/illustrated-bert/

Sentiment Analysis:
Single Sentence Classification



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

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

A Visual Guide to Using BERT for the First Time

(Jay Alammar, 2019)

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

Sentiment Classification: SST2 Sentences from movie reviews

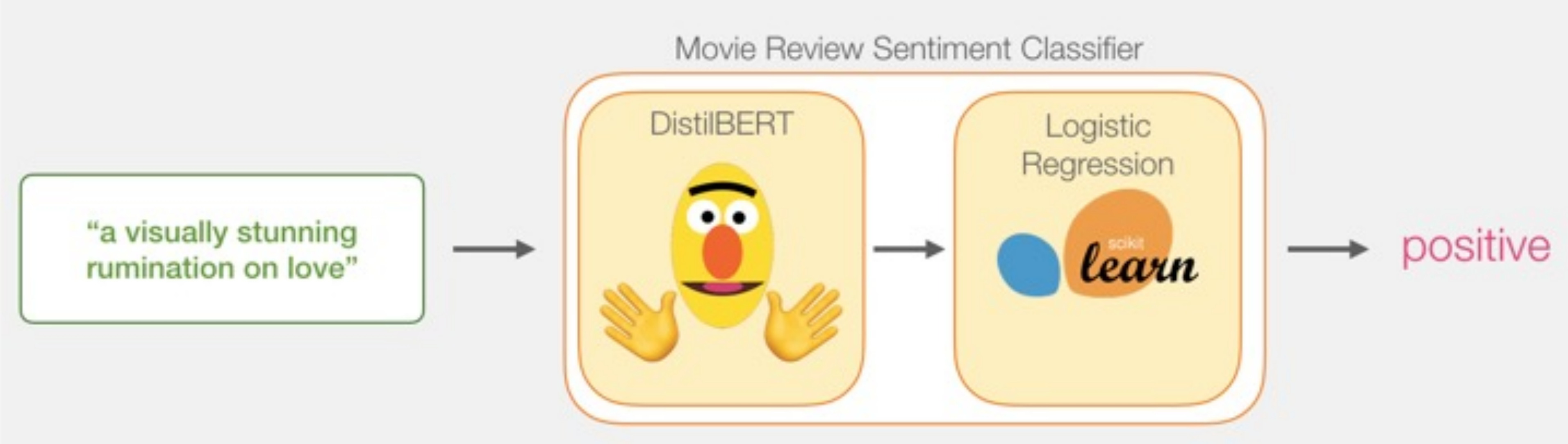
sentence	label
a stirring , funny and finally transporting re imagining of beauty and the beast and 1930s horror films	1
apparently reassembled from the cutting room floor of any given daytime soap	0
they presume their audience won't sit still for a sociology lesson	0
this is a visually stunning rumination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

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

Movie Review Sentiment Classifier

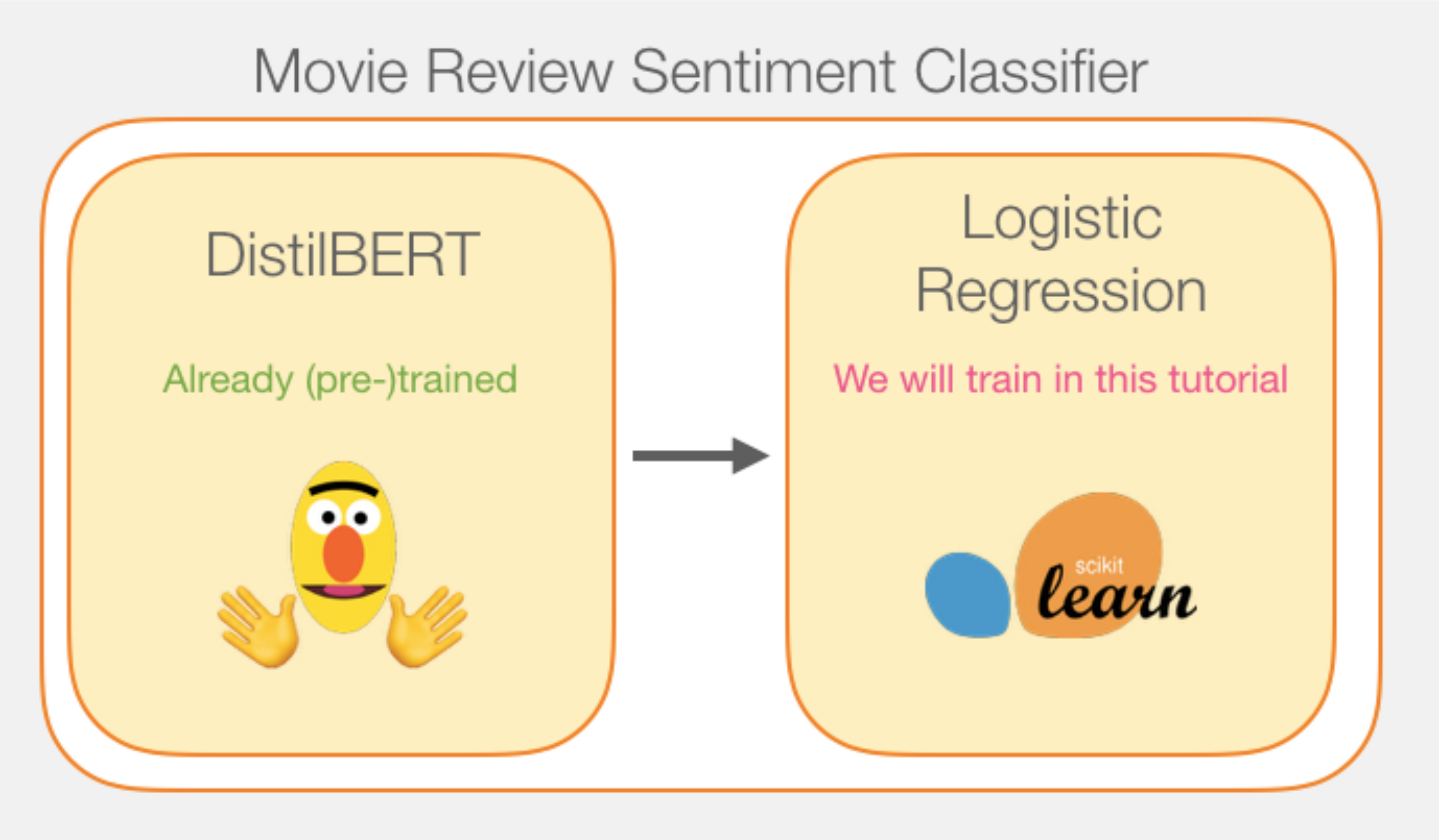
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Movie Review Sentiment Classifier



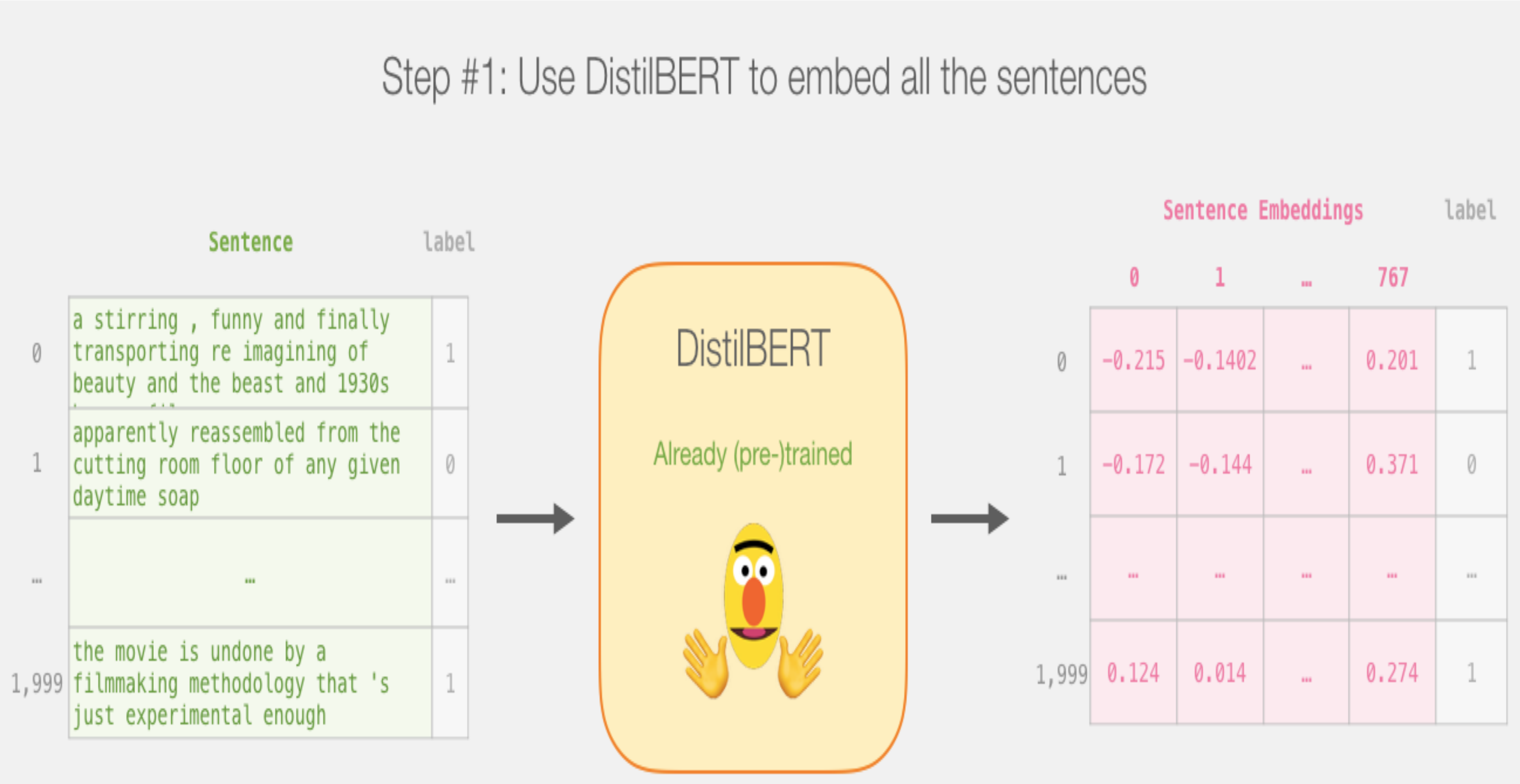
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Movie Review Sentiment Classifier Model Training



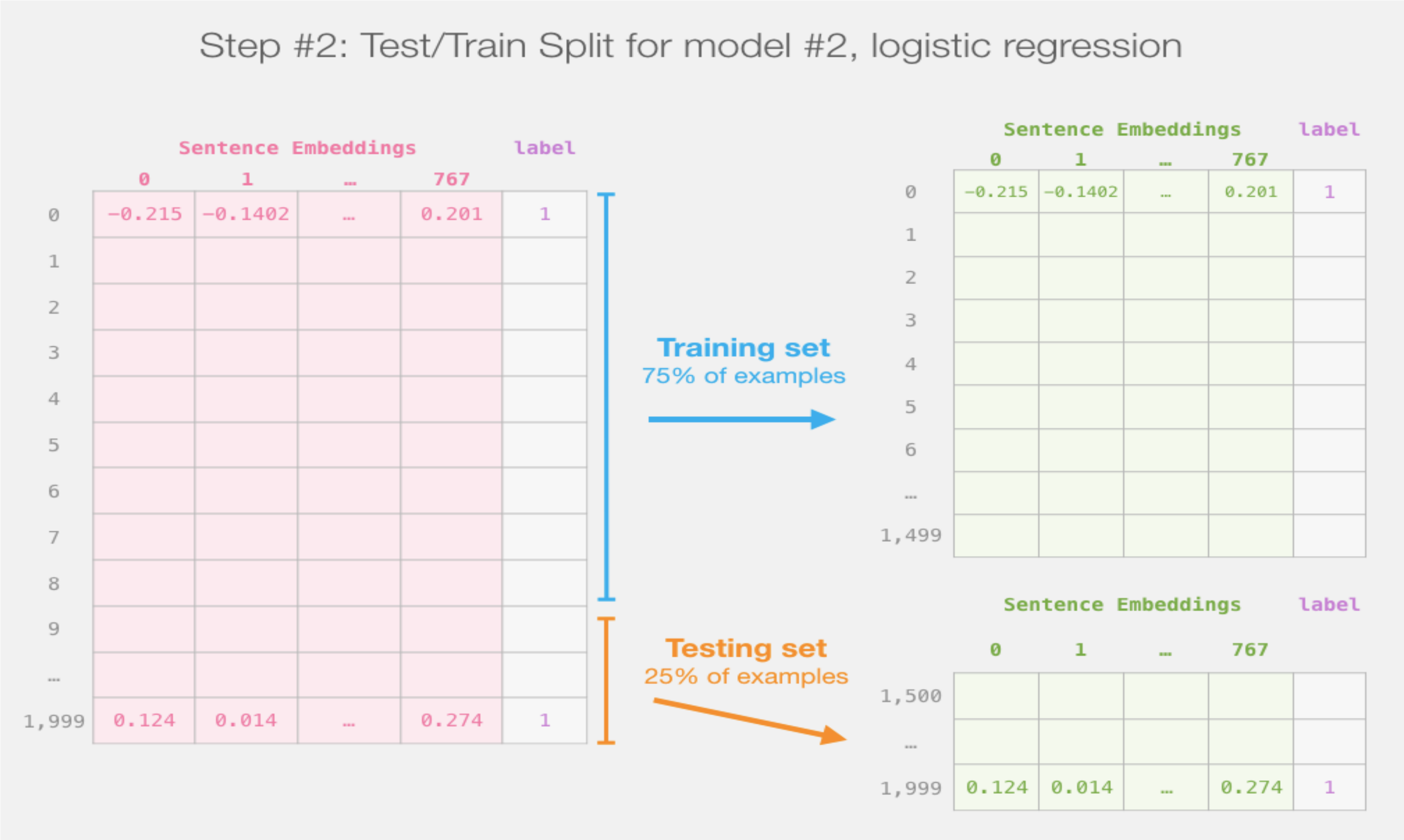
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Step # 1 Use distilBERT to Generate Sentence Embeddings



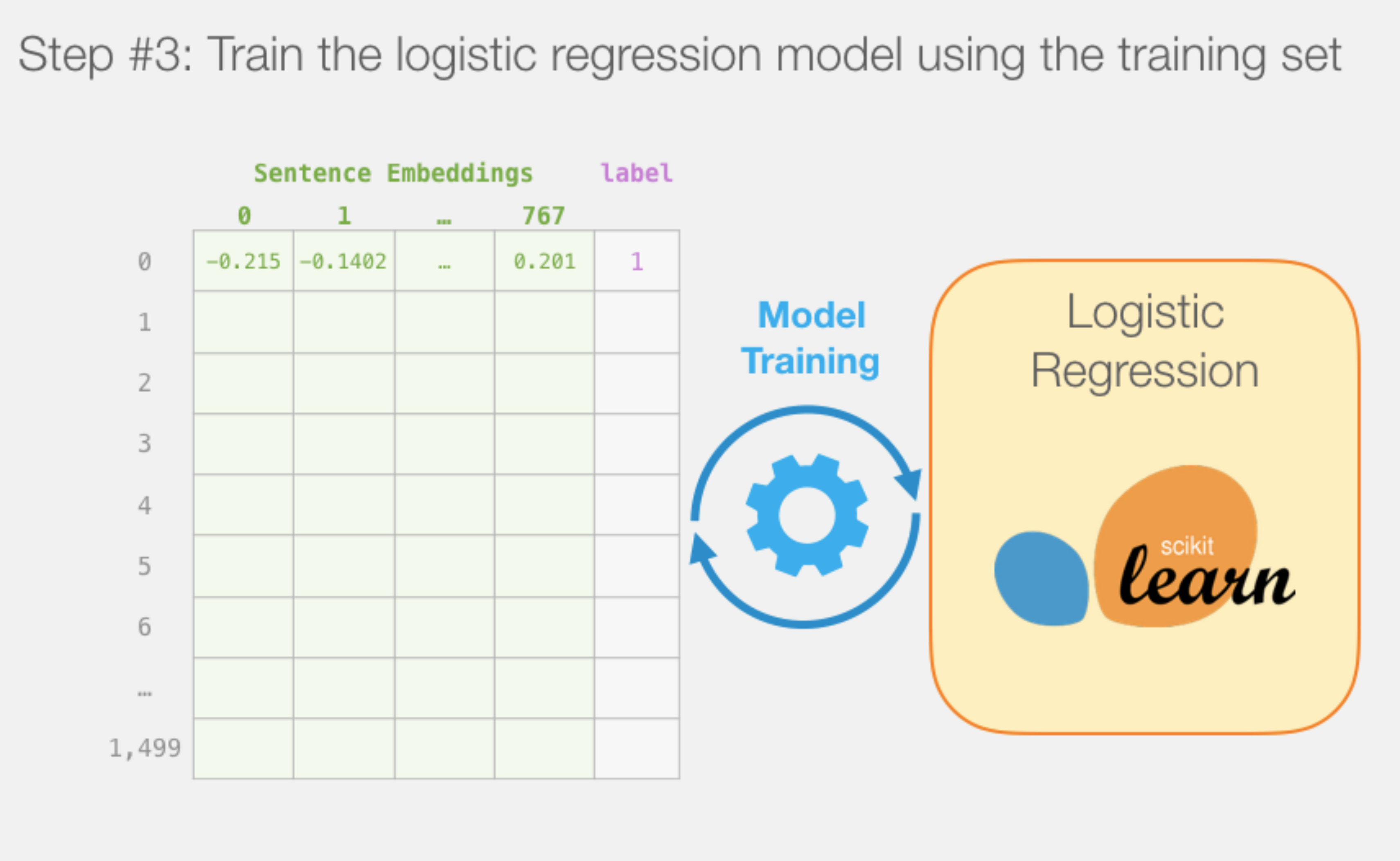
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

