

Artificial Intelligence

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>



Min-Yuh Day, Ph.D,
Associate Professor

Institute of Information Management, National Taipei University

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



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

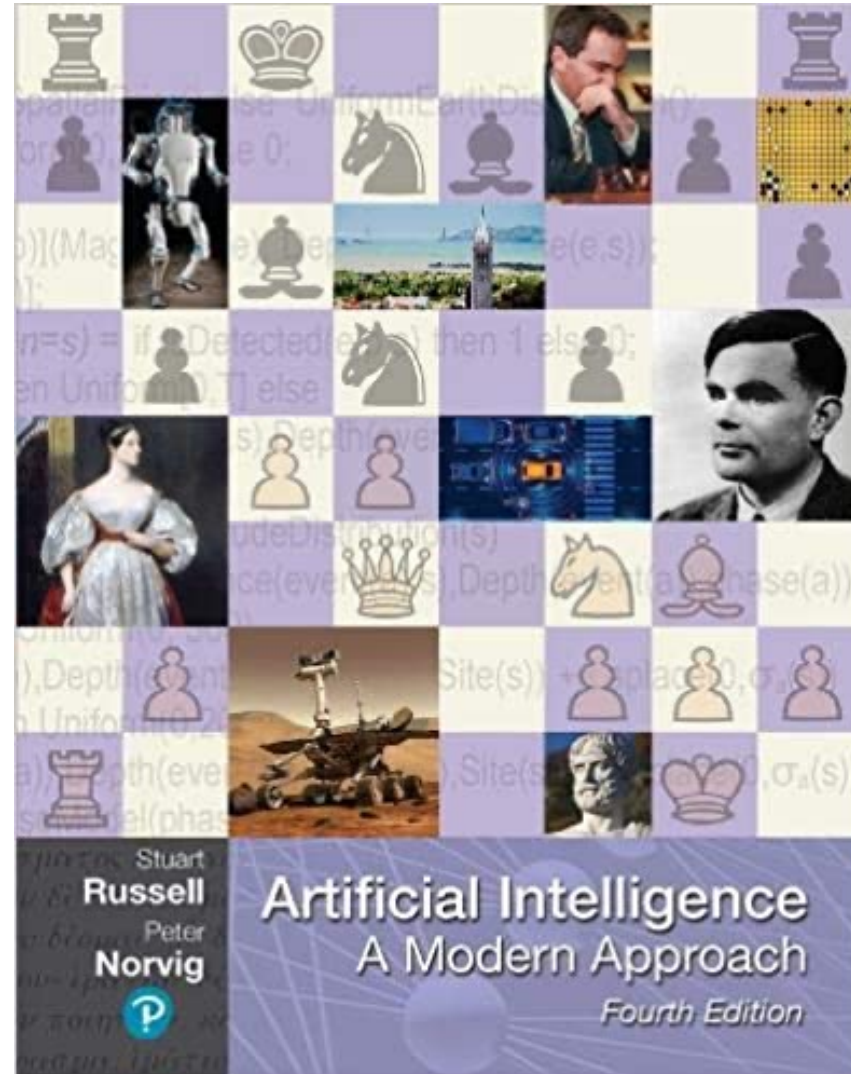
18 2023/01/11 Self-learning

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**

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

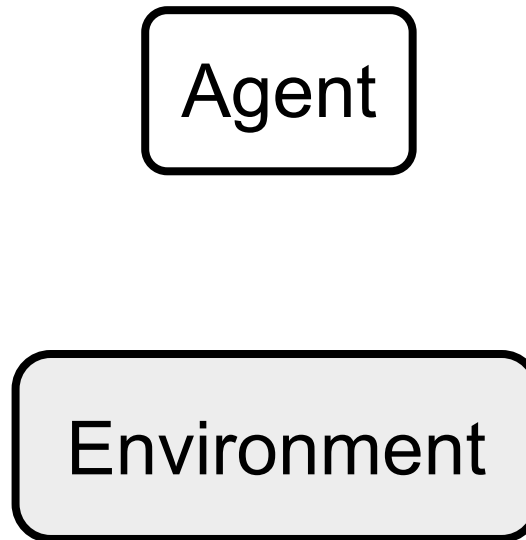
- **Language Models**
- **Grammar**
- **Parsing**
- **Augmented Grammars**
- **Complications of Real Natural Language**
- **Natural Language Tasks**