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



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

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

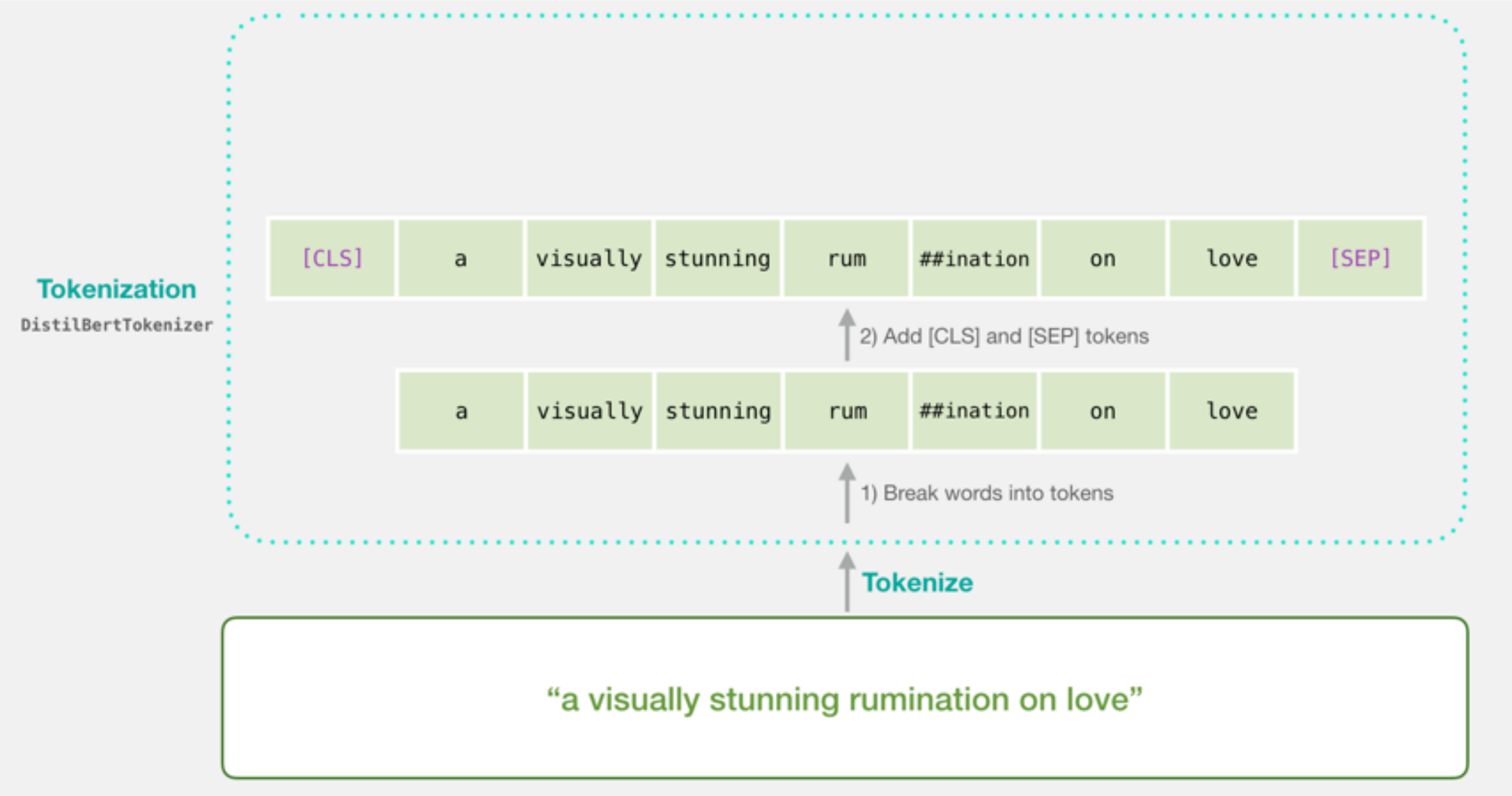


Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Tokenization

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

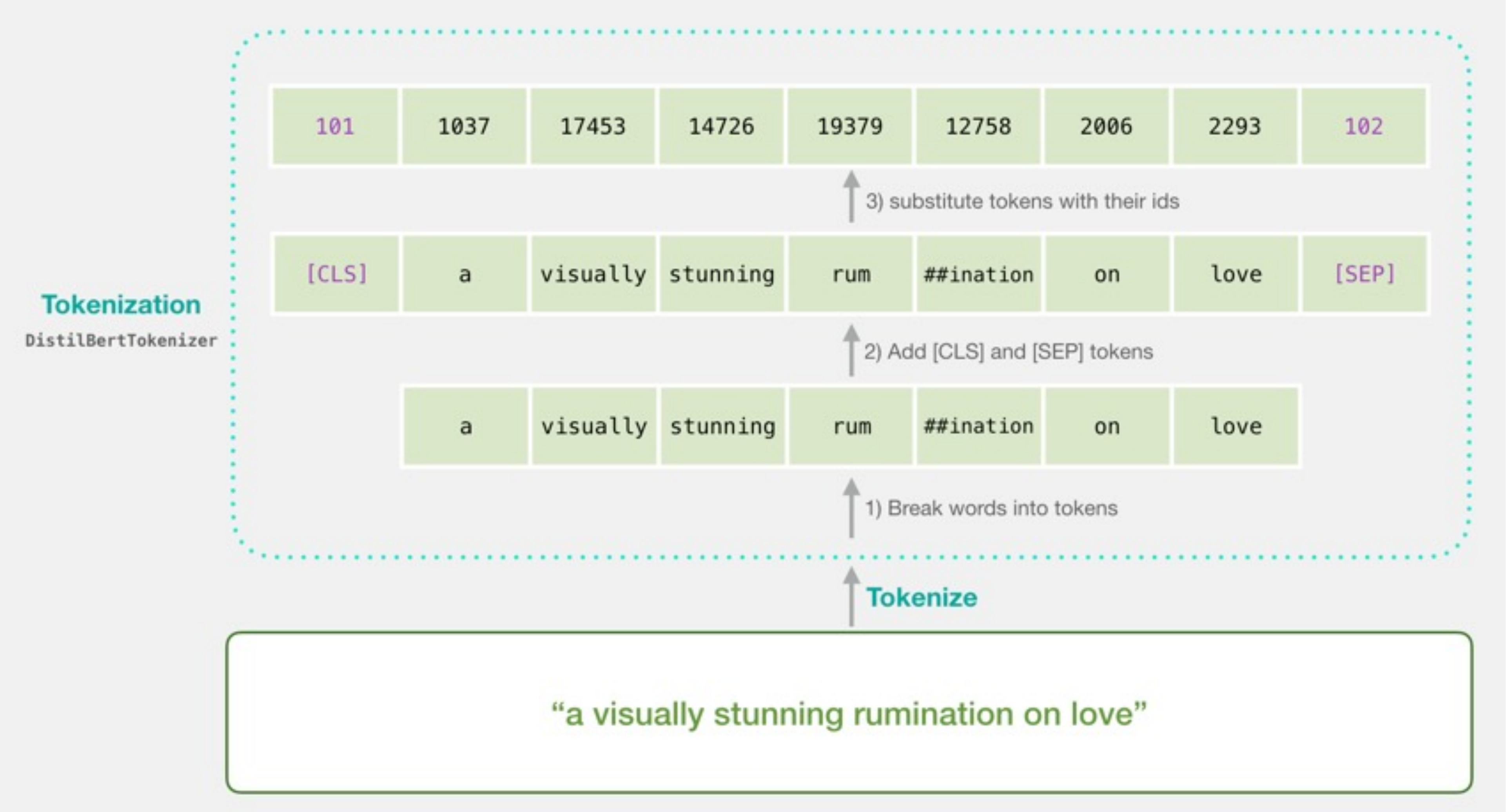
a visually stunning rumination on love



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

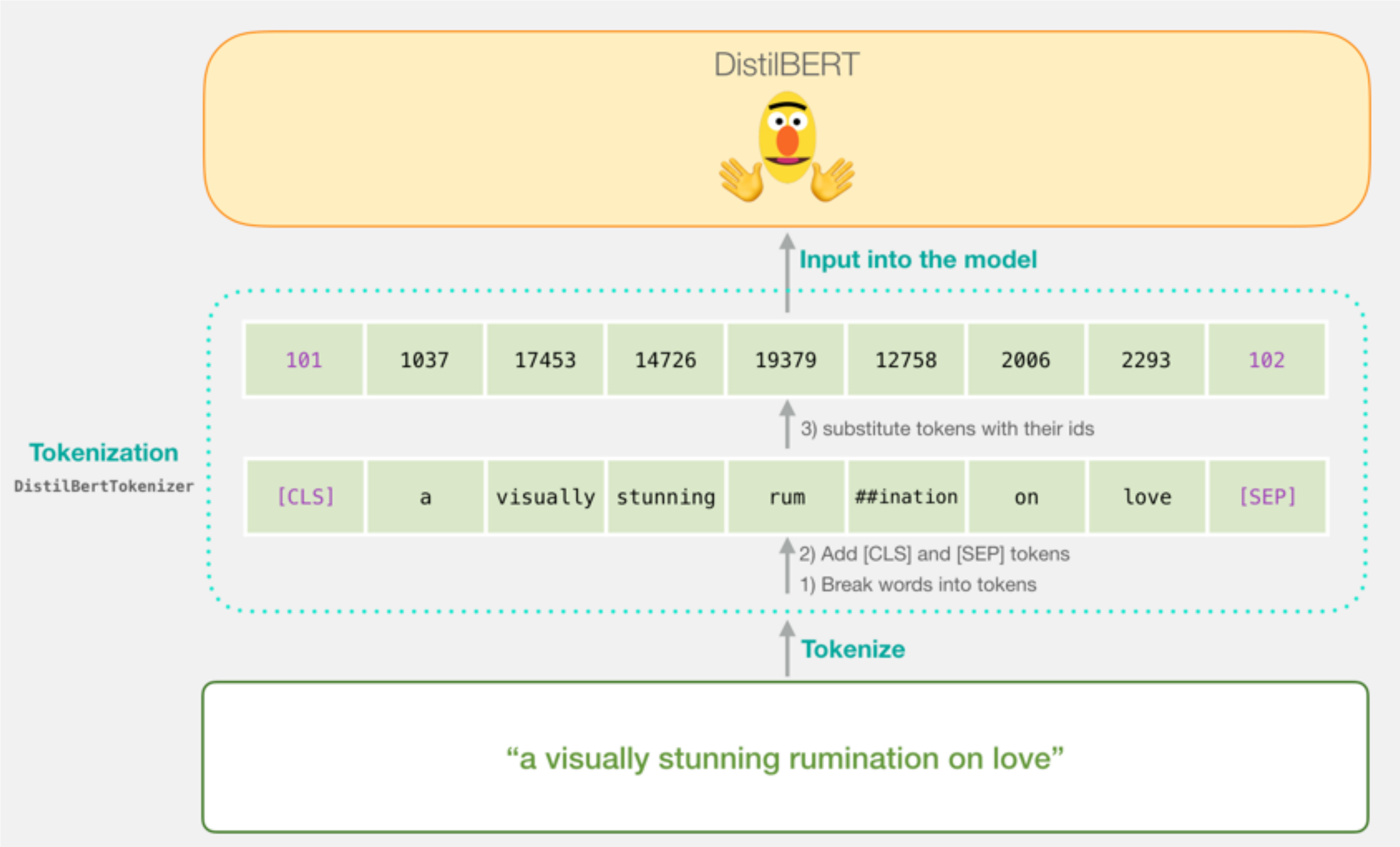
Tokenization

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



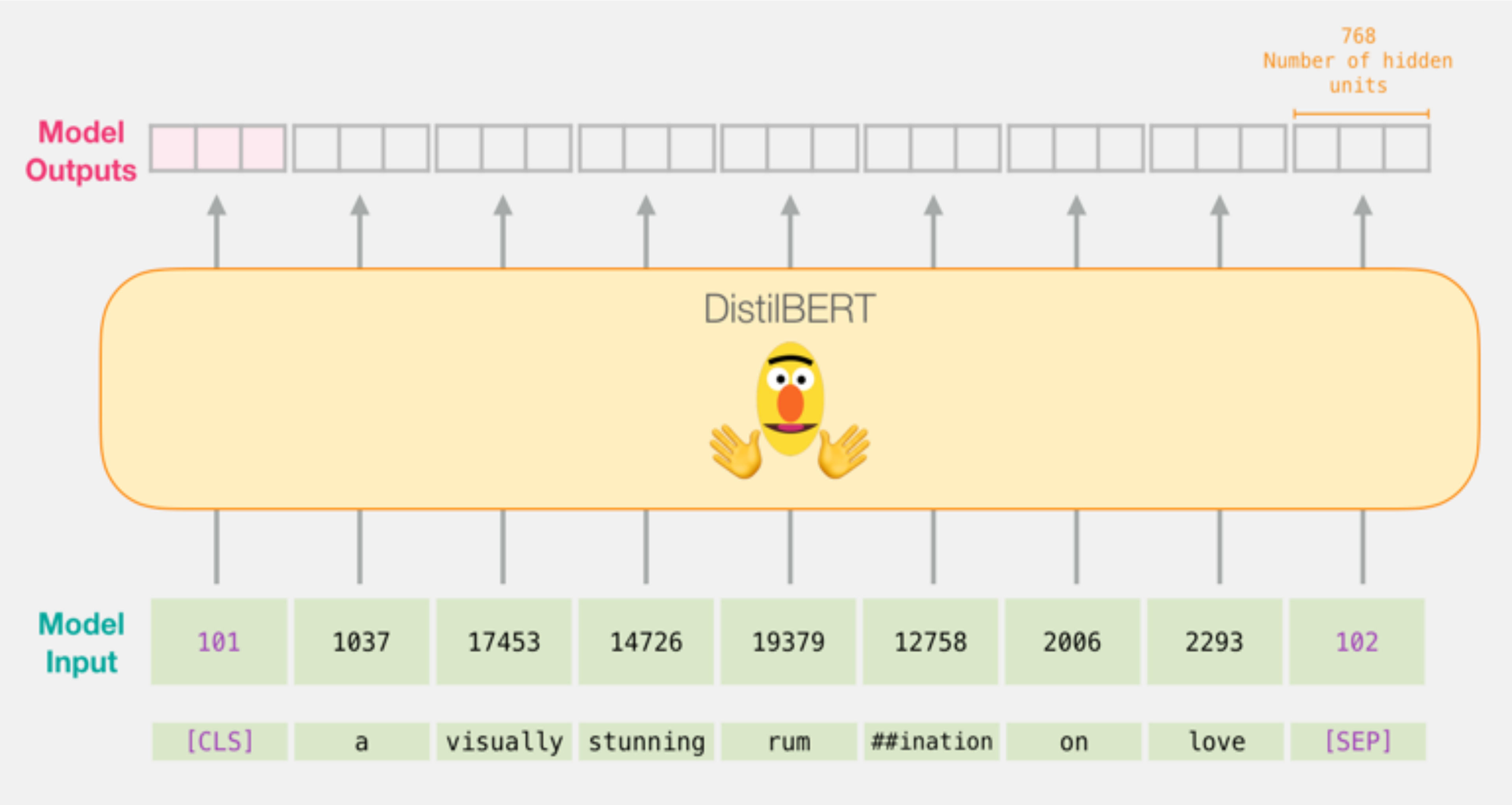
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Tokenization for BERT Model



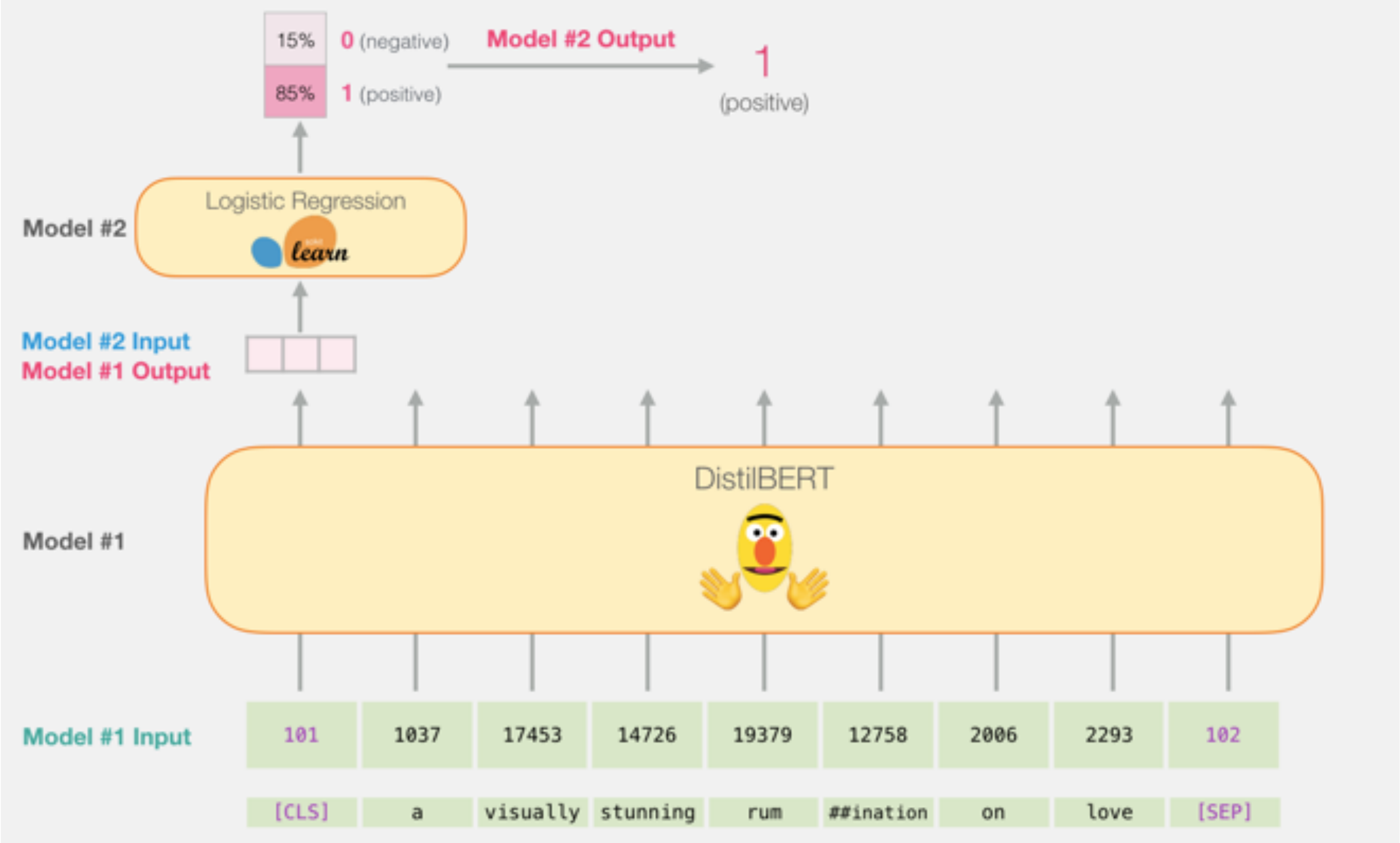
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Flowing Through DistilBERT (768 features)



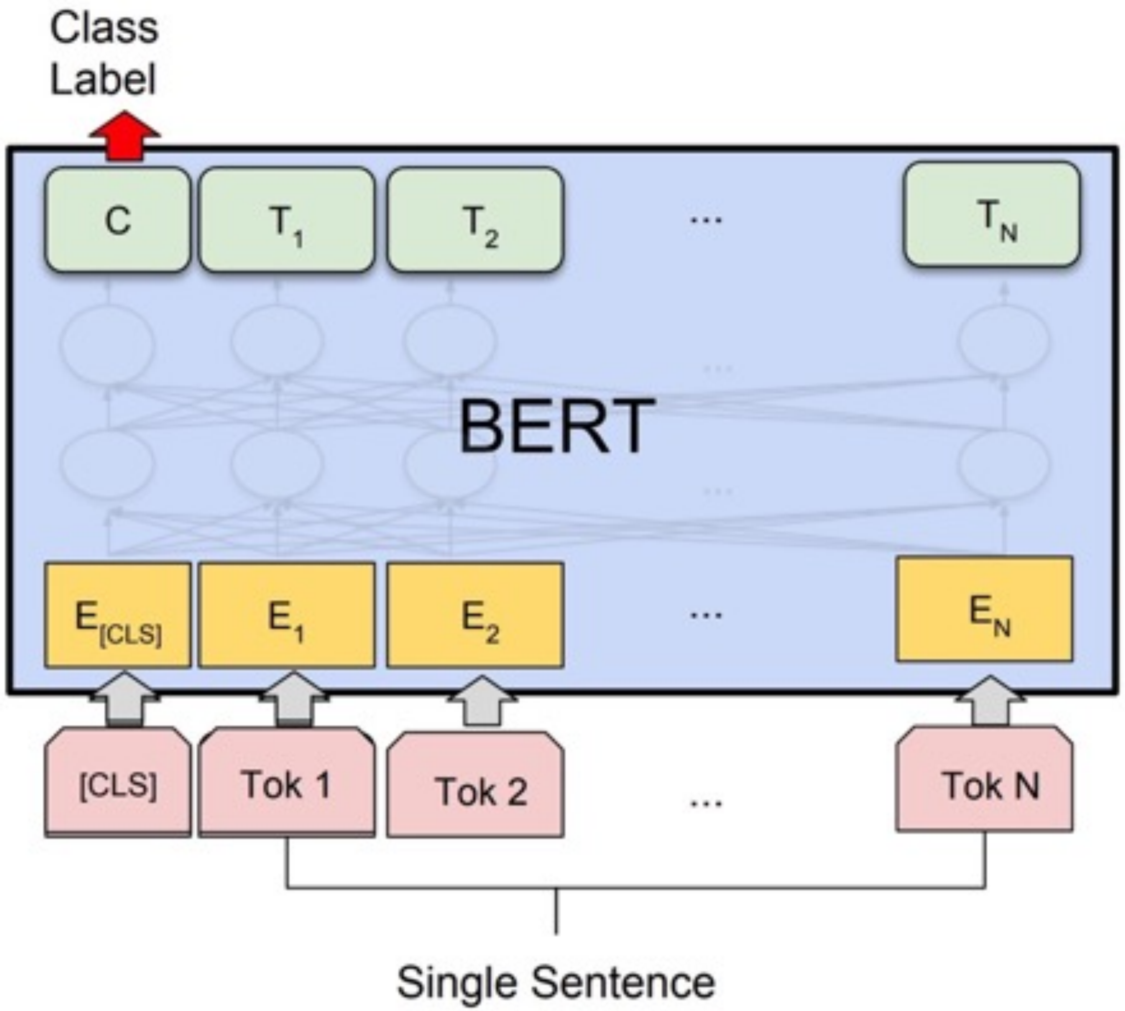
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Model #1 Output Class vector as Model #2 Input



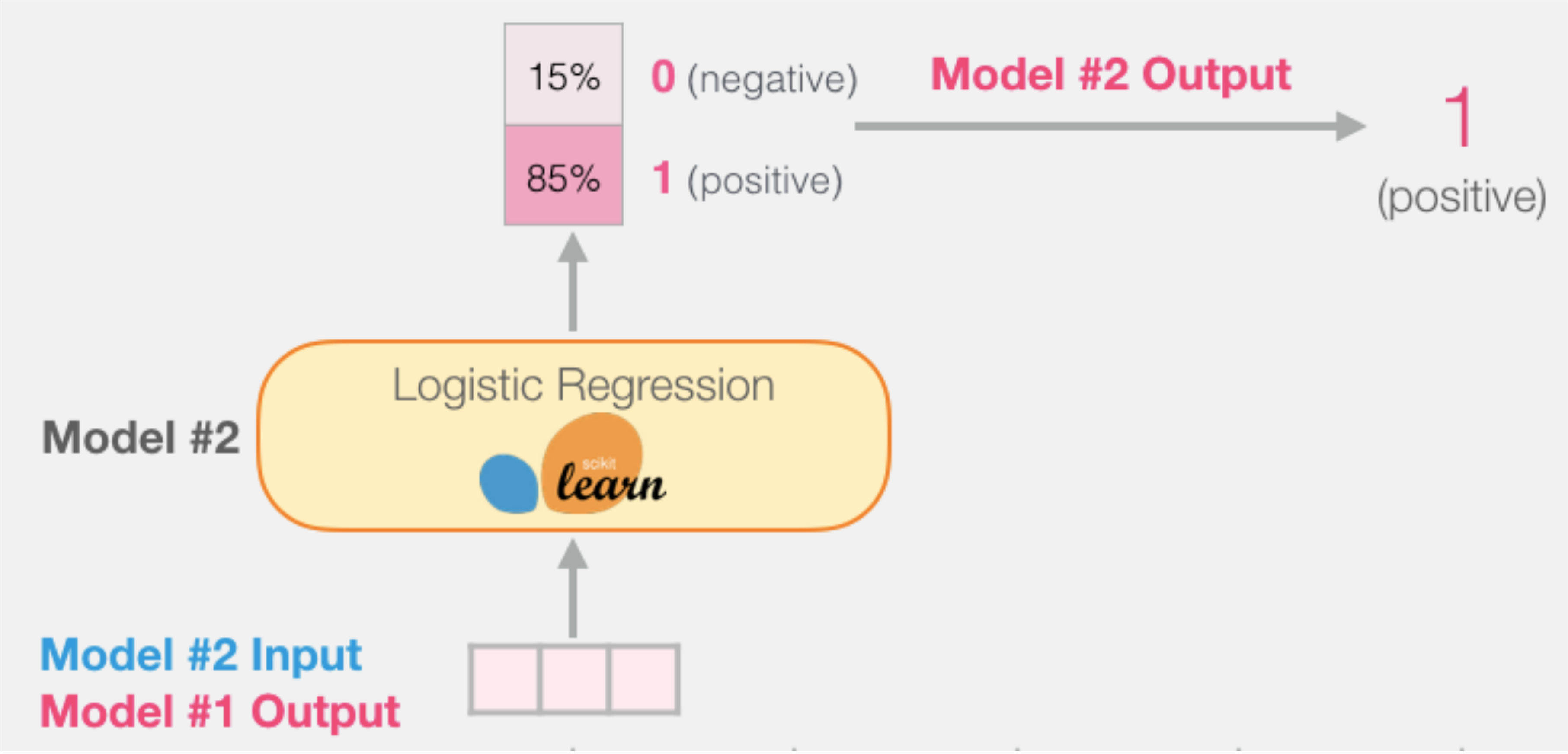
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Fine-tuning BERT on Single Sentence Classification Tasks



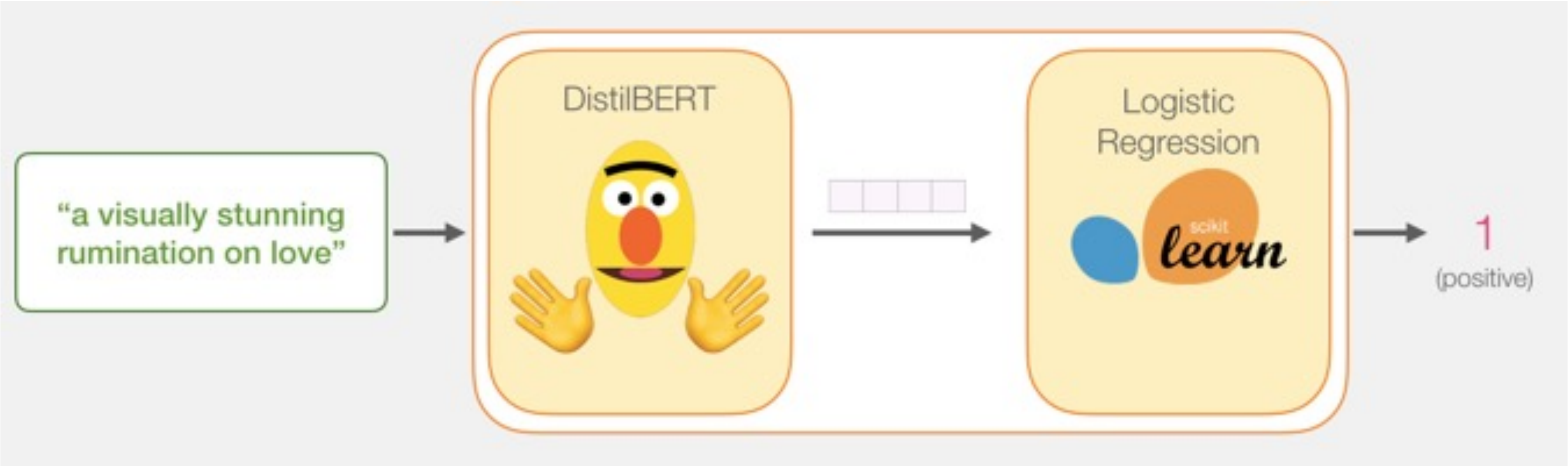
Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

Model #1 Output Class vector as Model #2 Input



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

Logistic Regression Model to classify Class vector



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

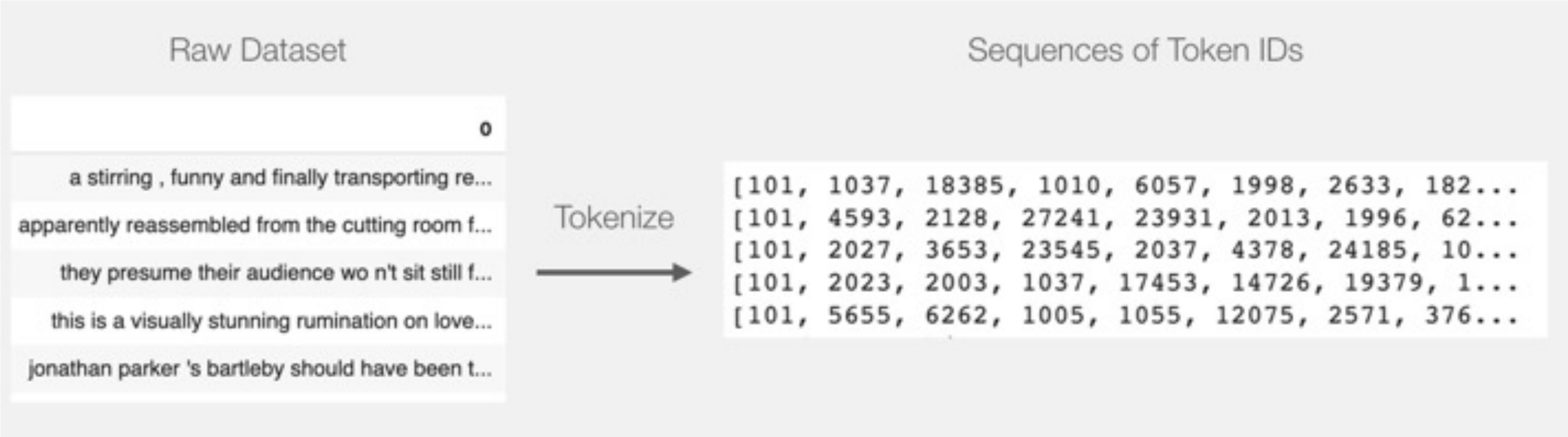
```python
df = pd.read_csv('https://github.com/clairett/pytorch-sentiment-classification/raw/master/data/SST2/train.tsv', delimiter='\t', header=None)

df.head()
```

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

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

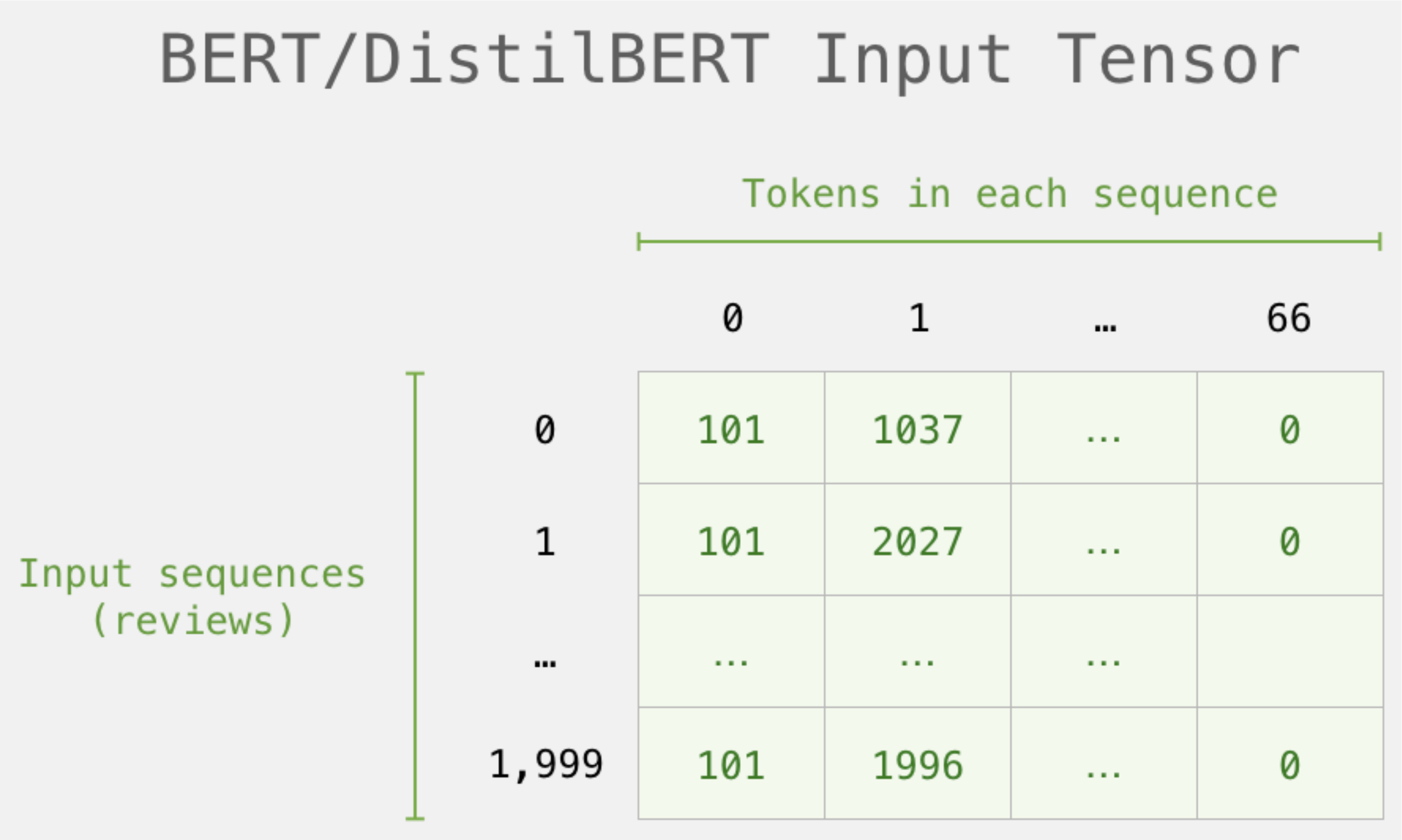
Tokenization

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



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

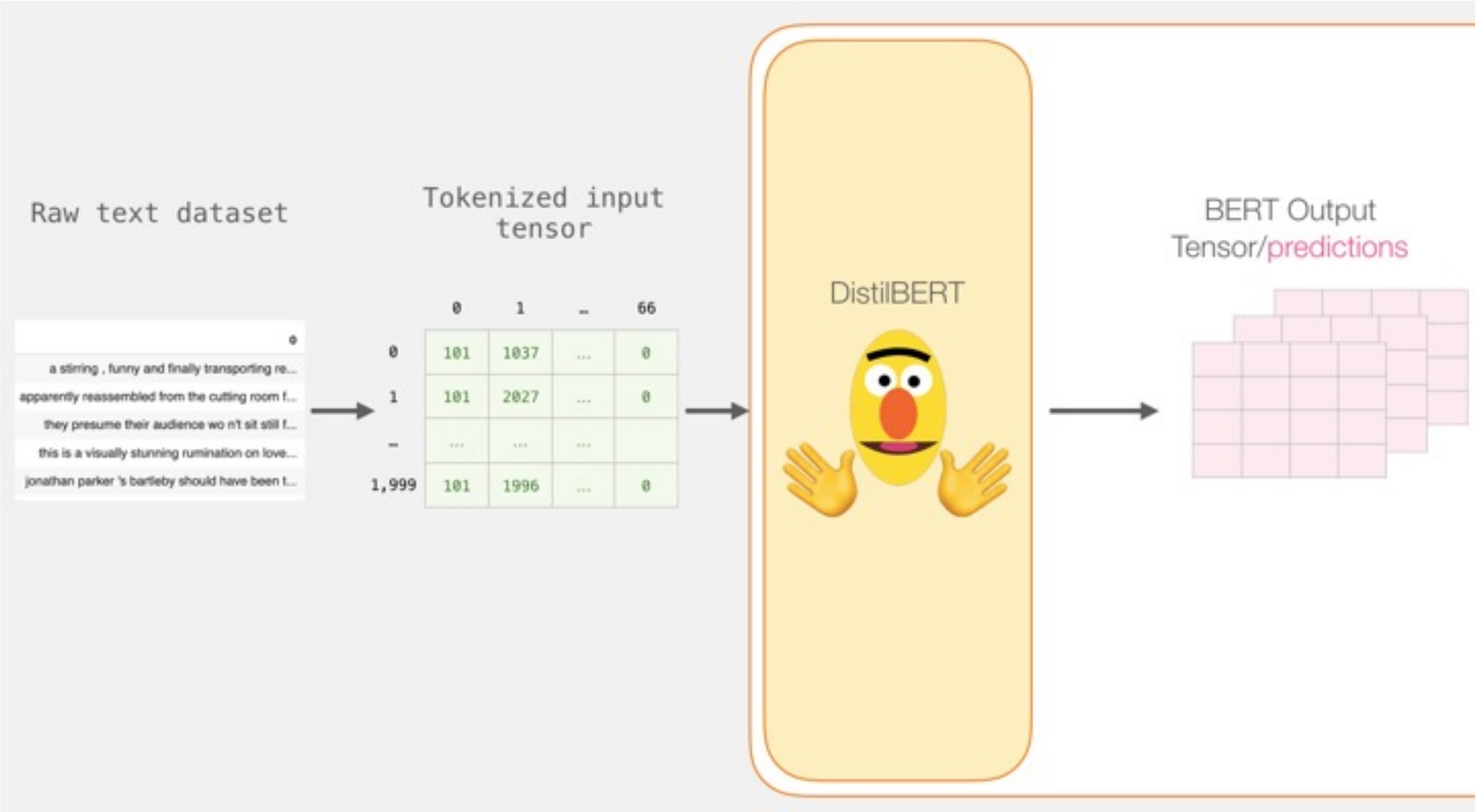
BERT Input Tensor



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

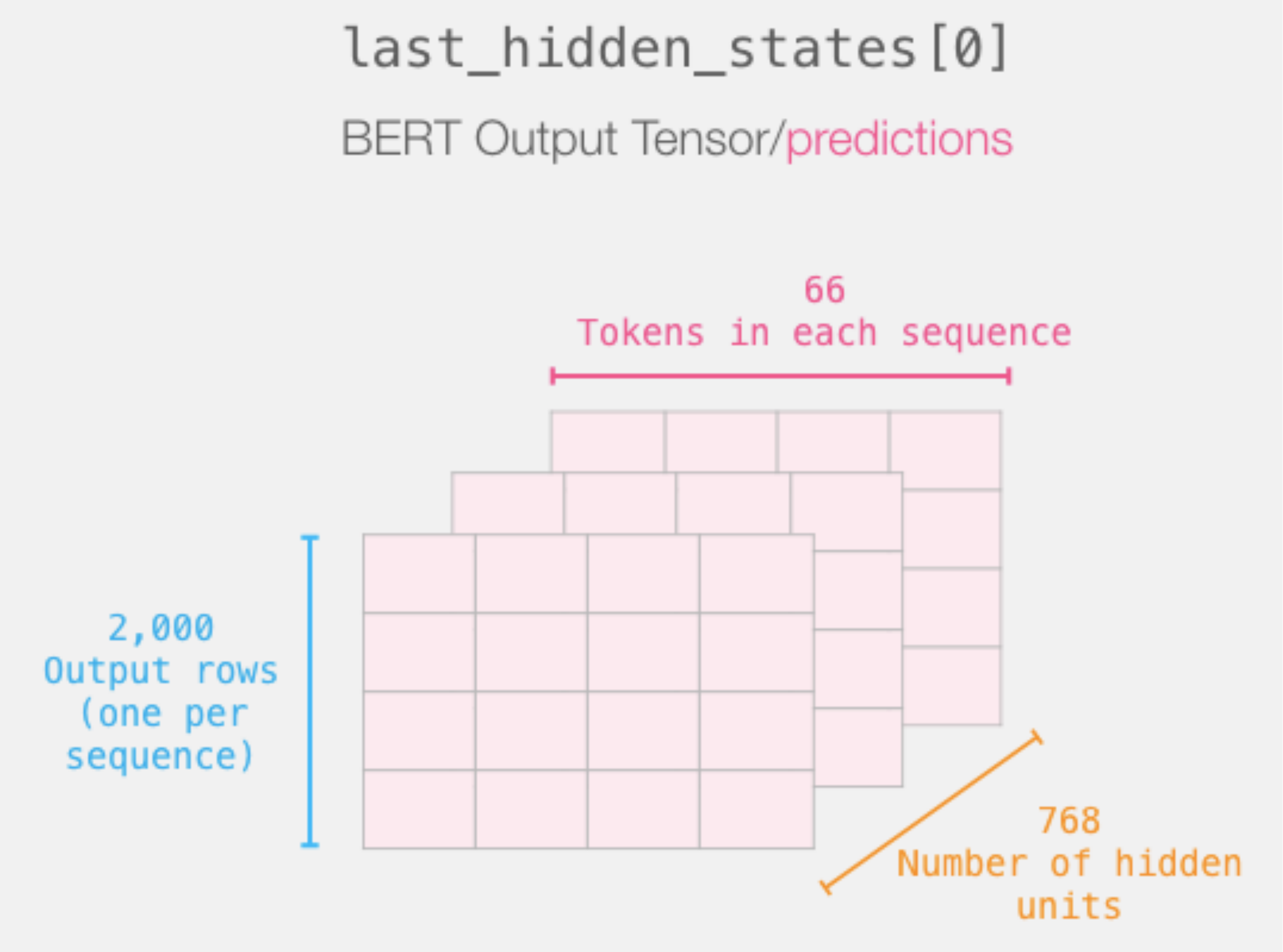
Processing with DistilBERT

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



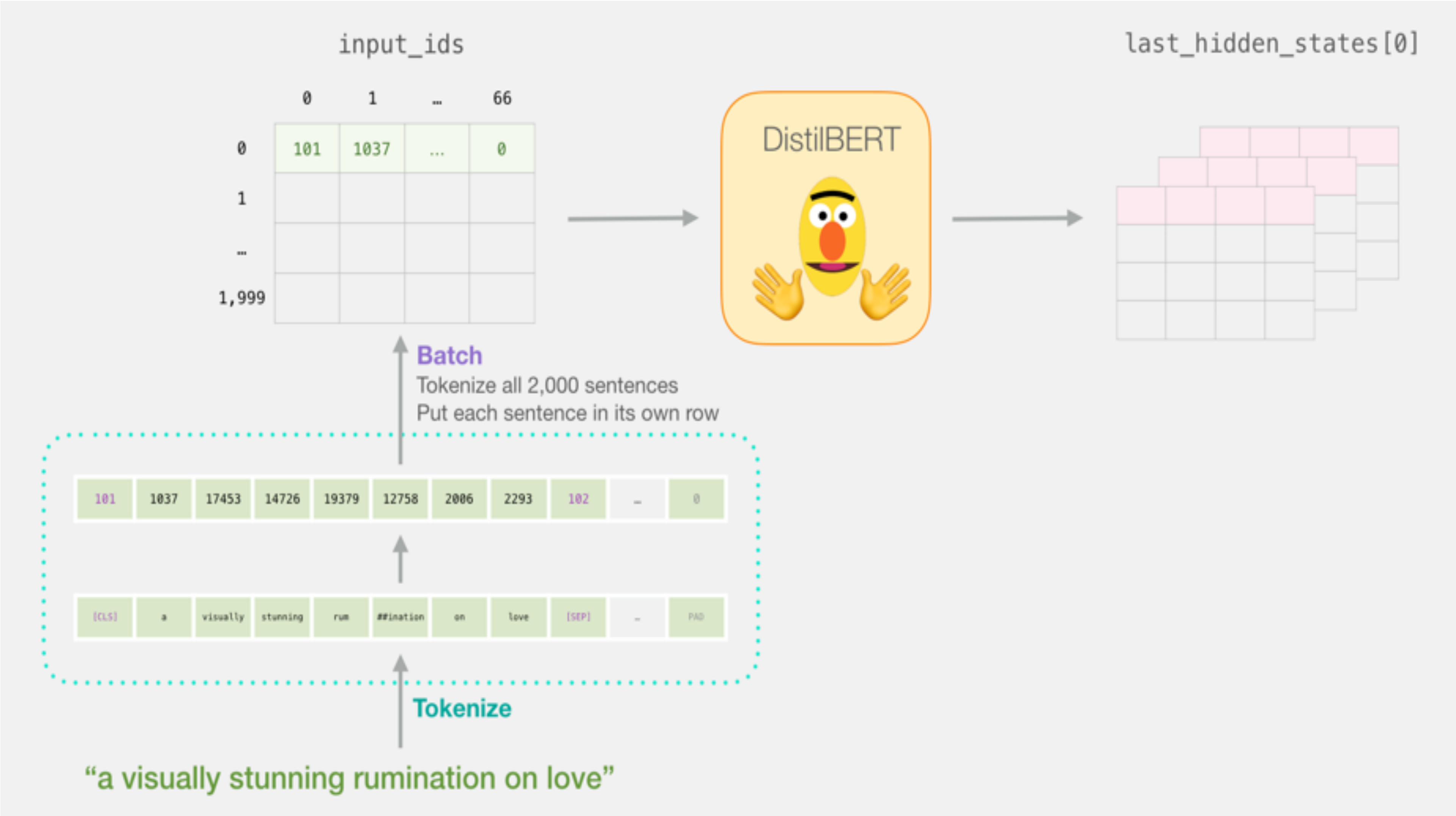
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Unpacking the BERT output tensor



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

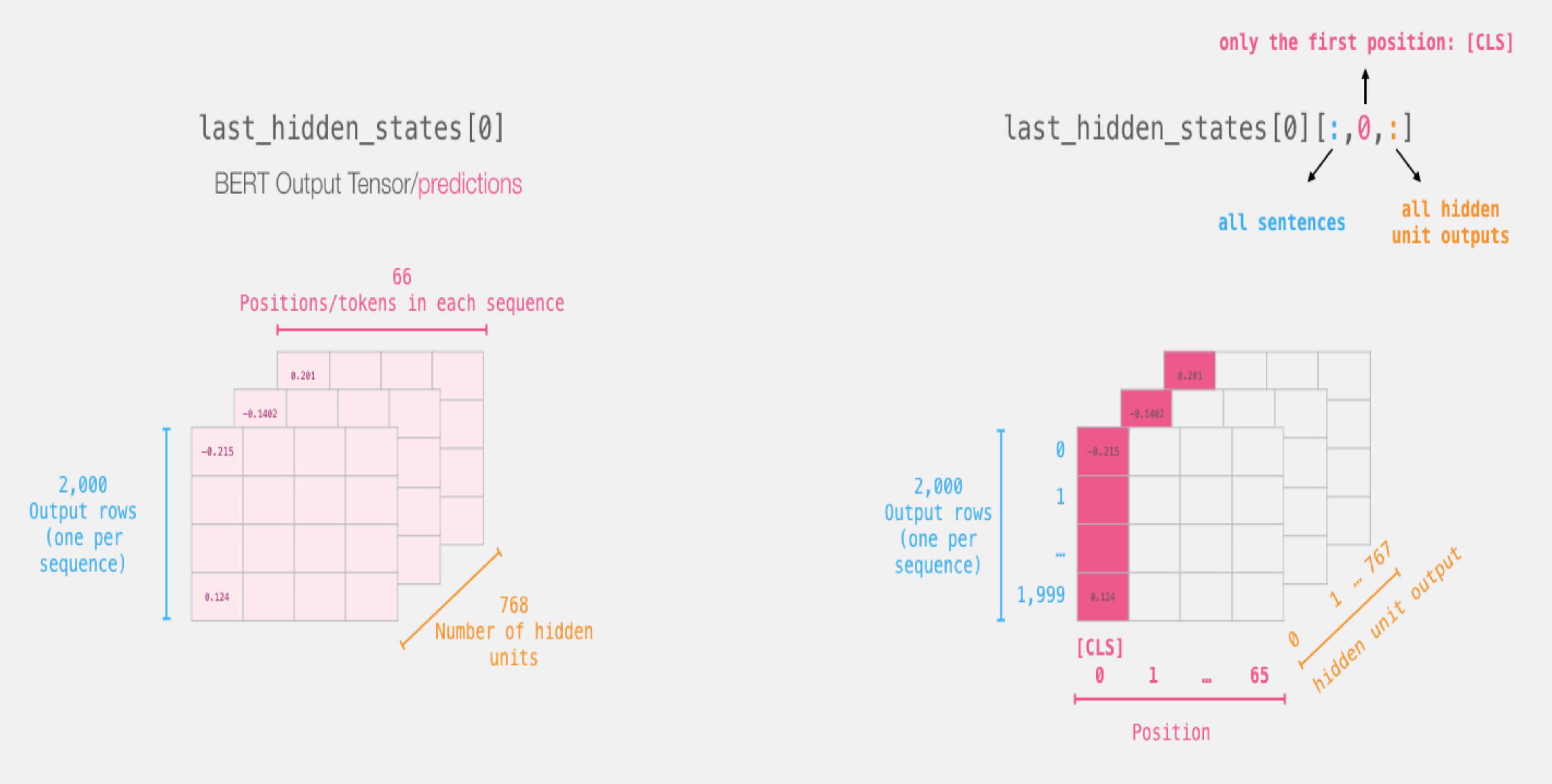
Sentence to last_hidden_state[0]



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

BERT's output for the [CLS] tokens

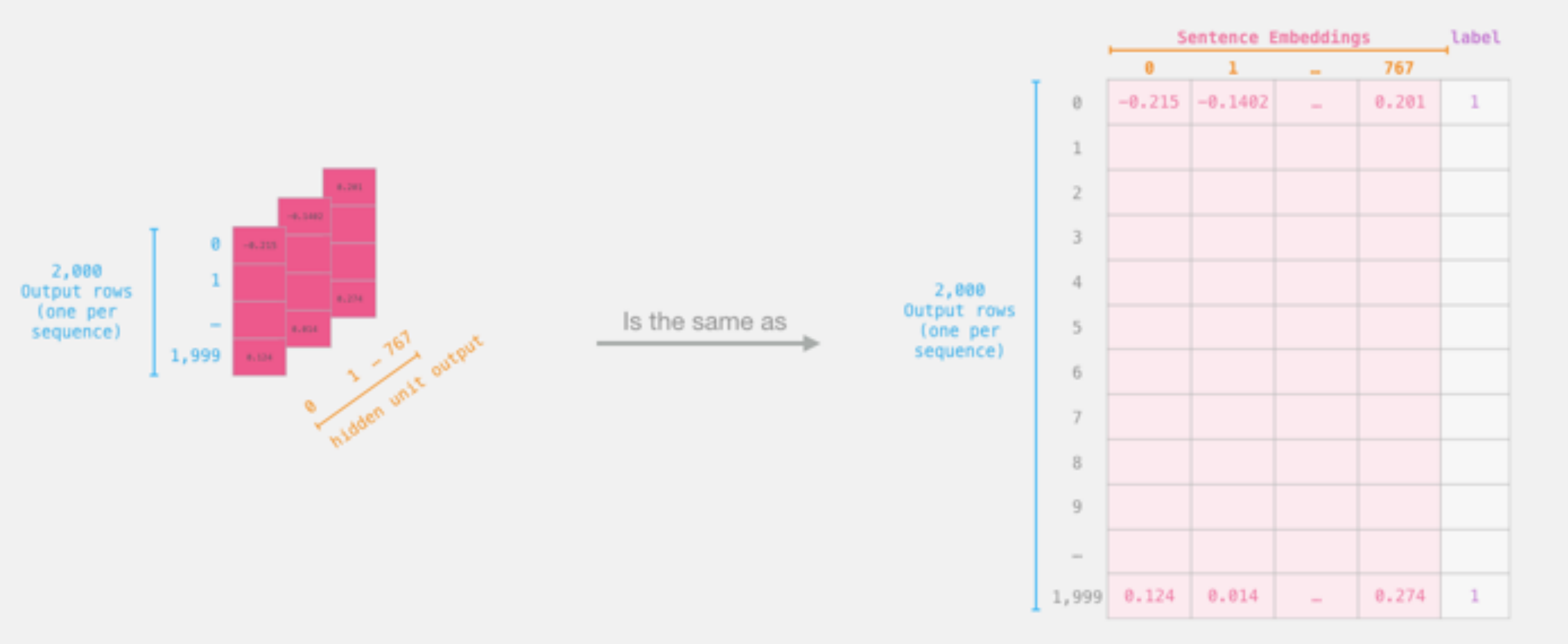
```
# Slice the output for the first position for all the
sequences, take all hidden unit outputs
features = last_hidden_states[0][:,0,:].numpy()
```