Artificial Intelligence:

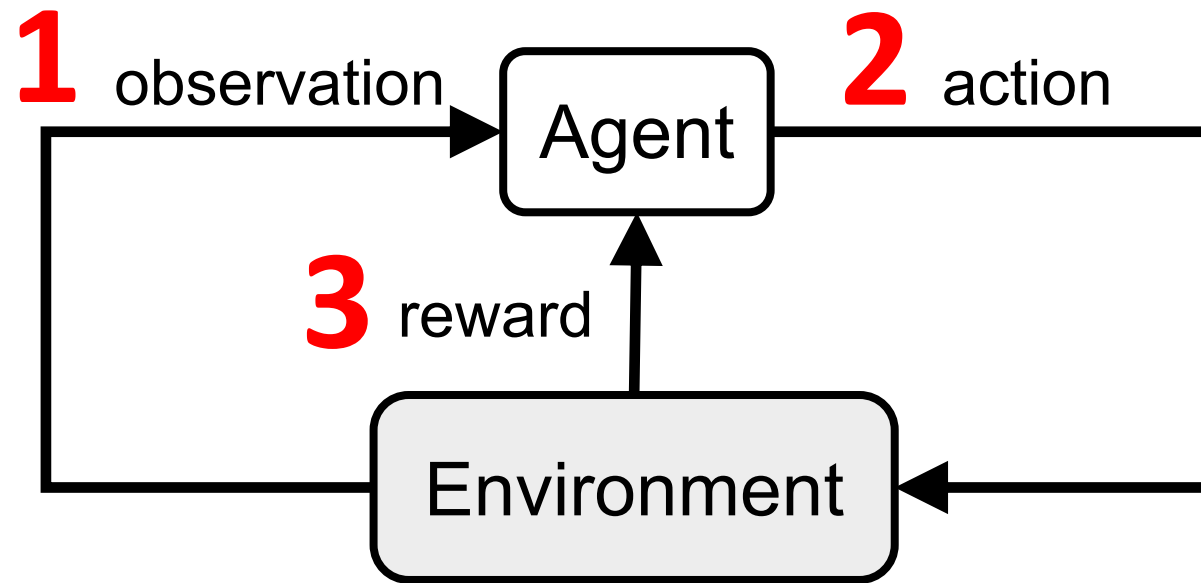
Deep Learning for Natural Language Processing

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

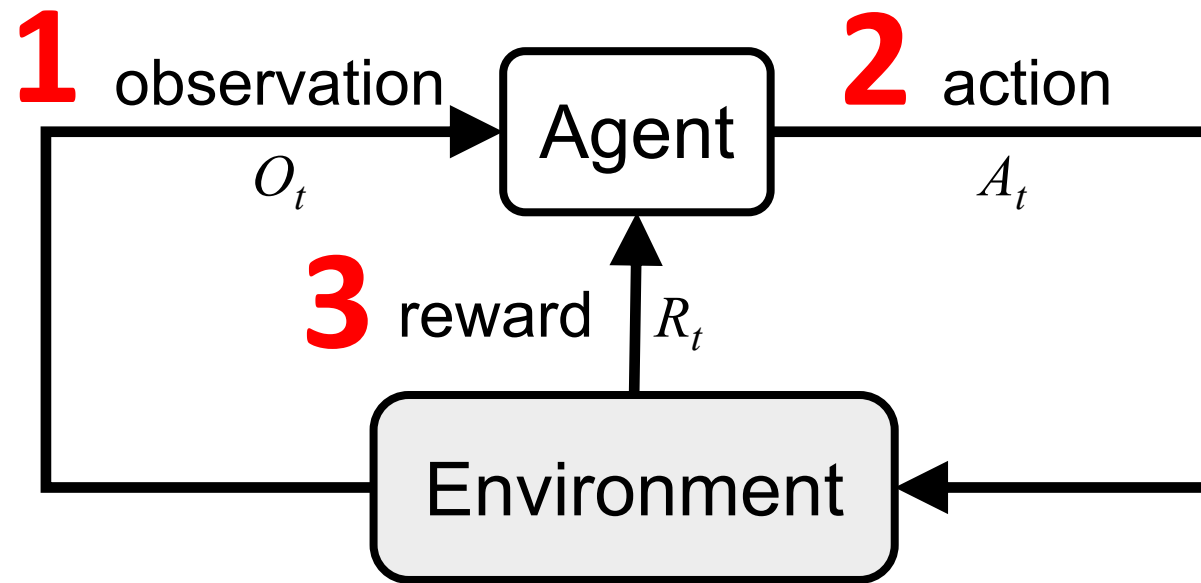
Reinforcement Learning (DL)