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

The tensor sliced from BERT's output

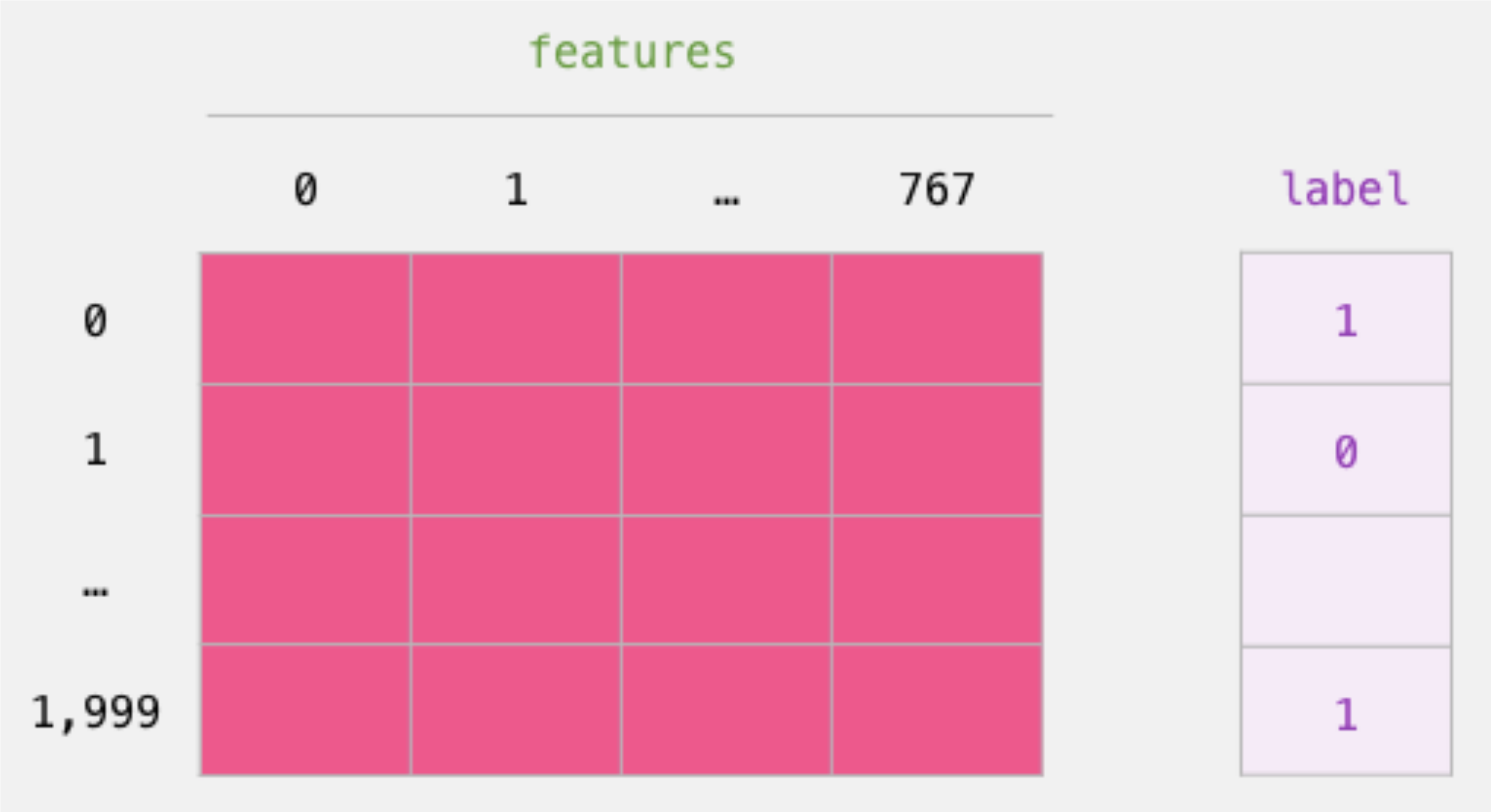
Sentence Embeddings



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

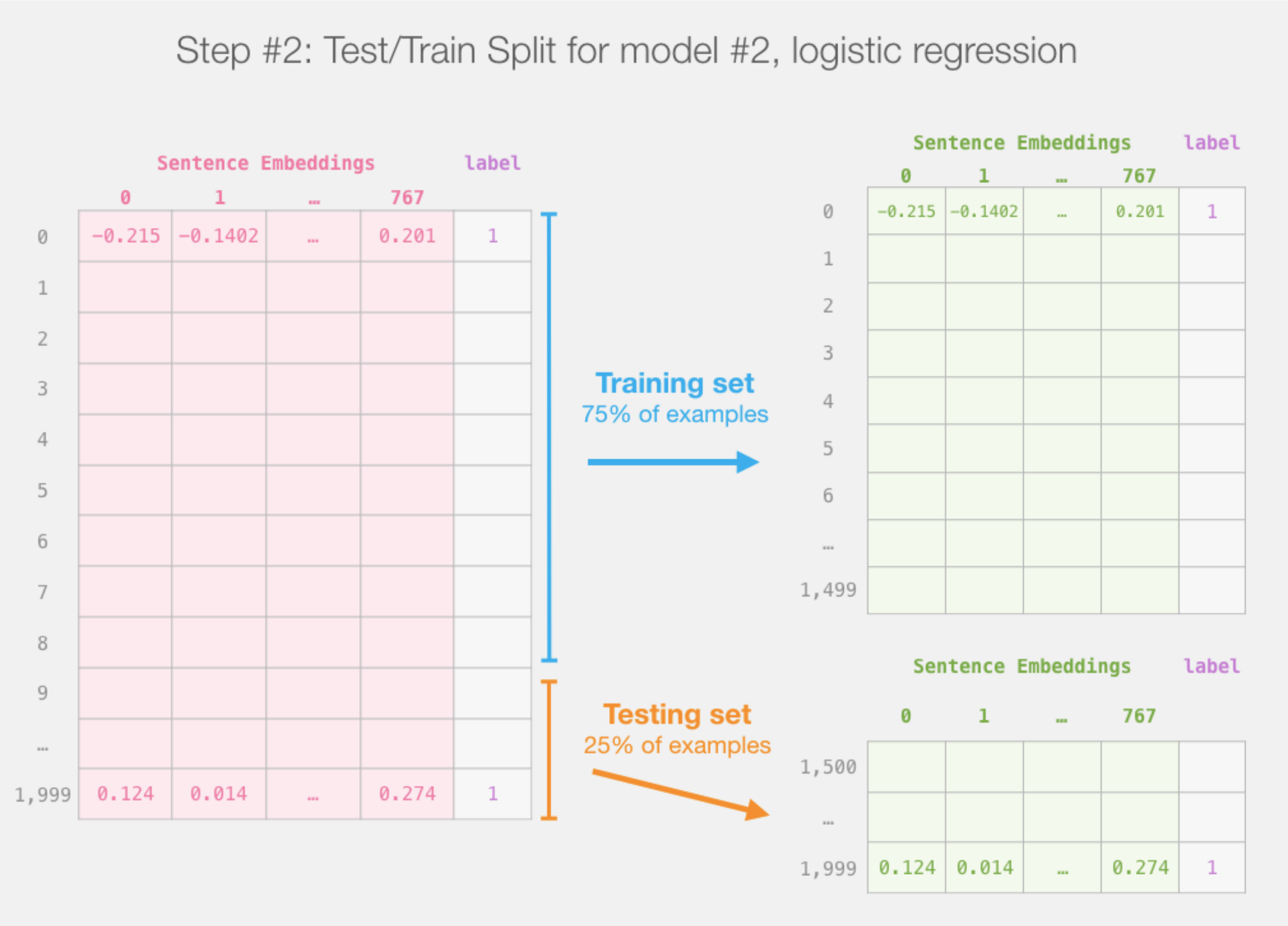
Dataset for Logistic Regression (768 Features)

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



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

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



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

Score Benchmarks
Logistic Regression Model
on SST-2 Dataset

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

#Testing
lr_clf.score(test_features, test_labels)

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

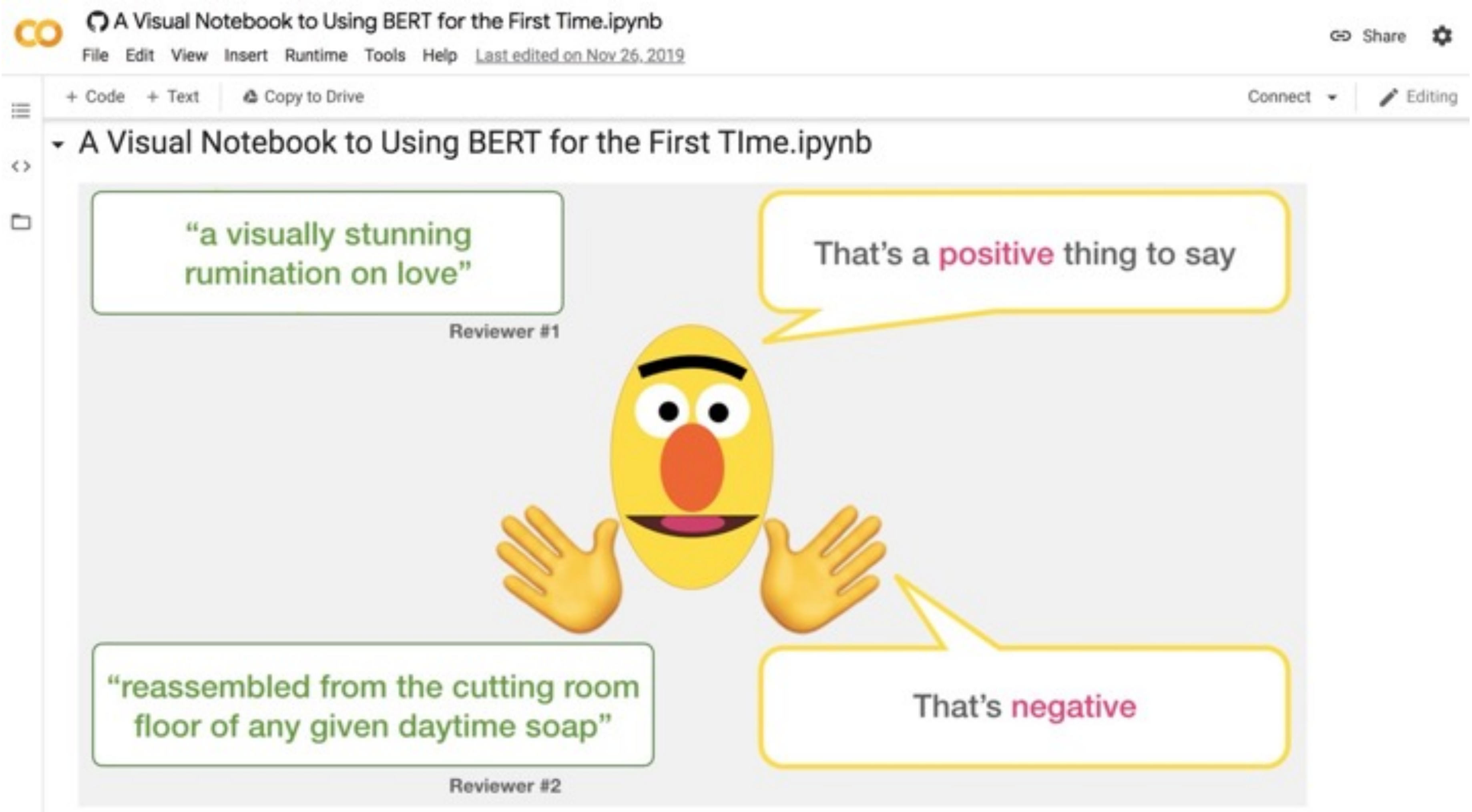
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Sentiment Classification: SST2
Sentences from movie reviews

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

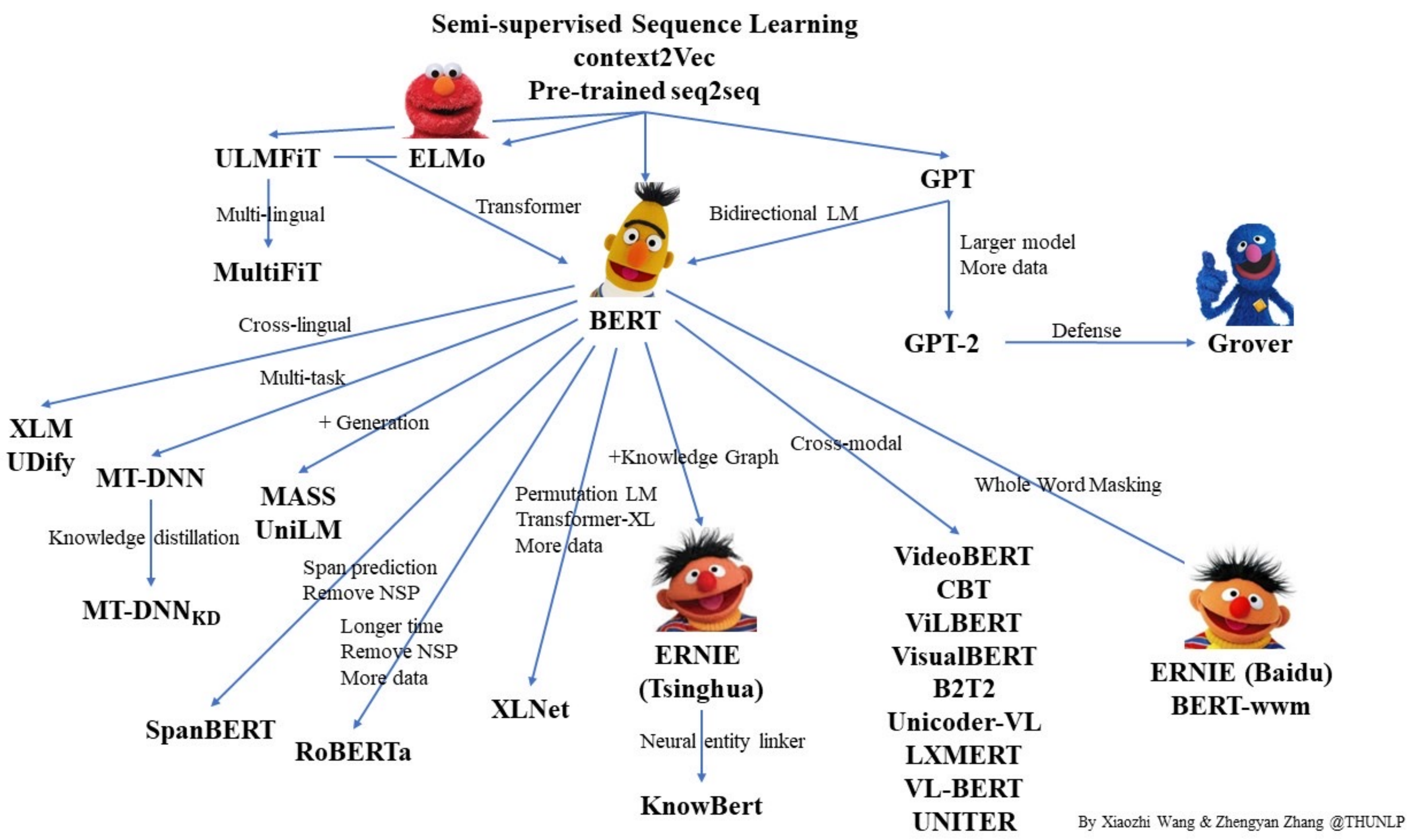
Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

A Visual Notebook to Using BERT for the First Time



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

Pre-trained Language Model (PLM)

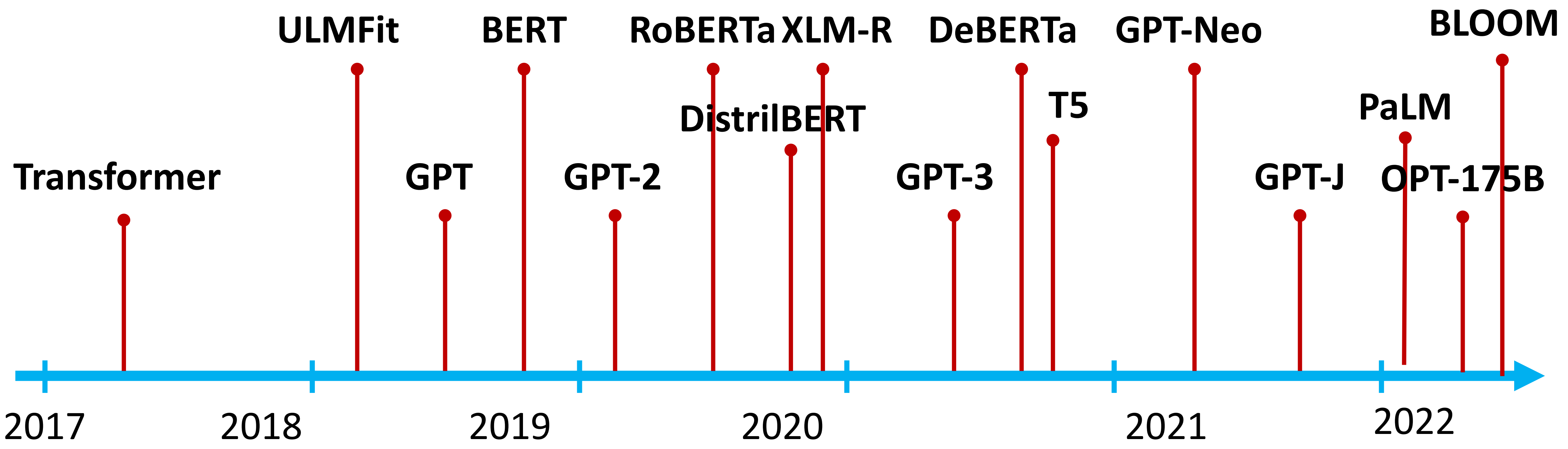


Source: https://github.com/thunlp/PLMpapers

Outline

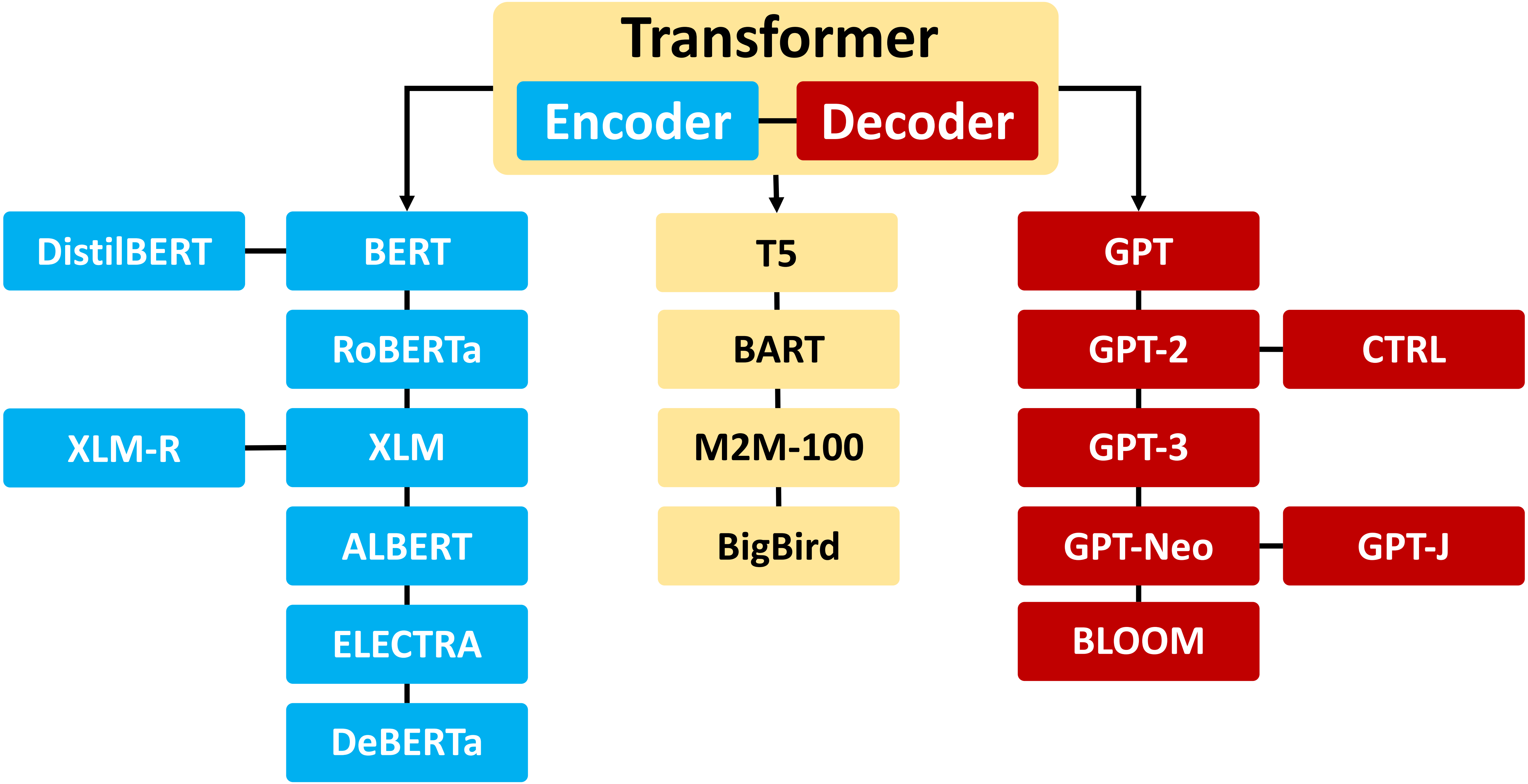
- Word Embeddings
- Recurrent Neural Networks for NLP
- Sequence-to-Sequence Models
- The Transformer Architecture
- Pretraining and Transfer Learning
- State of the art (SOTA)

The Transformers Timeline



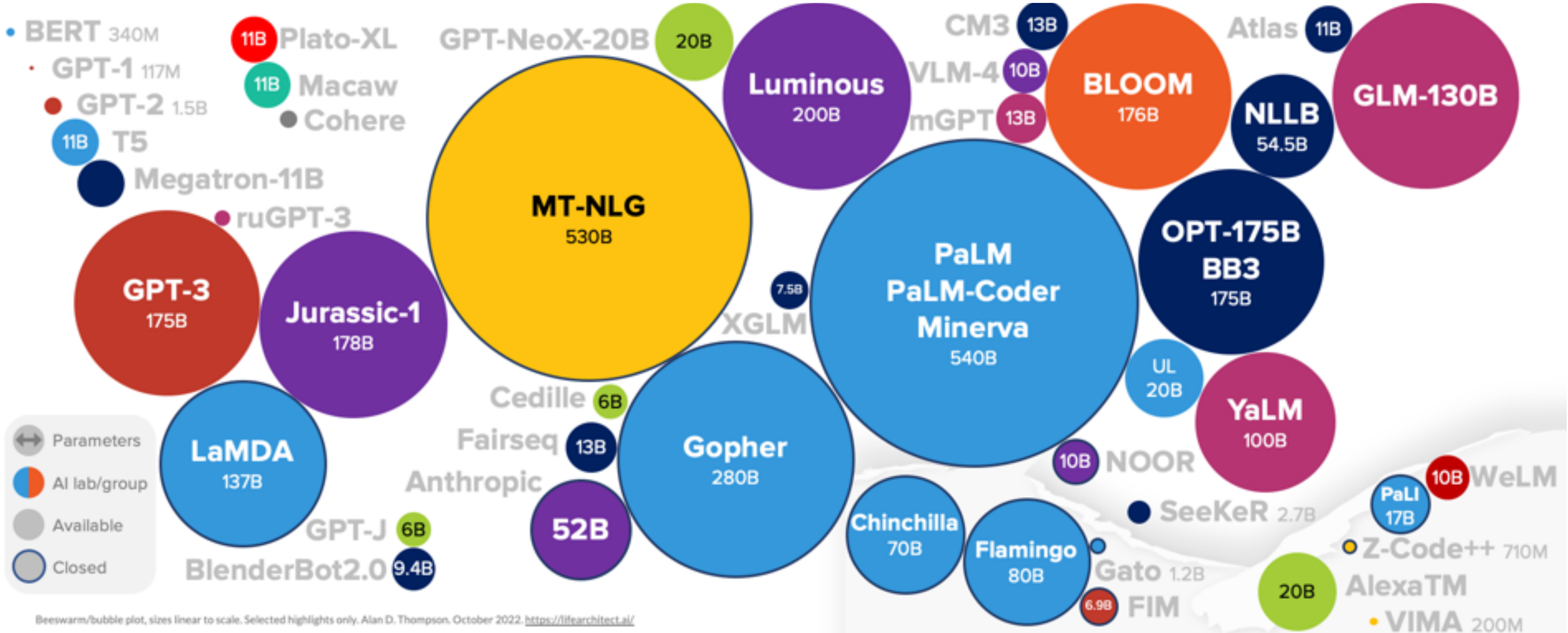
Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

Transformer Models



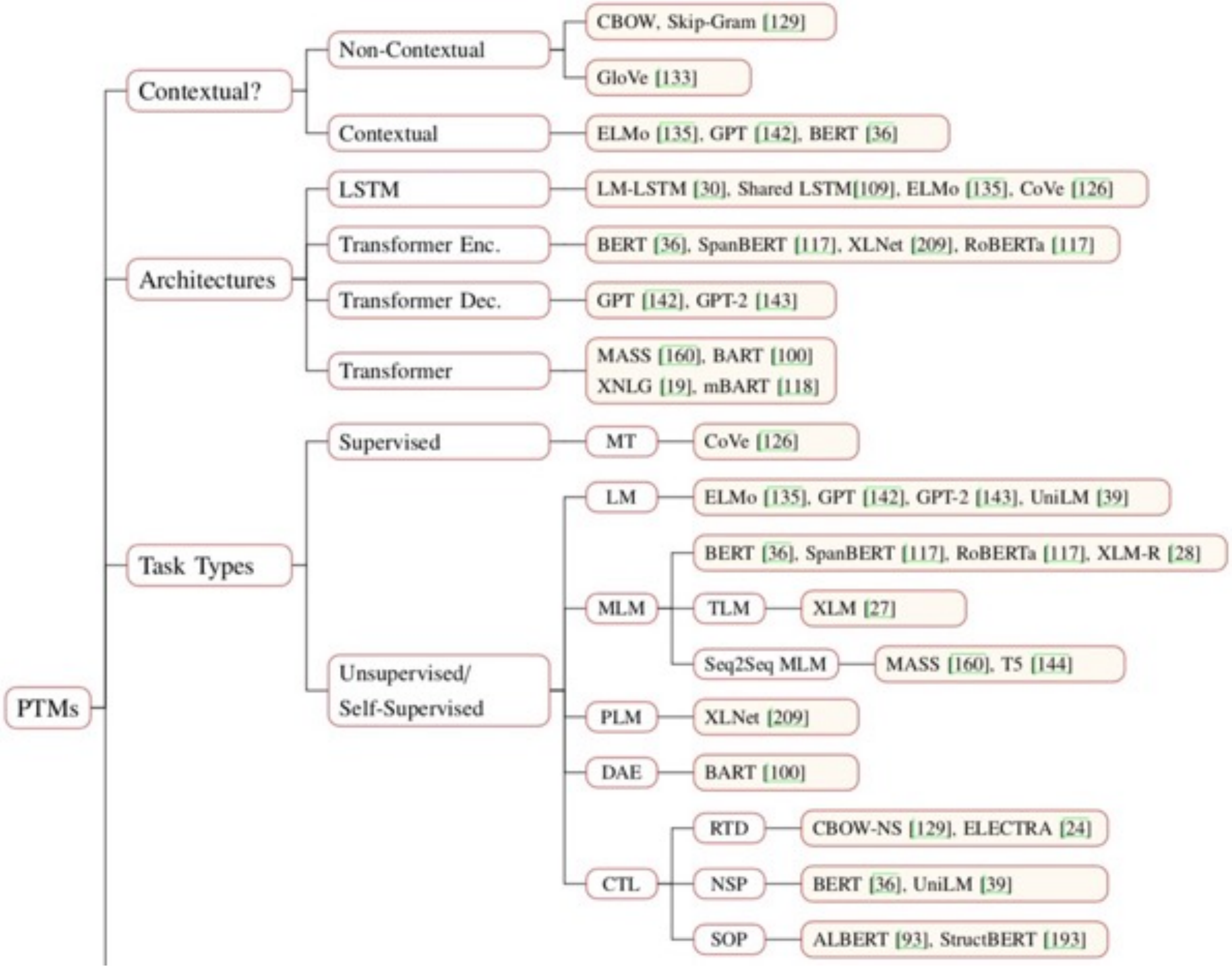
Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

Language Models Sizes
(GPT-3, PaLM, BLOOM)



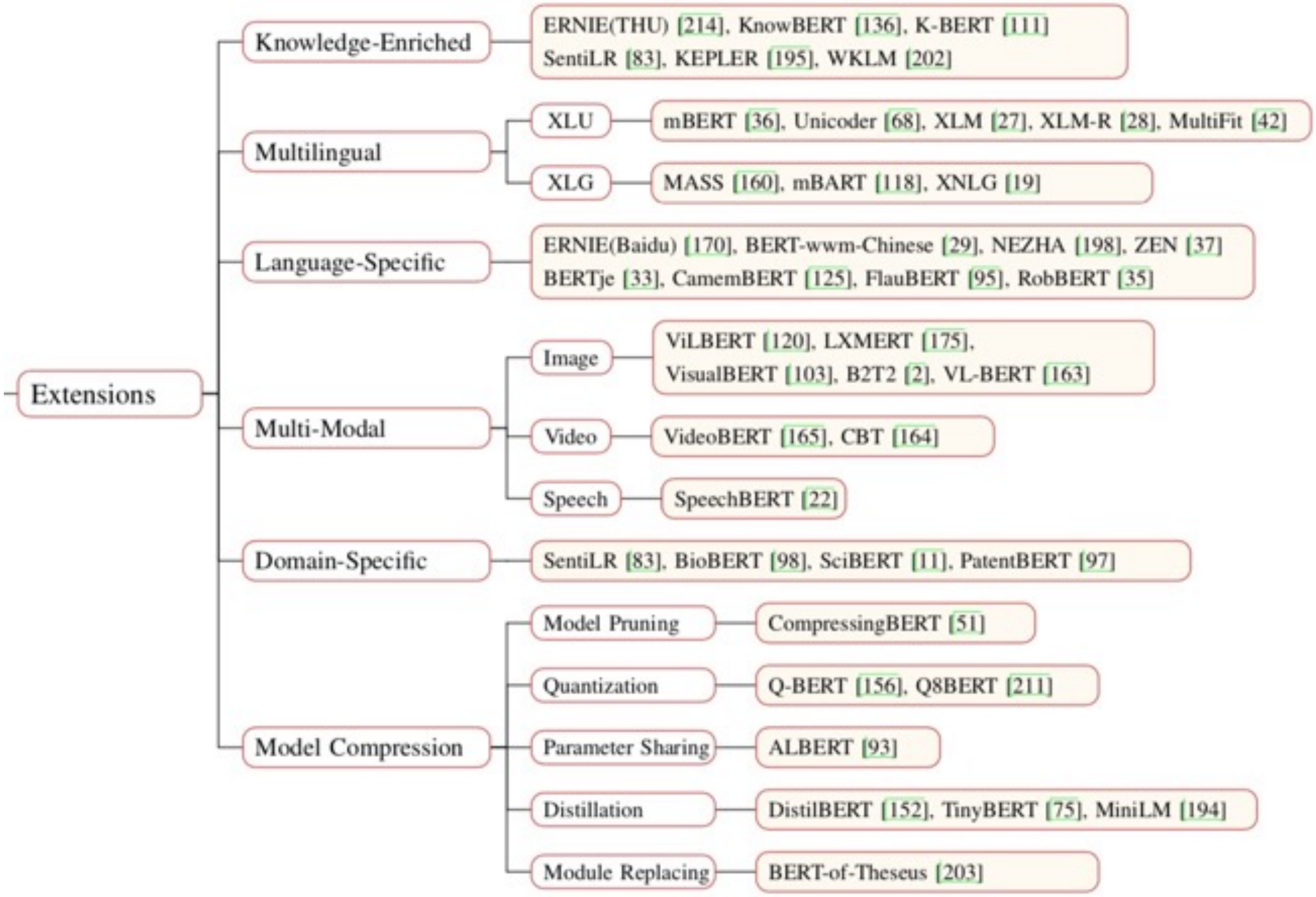
Source: https://lifearchitect.ai/models/

Pre-trained Models (PTM)



Source: Qiu, Xipeng, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. "Pre-trained Models for Natural Language Processing: A Survey." arXiv preprint arXiv:2003.08271 (2020).

Pre-trained Models (PTM)



Source: Qiu, Xipeng, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. "Pre-trained Models for Natural Language Processing: A Survey." arXiv preprint arXiv:2003.08271 (2020).

🤗 Transformers

Transformers

State-of-the-art Natural Language Processing for TensorFlow 2.0 and PyTorch

- Transformers
  - pytorch-transformers
  - pytorch-pretrained-bert
- provides state-of-the-art general-purpose architectures
  - (BERT, GPT-2, RoBERTa, XLM, DistilBert, XLNet, CTRL...)
  - for Natural Language Understanding (NLU) and
    Natural Language Generation (NLG)
    with over 32+ pretrained models
    in 100+ languages
    and deep interoperability between
    TensorFlow 2.0 and
    PyTorch.

Source: https://github.com/huggingface/transformers

NLP Benchmark Datasets

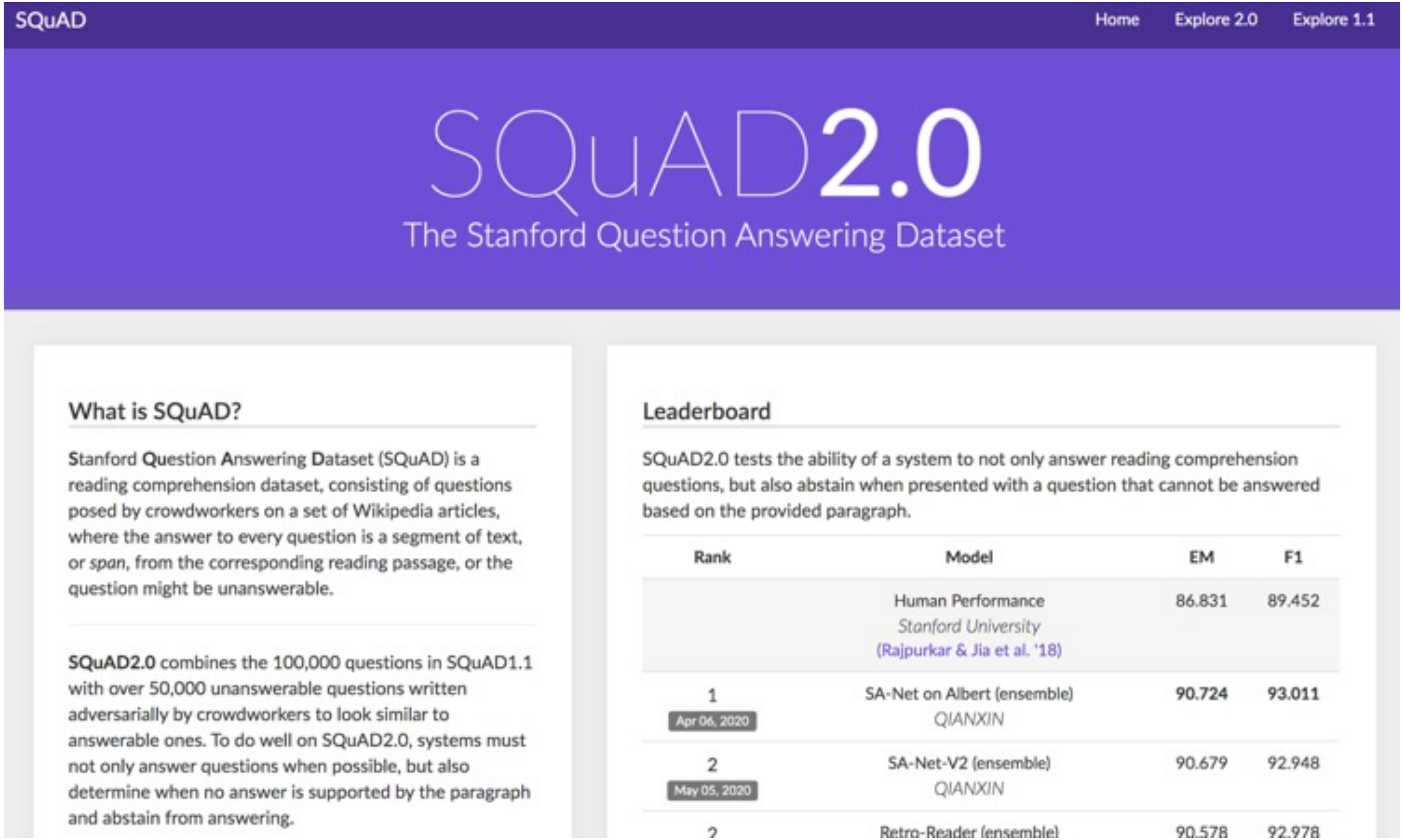
Task	Dataset	Link
Machine Translation	WMT 2014 EN-DE WMT 2014 EN-FR	http://www-lium.univ-lemans.fr/~schwenk/cslm_joint_paper/
Text Summarization	CNN/DM Newsroom DUC Gigaword	https://cs.nyu.edu/~kcho/DMQA/ https://summari.es/ https://www-nlpir.nist.gov/projects/duc/data.html https://catalog.ldc.upenn.edu/LDC2012T21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

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

Question Answering
(QA)
SQuAD

Stanford Question Answering Dataset

SQuAD



SQuAD Home Explore 2.0 Explore 1.1

SQuAD2.0

The Stanford Question Answering Dataset

What is SQuAD?

Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or *span*, from the corresponding reading passage, or the question might be unanswerable.

SQuAD2.0 combines the 100,000 questions in SQuAD1.1 with over 50,000 unanswerable questions written adversarially by crowdworkers to look similar to answerable ones. To do well on SQuAD2.0, systems must not only answer questions when possible, but also determine when no answer is supported by the paragraph and abstain from answering.

Leaderboard

SQuAD2.0 tests the ability of a system to not only answer reading comprehension questions, but also abstain when presented with a question that cannot be answered based on the provided paragraph.

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Apr 06, 2020	SA-Net on Albert (ensemble) *QIANXIN*	**90.724**	**93.011**
2 May 05, 2020	SA-Net-V2 (ensemble) *QIANXIN*	90.679	92.948
2	Retro-Reader (ensemble)	90.578	92.978

https://rajpurkar.github.io/SQuAD-explorer/

SQuAD

SQuAD: 100,000+ Questions for Machine Comprehension of Text

Pranav Rajpurkar and **Jian Zhang** and **Konstantin Lopyrev** and **Percy Liang**
{pranavsr,zjian,klopyrev,pliang}@cs.stanford.edu
Computer Science Department
Stanford University

Abstract

We present the Stanford Question Answering Dataset (SQuAD), a new reading comprehension dataset consisting of 100,000+ questions posed by crowdworkers on a set of Wikipedia articles, where the answer to each question is a segment of text from the corresponding reading passage. We analyze the dataset to understand the types of reasoning required to answer the questions, leaning heavily on dependency and constituency trees. We build a strong logistic regression model, which achieves an F1 score of 51.0%, a significant improvement over a simple baseline (20%). However, human performance (86.8%) is much higher, indicating that the dataset presents a good challenge problem for future research. The dataset is freely available at https://stanford-qa.com.

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud

Figure 1: Question-answer pairs for a sample passage in the

Source: Rajpurkar, Pranav, Jian Zhang, Konstantin Lopyrev, and Percy Liang.
"Squad: 100,000+ questions for machine comprehension of text." arXiv preprint arXiv:1606.05250 (2016).

SQuAD (Question Answering)

Q: What causes precipitation to fall?

Precipitation

From Wikipedia, the free encyclopedia

For other uses, see Precipitation (disambiguation).

In meteorology, **precipitation** is any product of the condensation of atmospheric water vapor that falls under gravity from clouds.[2] The main forms of precipitation include drizzle, rain, sleet, snow, ice pellets, graupel and hail. Precipitation occurs when a portion of the atmosphere becomes saturated with water vapor (reaching 100% relative humidity), so that the water condenses and "precipitates". Thus, fog and mist are not precipitation but suspensions, because the water vapor does not condense sufficiently to precipitate. Two processes, possibly acting together, can lead to air becoming saturated: cooling the air or adding water vapor to the air. Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called "showers."[3]

https://en.wikipedia.org/wiki/Precipitation

SQuAD (Question Answering)

Paragraph

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called “showers”.

Q: What causes precipitation to fall?

SQuAD (Question Answering)

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called “showers”.

Q: What causes precipitation to fall?

A: gravity

SQuAD (Question Answering)

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called “showers”.

Q: What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?

A: graupel

SQuAD (Question Answering)

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called “showers”.

Q: Where do water droplets collide with ice crystals to form precipitation?

A: within a cloud

SQuAD (Question Answering)

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called “showers”.

Q: What causes precipitation to fall?

A: gravity

Q: What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?

A: graupel

Q: Where do water droplets collide with ice crystals to form precipitation?

A: within a cloud

Natural Language Processing with Python

– Analyzing Text with the Natural Language Toolkit

www.nltk.org/book/

Natural Language Processing with Python

– Analyzing Text with the Natural Language Toolkit

NLTK

Steven Bird, Ewan Klein, and Edward Loper

This version of the NLTK book is updated for Python 3 and NLTK 3. The first edition of the book, published by O'Reilly, is available at http://nltk.org/book_1ed/. (There are currently no plans for a second edition of the book.)

Please post any questions about the materials to the nltk-users mailing list. Please report any errors on the issue tracker.

http://www.nltk.org/book/

spaCy

https://spacy.io/

gensim

https://radimrehurek.com/gensim/

TextBlob

TextBlob

Star 3,777

TextBlob is a Python (2 and 3) library for processing textual data. It provides a consistent API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, and more.

Useful Links

TextBlob @ PyPI
TextBlob @ GitHub
Issue Tracker

Stay Informed

Follow @sloria

Donate

If you find TextBlob useful,

TextBlob: Simplified Text Processing

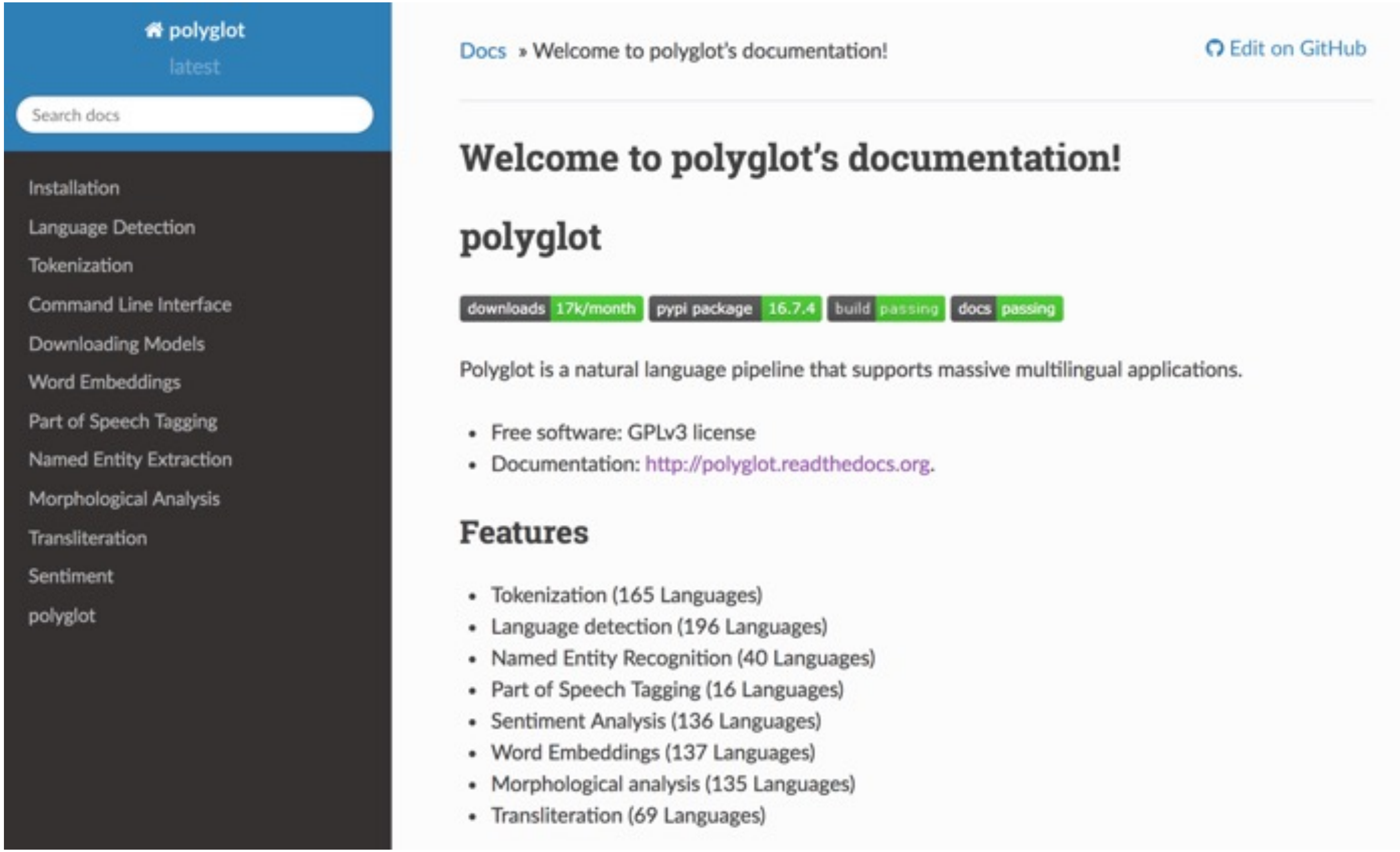
Release v0.12.0. (Changelog)

TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

```
from textblob import TextBlob

text = '''
The titular threat of The Blob has always struck me as the ultimate movie
monster: an insatiably hungry, amoeba-like mass able to penetrate
virtually any safeguard, capable of--as a doomed doctor chillingly
describes it--"assimilating flesh on contact.
Snide comparisons to gelatin be damned, it's a concept with the most
devastating of potential consequences, not unlike the grey goo scenario
proposed by technological theorists fearful of
artificial intelligence run rampant.
'''

blob = TextBlob(text)
blob.tags           # [('The', 'DT'), ('titular', 'JJ'),
                    #  ('threat', 'NN'), ('of', 'IN'), ...]

blob.noun_phrases   # WordList(['titular threat', 'blob',
                    #             'ultimate movie monster',
                    #             'amoeba-like mass', ...])

for sentence in blob.sentences:
    print(sentence.sentiment.polarity)
# 0.060
```

https://textblob.readthedocs.io

Polyglot



polyglot
latest
Search docs

- Installation
- Language Detection
- Tokenization
- Command Line Interface
- Downloading Models
- Word Embeddings
- Part of Speech Tagging
- Named Entity Extraction
- Morphological Analysis
- Transliteration
- Sentiment
- polyglot

Docs » Welcome to polyglot's documentation!   Edit on GitHub

Welcome to polyglot's documentation!

polyglot

downloads 17k/month | pypi package 16.7.4 | build passing | docs passing

Polyglot is a natural language pipeline that supports massive multilingual applications.

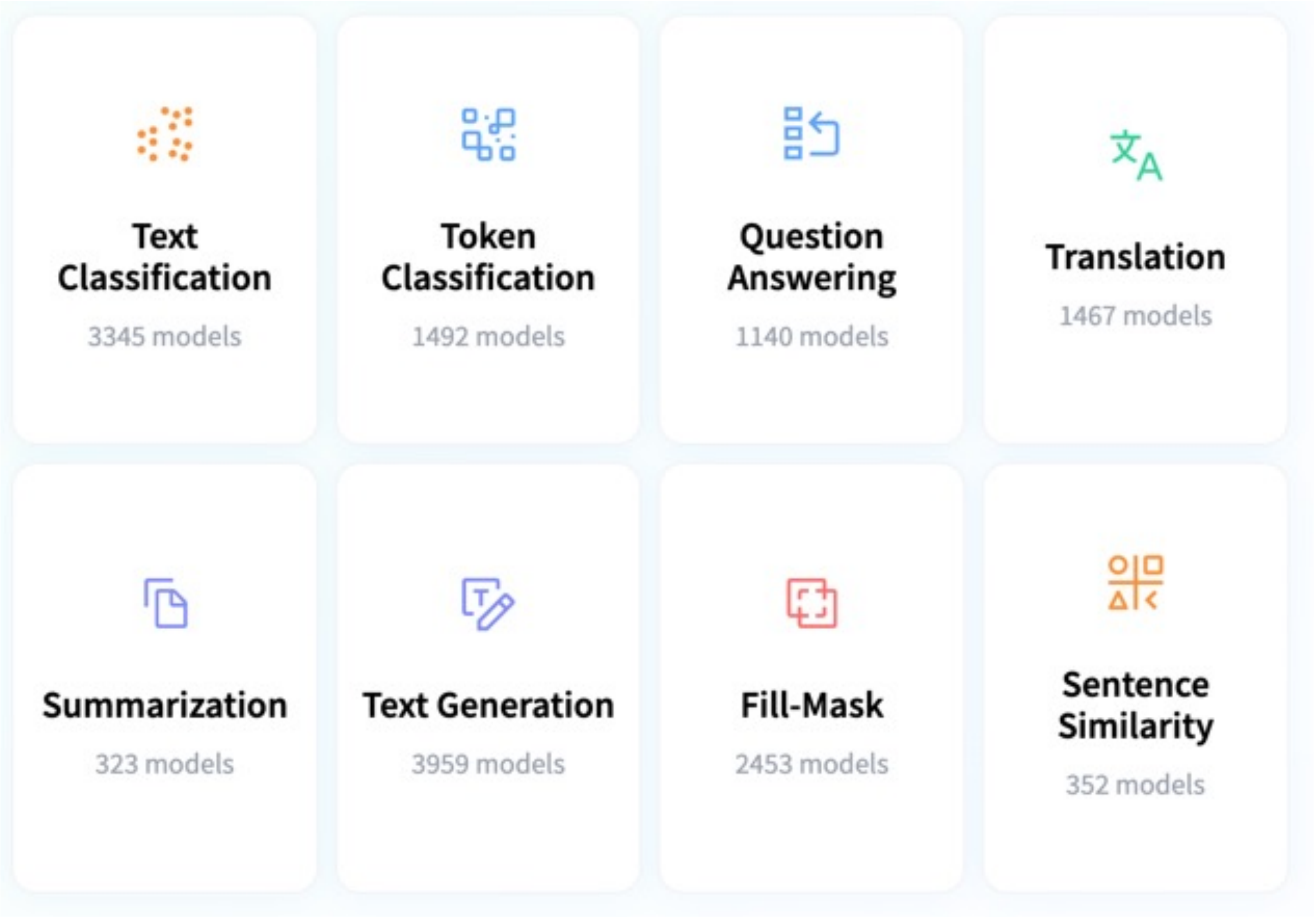
- Free software: GPLv3 license
- Documentation: http://polyglot.readthedocs.org.

Features

- Tokenization (165 Languages)
- Language detection (196 Languages)
- Named Entity Recognition (40 Languages)
- Part of Speech Tagging (16 Languages)
- Sentiment Analysis (136 Languages)
- Word Embeddings (137 Languages)
- Morphological analysis (135 Languages)
- Transliteration (69 Languages)

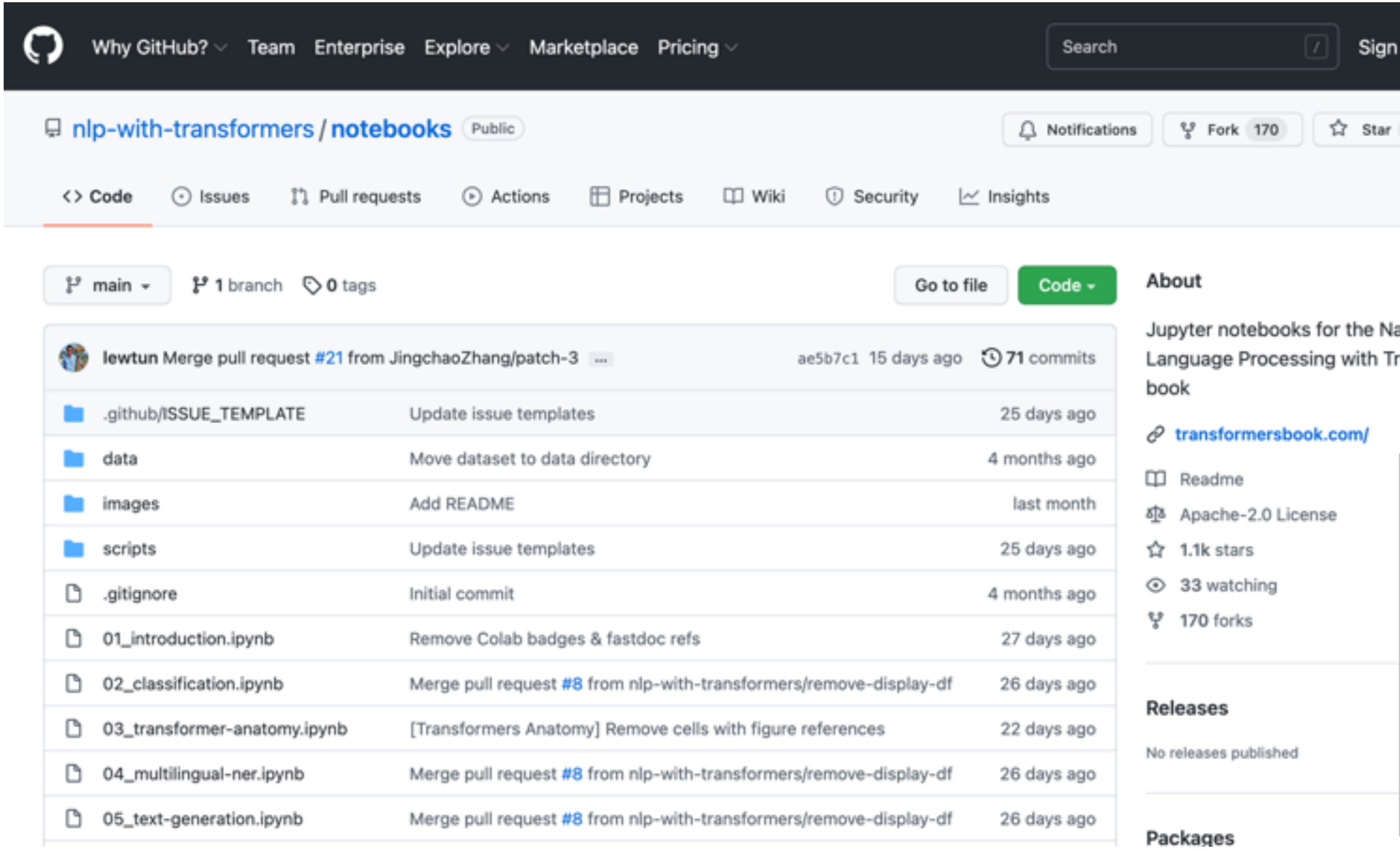
https://polyglot.readthedocs.io/

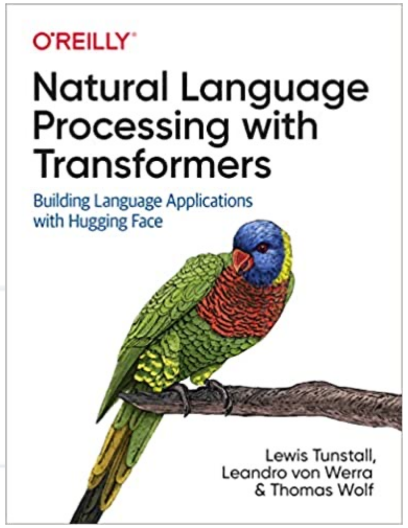
Hugging Face Tasks
Natural Language Processing



https://huggingface.co/tasks

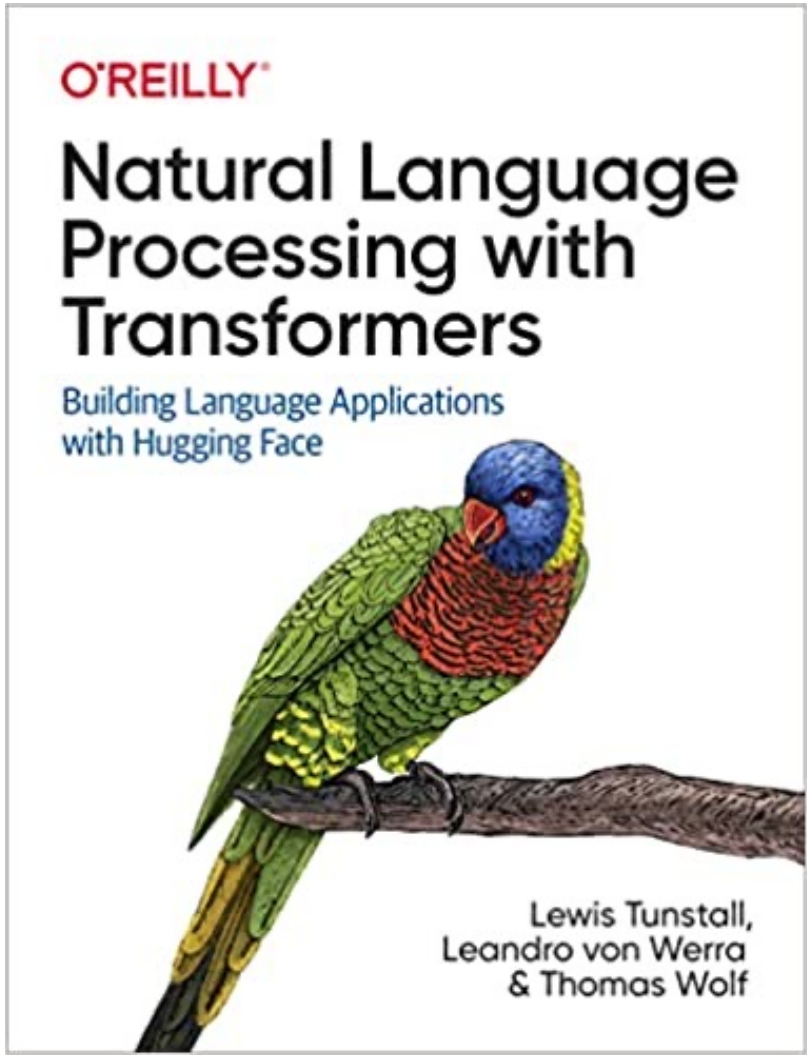
NLP with Transformers Github





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

NLP with Transformers Github Notebooks



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

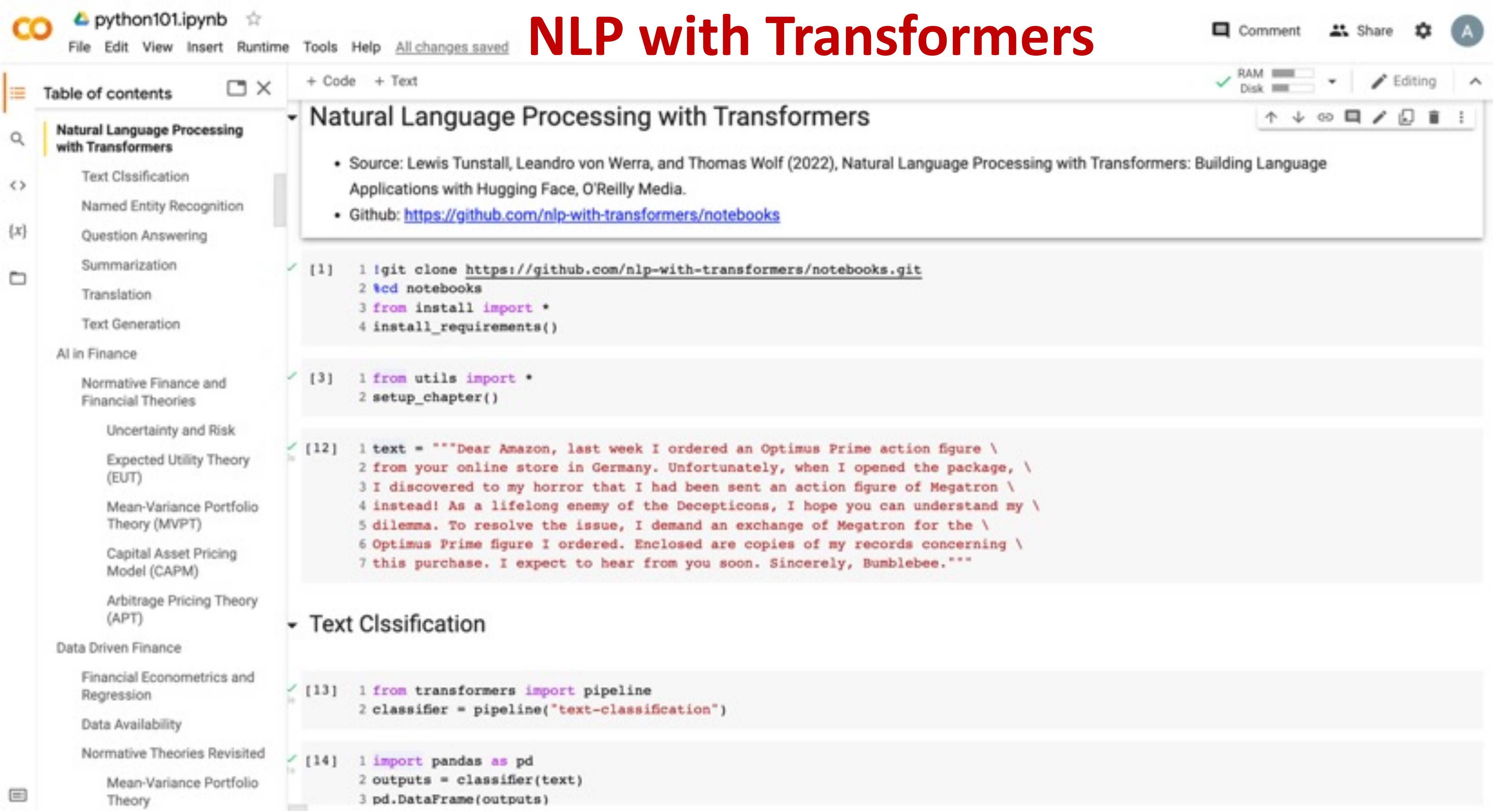
Nowadays, the GPUs on Colab tend to be K80s (which have limited memory), so we recommend using Kaggle, Gradient, or SageMaker Studio Lab. These platforms tend to provide more performant GPUs like P100s, all for free!

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

Python in Google Colab (Python101)

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

NLP with Transformers

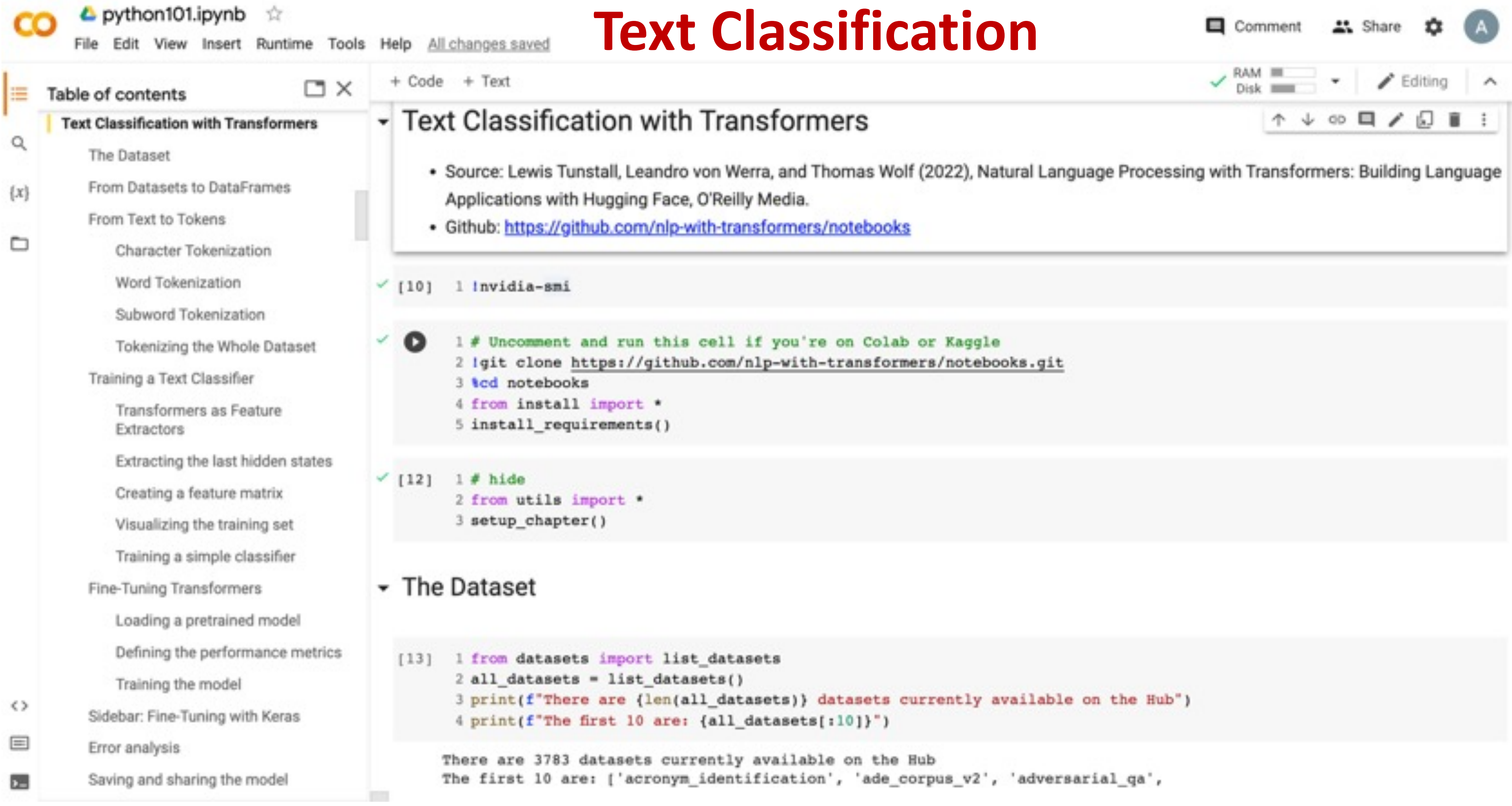


https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

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

Text Classification



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

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

Named Entity Recognition (NER)

python101.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

Multilingual Named Entity Recognition (NER)

- 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

```
#NER: https://huggingface.co/tasks/token-classification
!pip install transformers
from transformers import pipeline
classifier = pipeline("ner")
classifier("Hello I'm Omar and I live in Zürich.")
```

```
from transformers import pipeline
classifier = pipeline("ner")
classifier("Hello I'm Omar and I live in Zürich.")
```

```
No model was supplied, defaulted to dbmdz/bert-large-cased-finetuned-conll03-english (https://huggingface.co/dbmdz/bert-large-cased-finetuned-conll03-eng
[{'end': 14,
  'entity': 'I-PER',
  'index': 5,
  'score': 0.99770516,
  'start': 10,
  'word': 'Omar'},
 {'end': 35,
  'entity': 'I-LOC',
  'index': 10,
  'score': 0.9968976,
  'start': 29,
  'word': 'Zürich'}]
```

https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

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

python101.ipynb

File Edit View Insert Runtime Tools Help Saving...

Text Summarization

Comment Share

+ Code + Text

RAM Disk Editing

Text Summarization

- 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

```
#Source: https://huggingface.co/tasks/summarization
!pip install transformers
from transformers import pipeline
classifier = pipeline("summarization")
text = "Paris is the capital and most populous city of France, with an estimated population of 2,175,601 residents as of 2018, in an area of more thar
classifier(text, max_length=30)
```

```
No model was supplied, defaulted to sshleifer/distilbart-cnn-12-6 (https://huggingface.co/sshleifer/distilbart-cnn-12-6)
Your min_length=56 must be inferior than your max_length=30.
[{'summary_text': ' Paris is the capital and most populous city of France, with an estimated population of 2,175,601 residents . The City of Paris'}]
```

```
#!pip install transformers
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
summarizer = pipeline("summarization")
outputs = summarizer(text, max_length=45, clean_up_tokenization_spaces=True)
print(outputs[0]['summary_text'])
```

https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

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

python101.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Text Generation

Comment Share

+ Code + Text

RAM Disk Editing

Text Generation

- 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

[9]

```
#Source: https://huggingface.co/tasks/text-generation
#!pip install transformers
from transformers import pipeline
generator = pipeline('text-generation', model = 'gpt2')
generator("Hello, I'm a language model", max_length = 30, num_return_sequences=3)
```

```
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
[{'generated_text': "Hello, I'm a language model.\n\nBut then, one day, I'm not trying to teach the language in my head.\n\n"},
 {'generated_text': "Hello, I'm a language model. I'm an implementation for the type system. I'm working with types and programming language constructs. I
 {'generated_text': "Hello, I'm a language modeler, not a programmer. As you know, languages are not a linear model. The thing that jumps out at"}]
```

```
from transformers import pipeline
generator = pipeline('text-generation', model = 'gpt2')
outputs = generator("Once upon a time", max_length = 30, num_return_sequences=3)
print(outputs[0]['generated_text'])
```

```
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Once upon a time, every person who ever saw Jesus, knew that He was Christ. And even though he might not have known Him, He was
```

[1]

```
from transformers import pipeline
```

https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

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

python101.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Question Answering and Dialogue Systems

Editing

Question Answering and Dialogue Systems

- 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

Question Answering

[3]

```
!pip install transformers
from transformers import pipeline
qamodel = pipeline("question-answering")
question = "Where do I live?"
context = "My name is Michael and I live in Taipei."
qamodel(question = question, context = context)
```

```
from transformers import pipeline
qamodel = pipeline("question-answering")
question = "Where do I live?"
context = "My name is Michael and I live in Taipei."
qamodel(question = question, context = context)
#{'answer': 'Taipei', 'end': 39, 'score': 0.9730741381645203, 'start': 33}
```

```
No model was supplied, defaulted to distilbert-base-cased-distilled-squad (https://huggingface.co/distilbert-base-cased-distilled-squad)
{'answer': 'Taipei', 'end': 39, 'score': 0.9730741381645203, 'start': 33}
```

https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

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

python101.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Question Answering

RAM Disk Editing

[12]

```python
from transformers import pipeline
qamodel = pipeline("question-answering", model ='deepset/roberta-base-squad2')
question = "What causes precipitation to fall?"
context = """In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravi
output = qamodel(question = question, context = context)
print(output['answer'])
```

```
gravity
```

[13]

```python
from transformers import pipeline
qamodel = pipeline("question-answering", model ='deepset/roberta-base-squad2')
question = "What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?"
context = """In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravi
output = qamodel(question = question, context = context)
print(output['answer'])
```

```
graupel
```

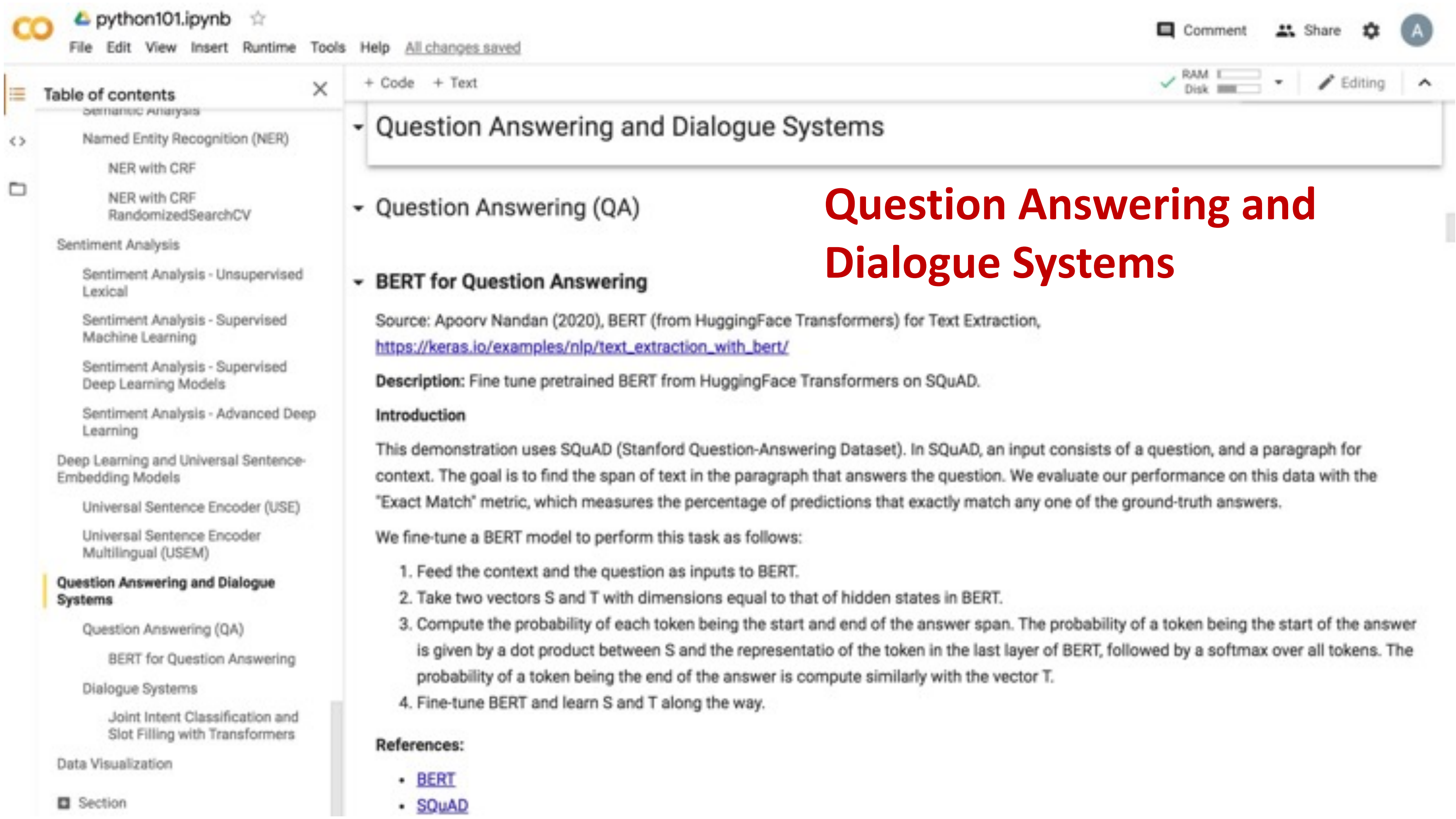
```python
#from transformers import pipeline
#qamodel = pipeline("question-answering", model ='deepset/roberta-base-squad2')
question = "Where do water droplets collide with ice crystals to form precipitation?"
context = """In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravi
output = qamodel(question = question, context = context)
print(output['answer'])
```

```
within a cloud
```

https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

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

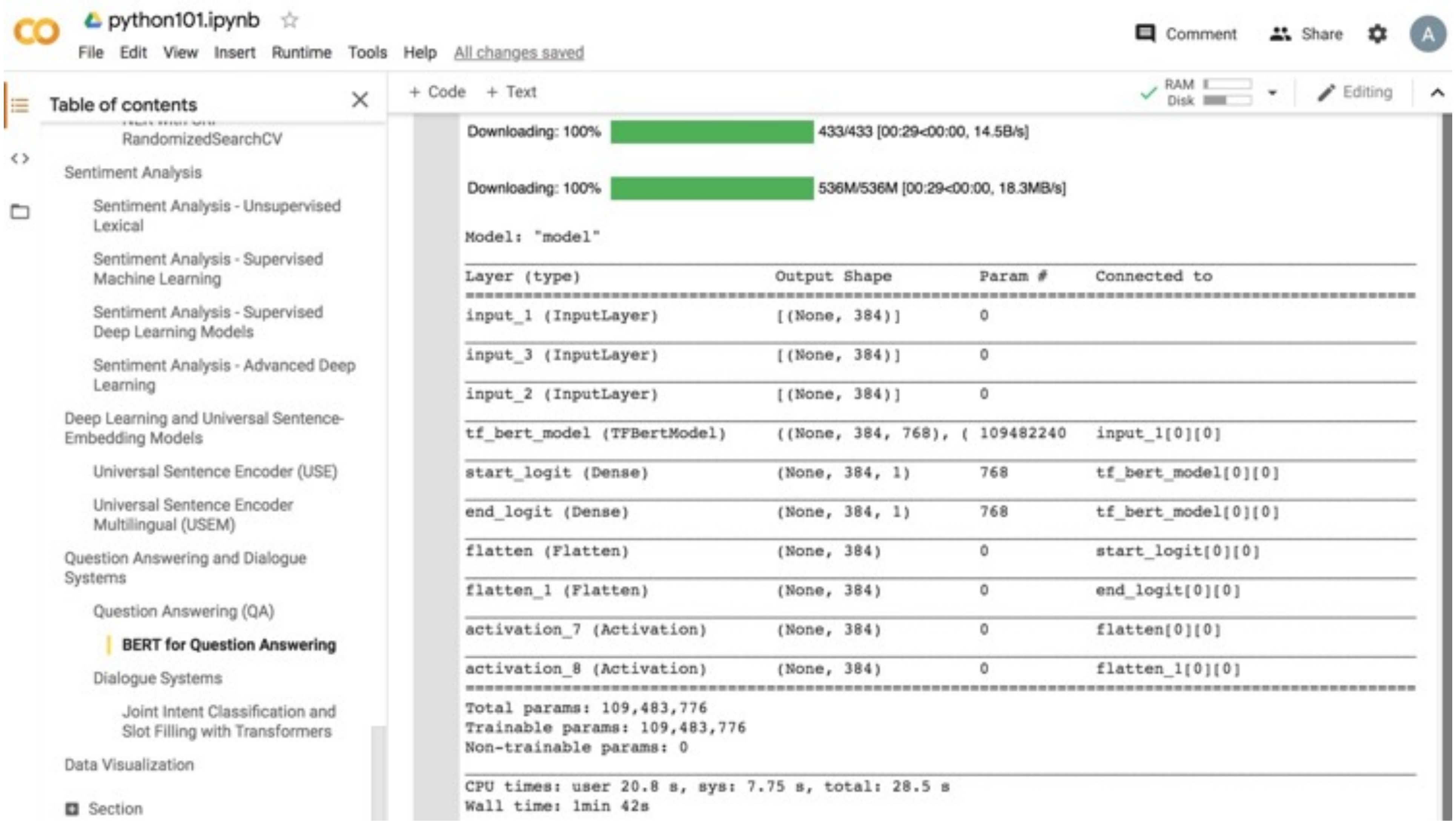


Question Answering and Dialogue Systems

https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

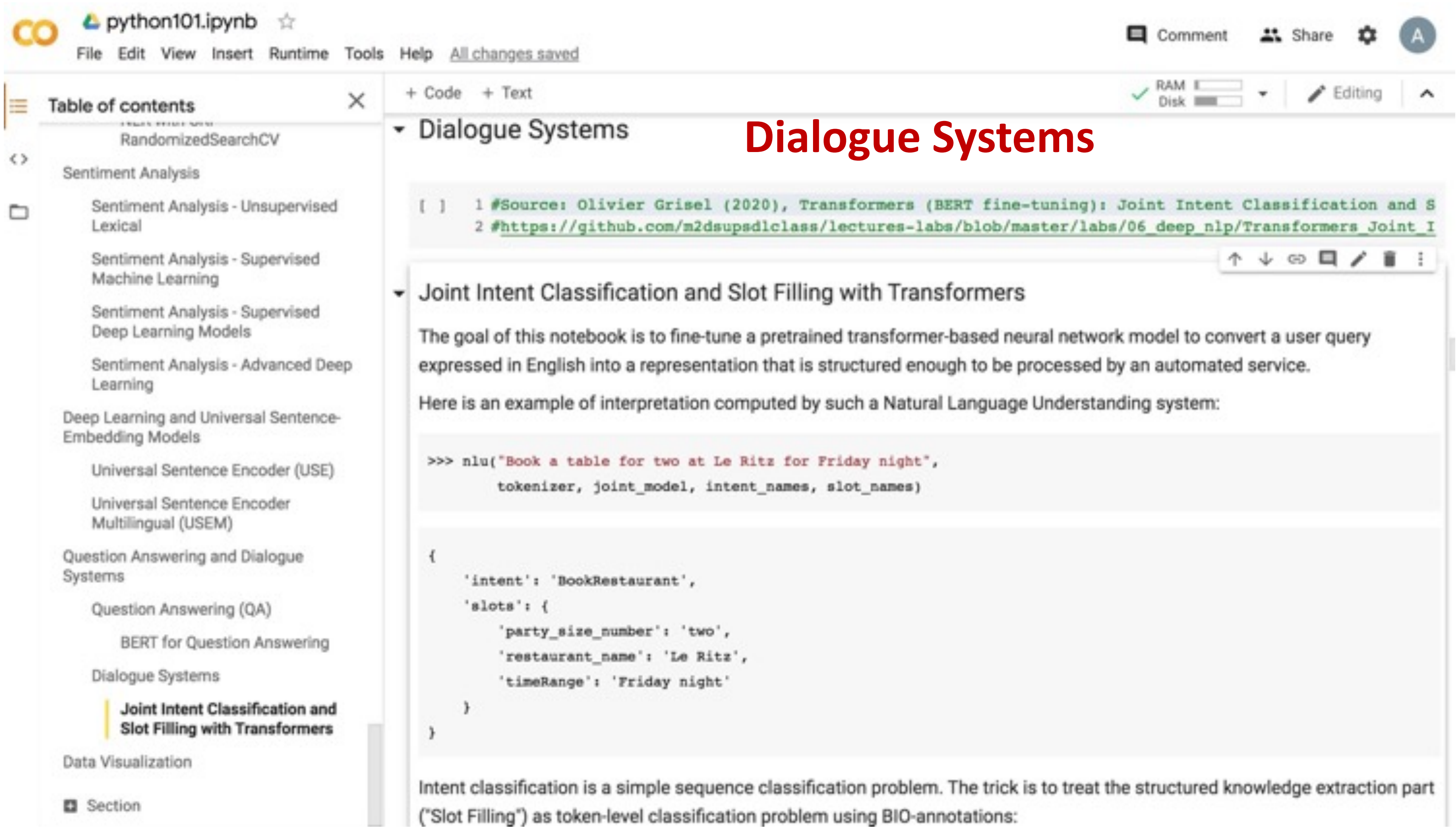
https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

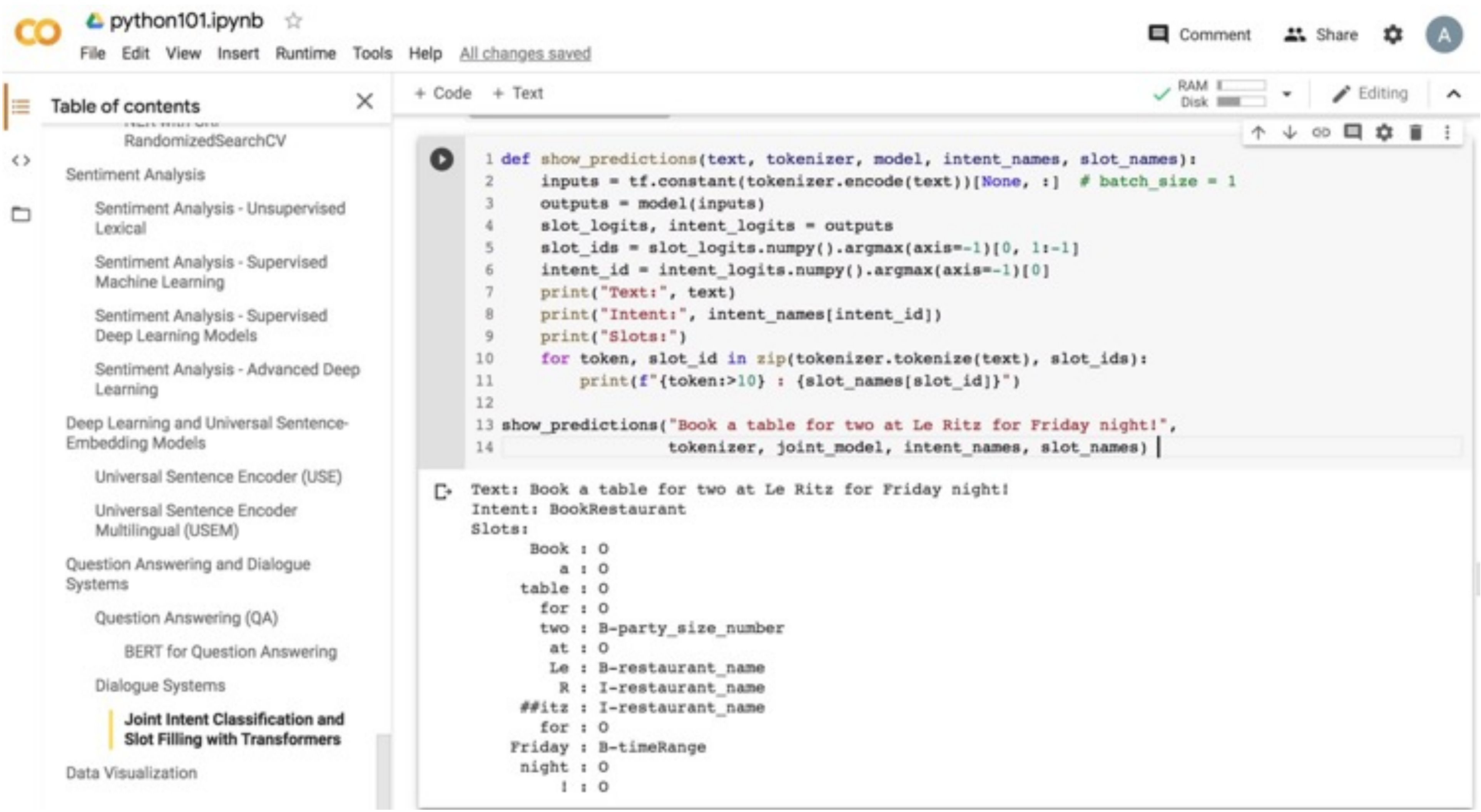
https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

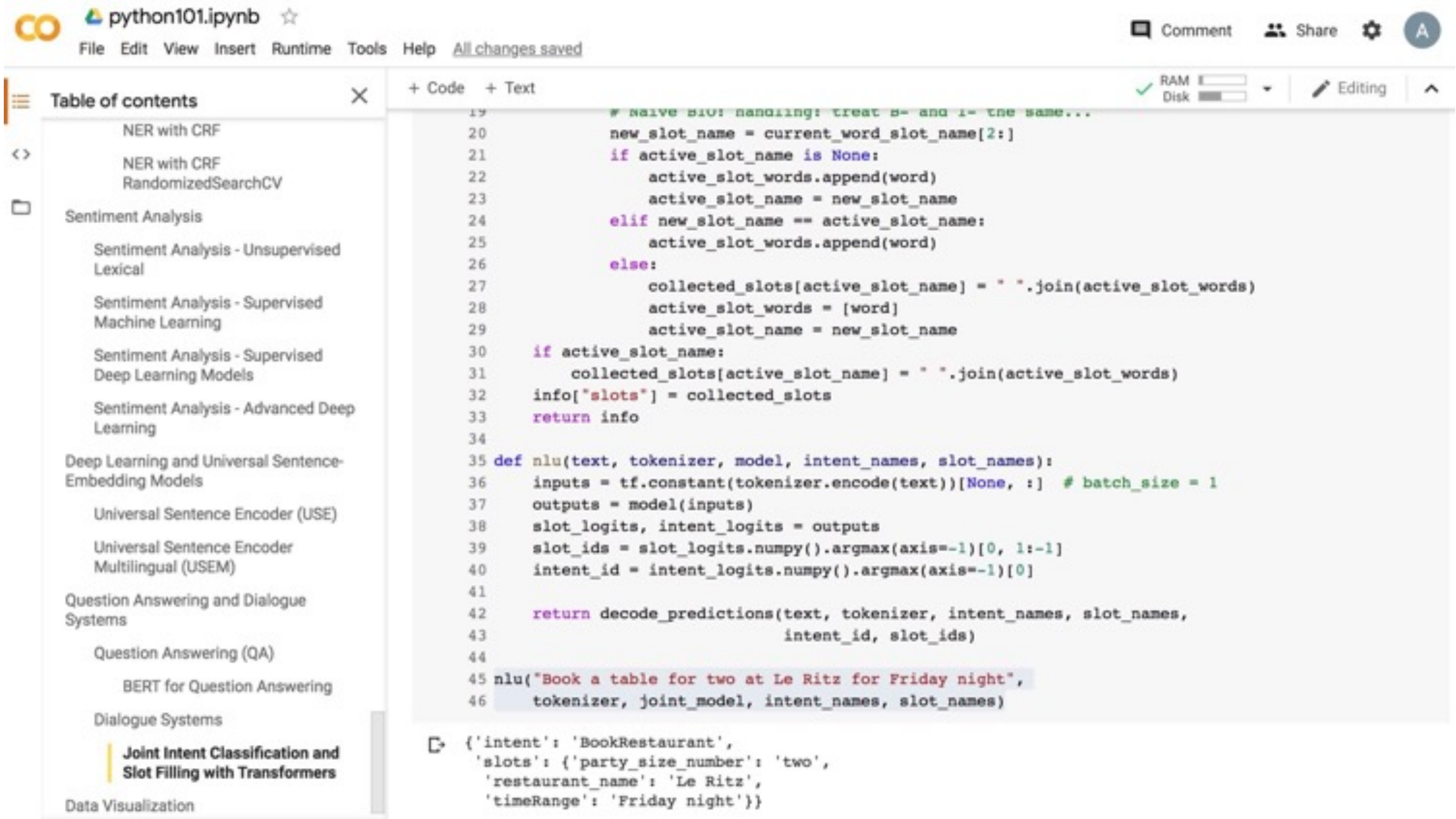
https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

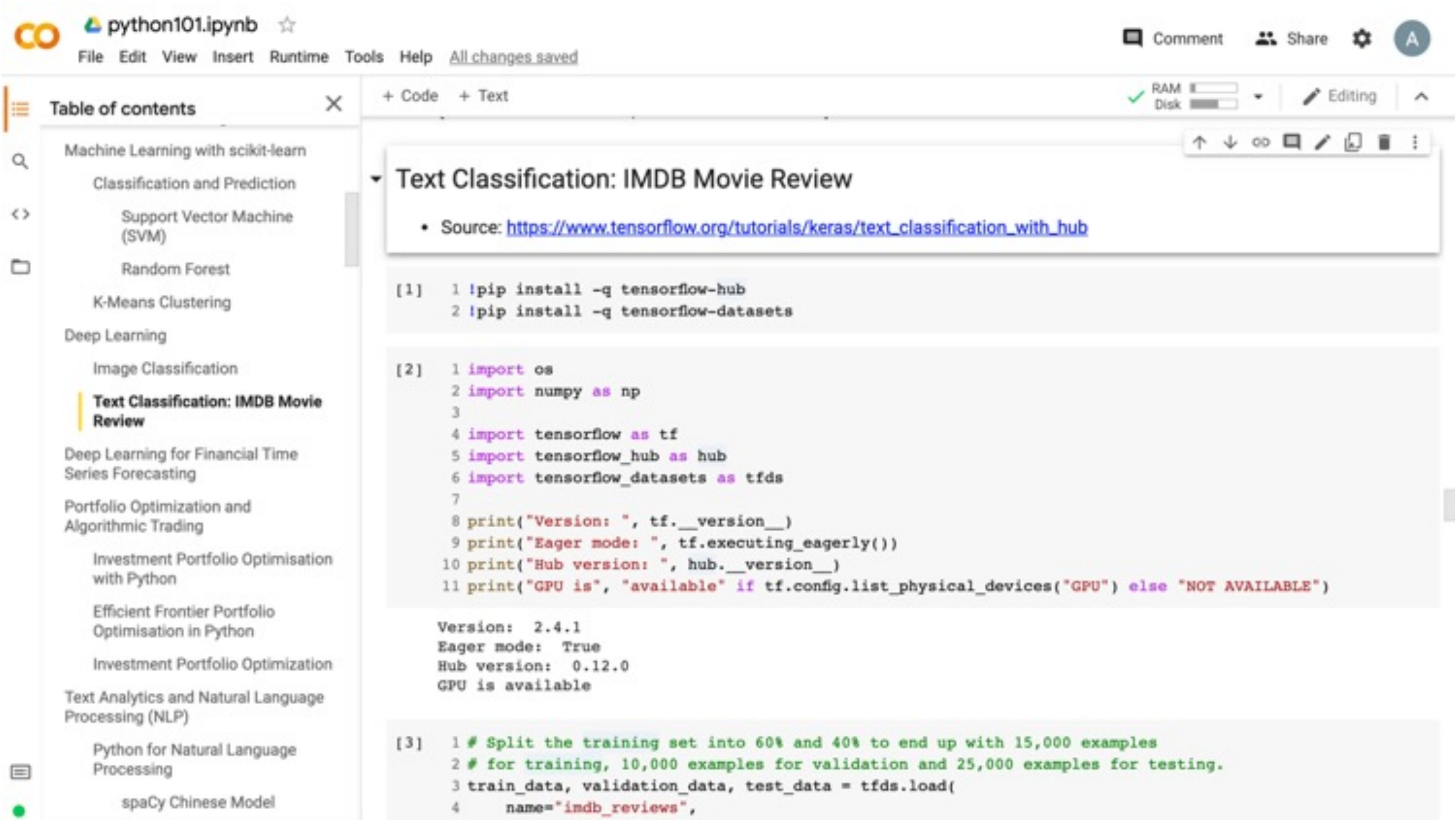
https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

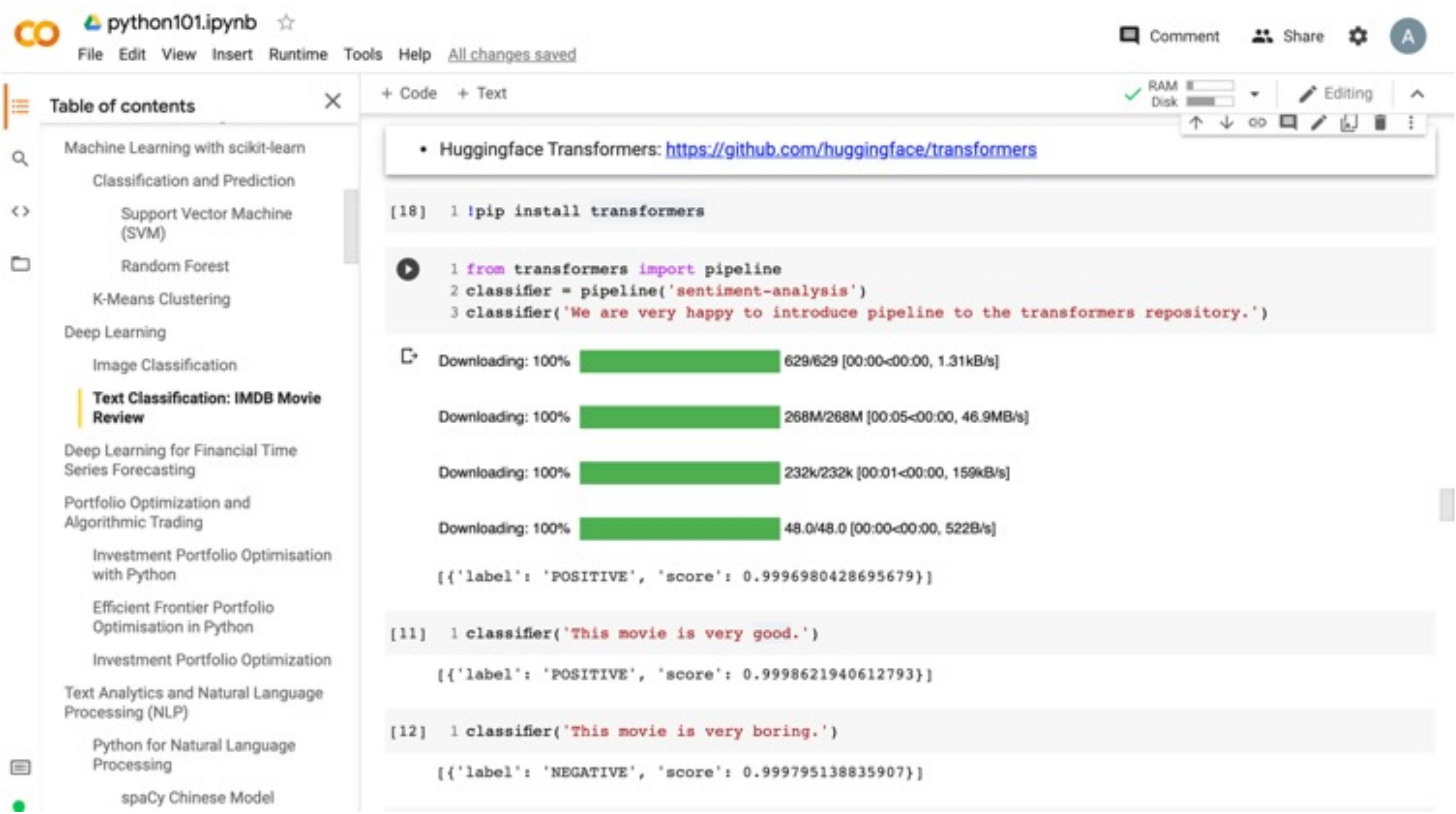
https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

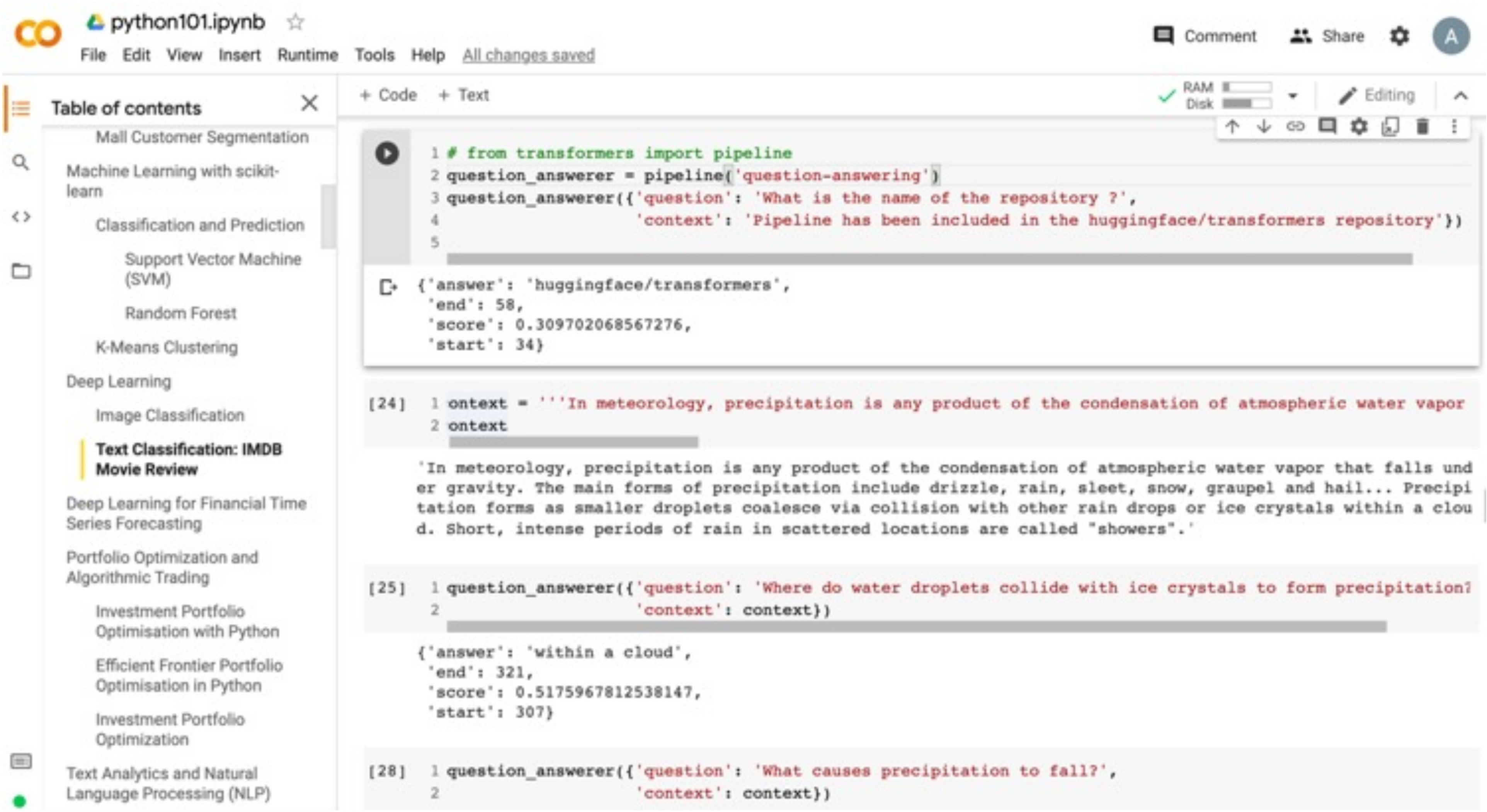
https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

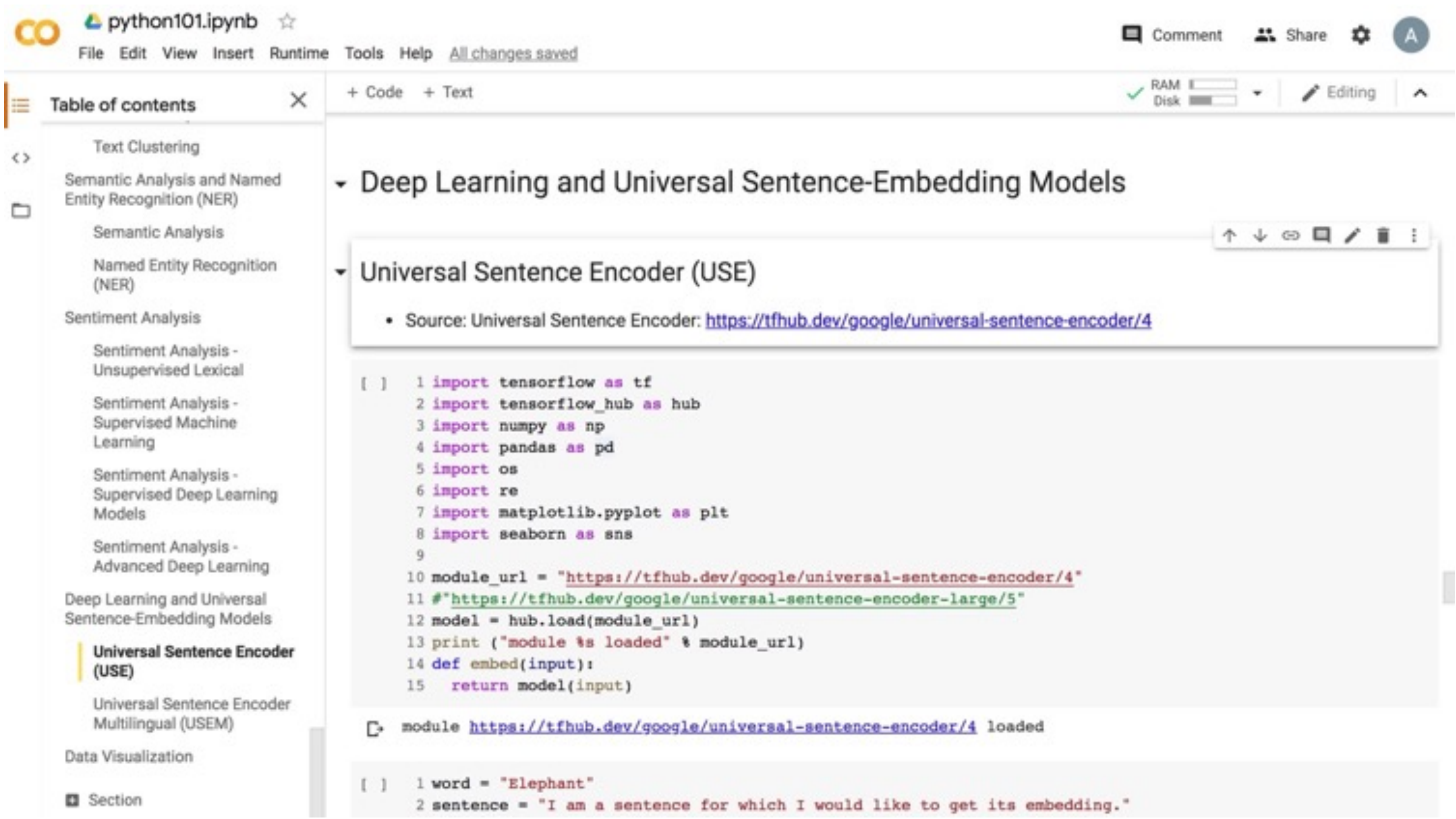
https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

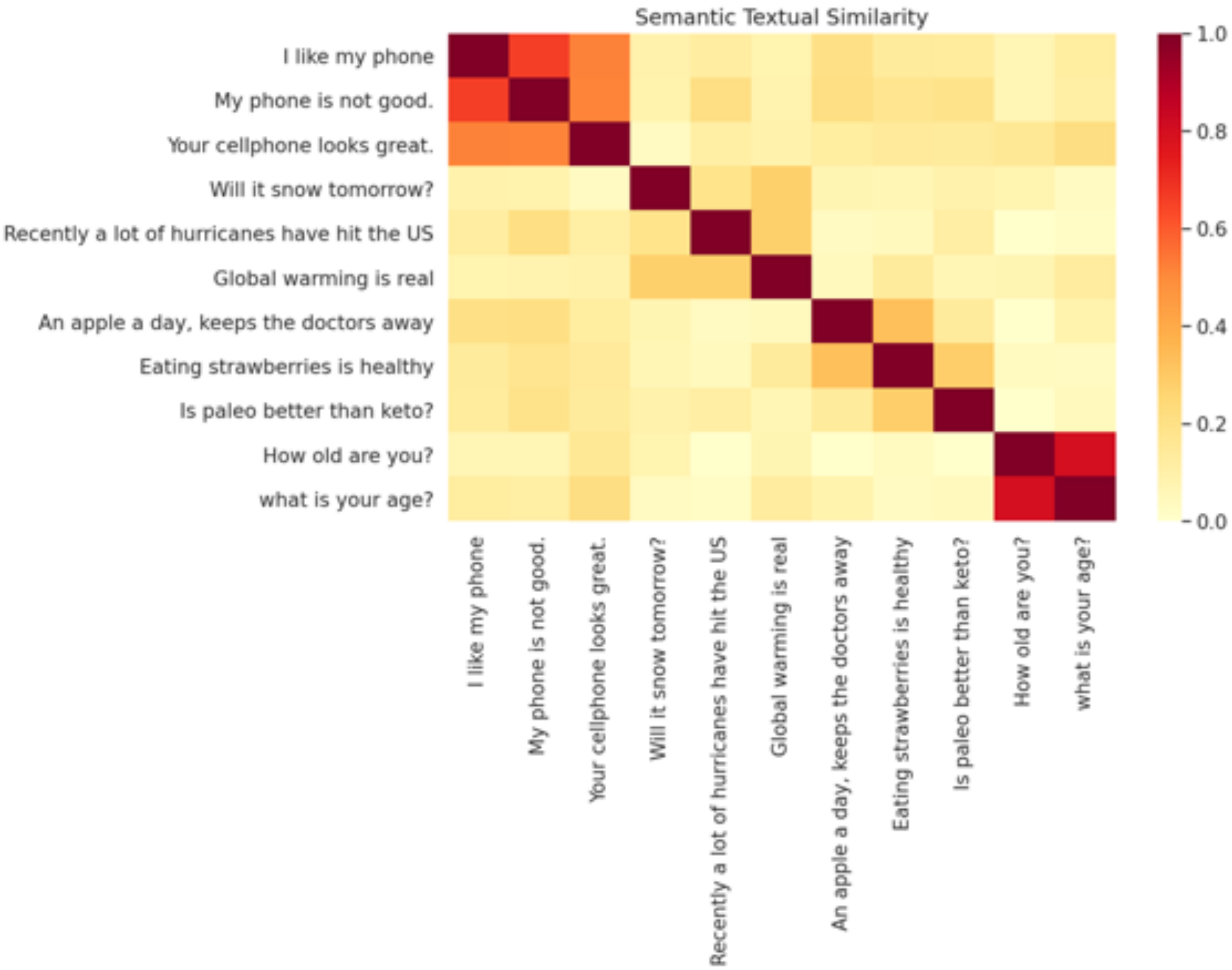
https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

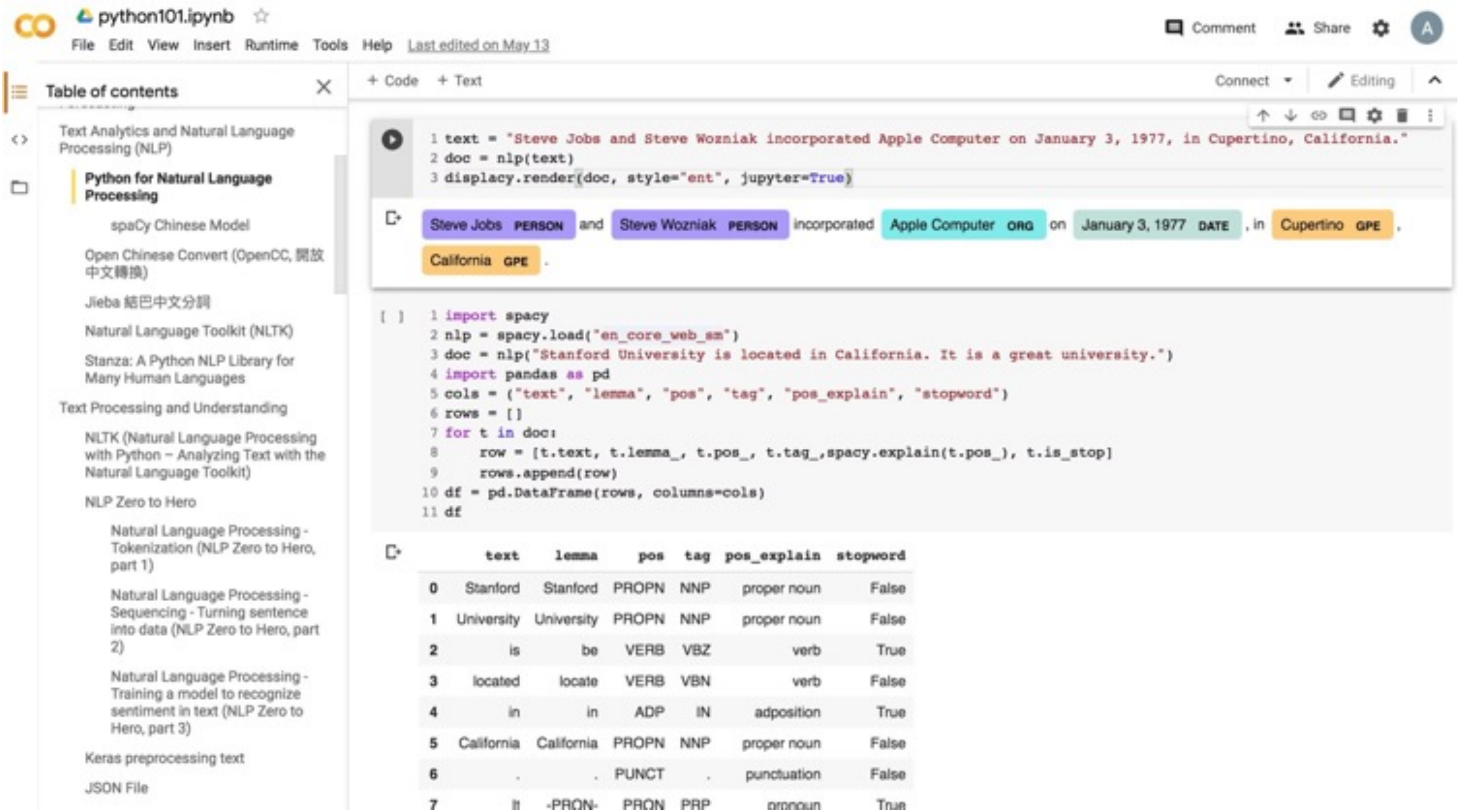
https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

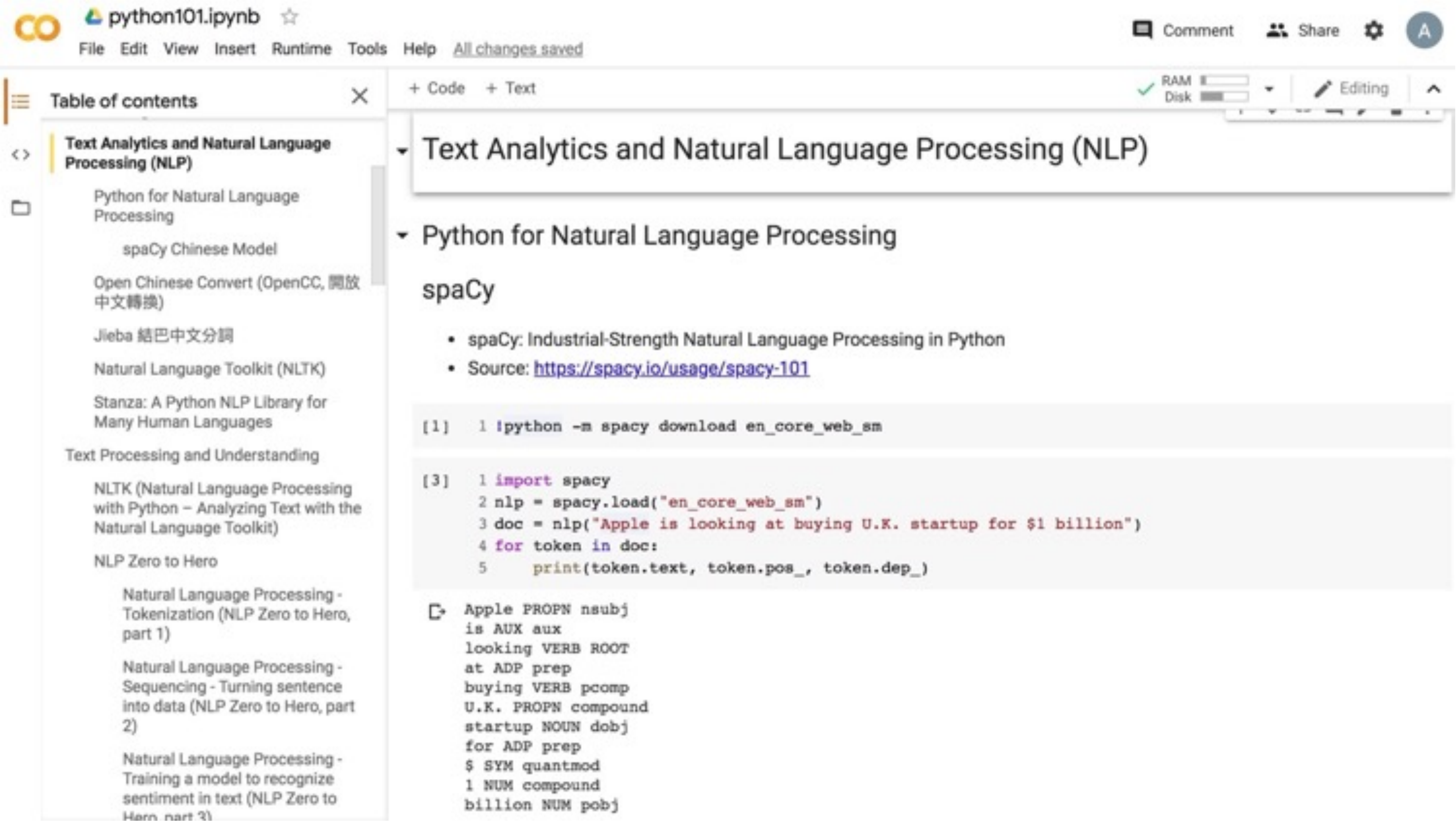
https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

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



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

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

python101.ipynb

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

	text	lemma	POS	explain	stopword
0	Apple	Apple	PROPN	proper noun	False
1	is	be	VERB	verb	True
2	looking	look	VERB	verb	False
3	at	at	ADP	adposition	True
4	buying	buy	VERB	verb	False
5	U.K.	U.K.	PROPN	proper noun	False
6	startup	startup	NOUN	noun	False
7	for	for	ADP	adposition	True
8	$	$	SYM	symbol	False
9	1	1	NUM	numeral	False
10	billion	billion	NUM	numeral	False

https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

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

python101.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

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

	text	lemma	POS	explain	stopword
0	Stanford	Stanford	PROPN	proper noun	False
1	University	University	PROPN	proper noun	False
2	is	be	VERB	verb	True
3	located	locate	VERB	verb	False
4	in	in	ADP	adposition	True
5	California	California	PROPN	proper noun	False
6	.	.	PUNCT	punctuation	False
7	It	-PRON-	PRON	pronoun	True
8	is	be	VERB	verb	True
9	a	a	DET	determiner	True
10	great	great	ADJ	adjective	False
11	university	university	NOUN	noun	False
12	.	.	PUNCT	punctuation	False

https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

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

python101.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

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

```
Stanford University ORG
California GPE
```

```
from spacy import displacy
text = "Stanford University is located in California. It is a great university."
doc = nlp(text)
displacy.render(doc, style="ent", jupyter=True)
```

Stanford University ORG is located in California GPE . It is a great university.

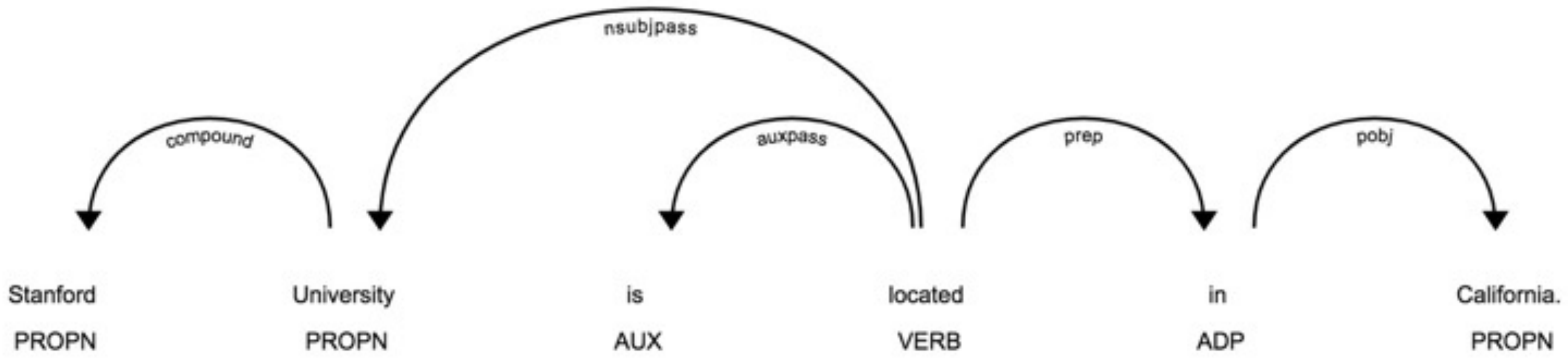
https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

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

```python
from spacy import displacy
text = "Stanford University is located in California. It is a great university."
doc = nlp(text)
displacy.render(doc, style="ent", jupyter=True)
displacy.render(doc, style="dep", jupyter=True)
```

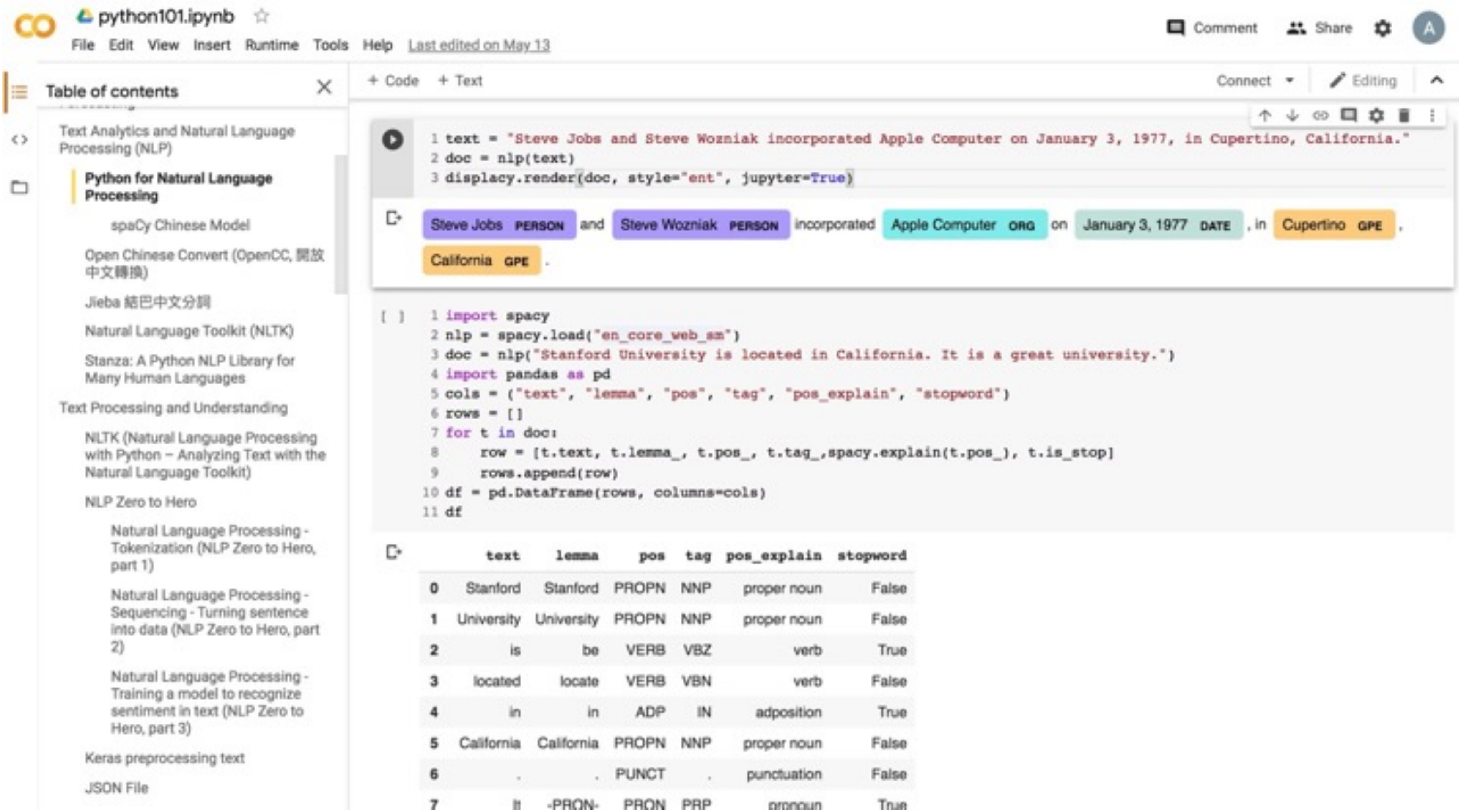
Stanford University ORG is located in California GPE . It is a great university.



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

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



https://tinyurl.com/aintpupython101

Summary

- Word Embeddings
- Recurrent Neural Networks for NLP
- Sequence-to-Sequence Models
- The Transformer Architecture
- Pretraining and Transfer Learning
- State of the art (SOTA)

References

- Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson.
- Aurélien Géron (2019), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd Edition, O'Reilly Media.
- Steven D'Ascoli (2022), Artificial Intelligence and Deep Learning with Python: Every Line of Code Explained For Readers New to AI and New to Python, Independently published.
- Nithin Buduma, Nikhil Buduma, Joe Papa (2022), Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms, 2nd Edition, O'Reilly Media.
- Dipanjan Sarkar (2019), Text Analytics with Python: A Practitioner's Guide to Natural Language Processing, Second Edition. APress. https://github.com/Apress/text-analytics-w-python-2e
- Benjamin Bengfort, Rebecca Bilbro, and Tony Ojeda (2018), Applied Text Analysis with Python, O'Reilly Media. https://www.oreilly.com/library/view/applied-text-analysis/9781491963036/
- Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, Ray Kurzweil (2018). Universal Sentence Encoder. arXiv:1803.11175.
- Yinfei Yang, Daniel Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, Gustavo Hernandez Abrego , Steve Yuan, Chris Tar, Yun-hsuan Sung, Ray Kurzweil (2019). Multilingual Universal Sentence Encoder for Semantic Retrieval.
- Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang (2020). "Pre-trained Models for Natural Language Processing: A Survey." arXiv preprint arXiv:2003.08271.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.
- Jay Alammar (2019), 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/
- Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- HuggingFace (2020), Transformers Notebook, https://huggingface.co/transformers/notebooks.html
- The Super Duper NLP Repo, https://notebooks.quantumstat.com/
- Min-Yuh Day (2022), Python 101, https://tinyurl.com/aintpupython101