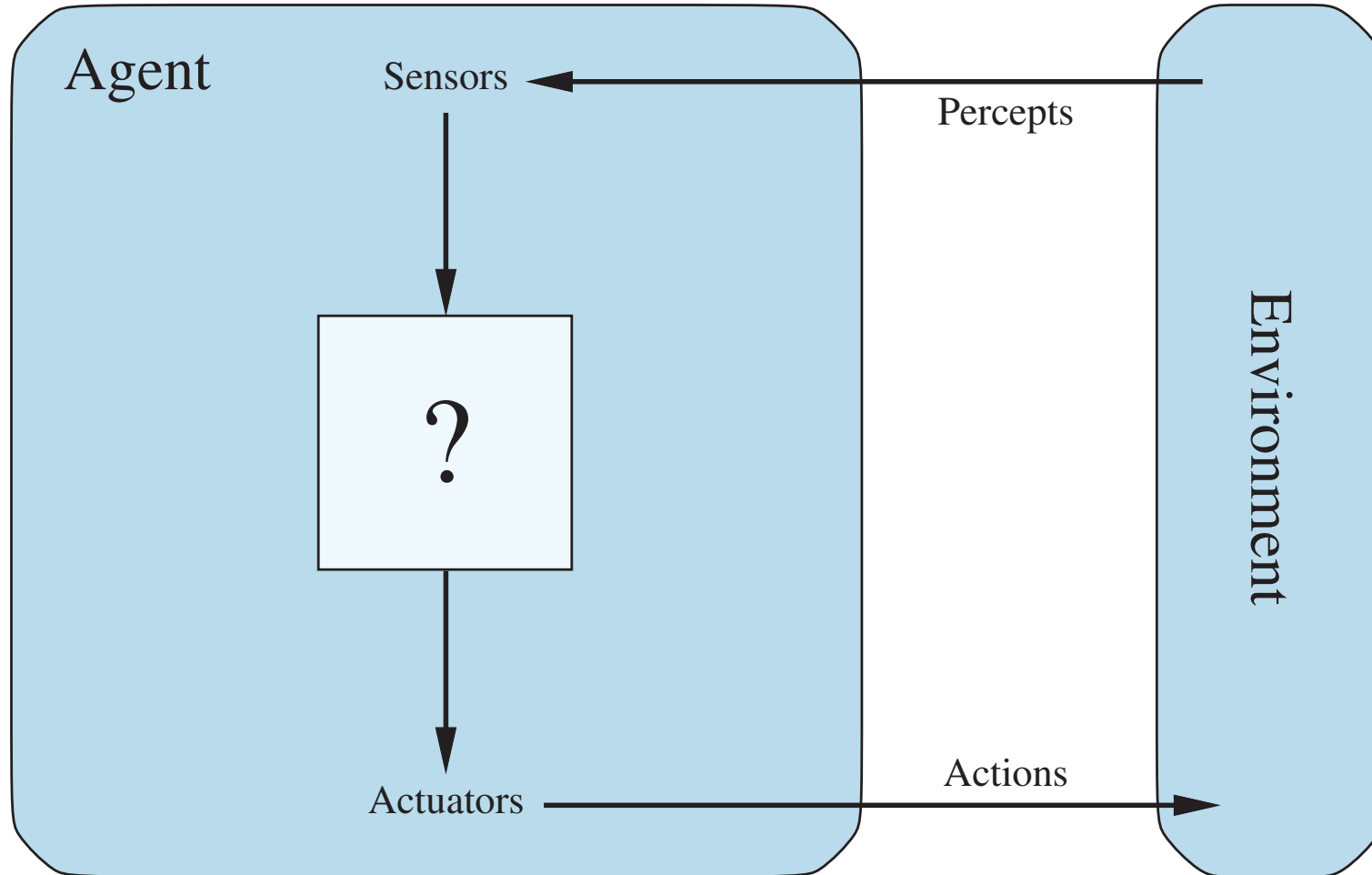
Reinforcement Learning (DL)



Reinforcement Learning (DL)



Agents interact with environments through sensors and actuators



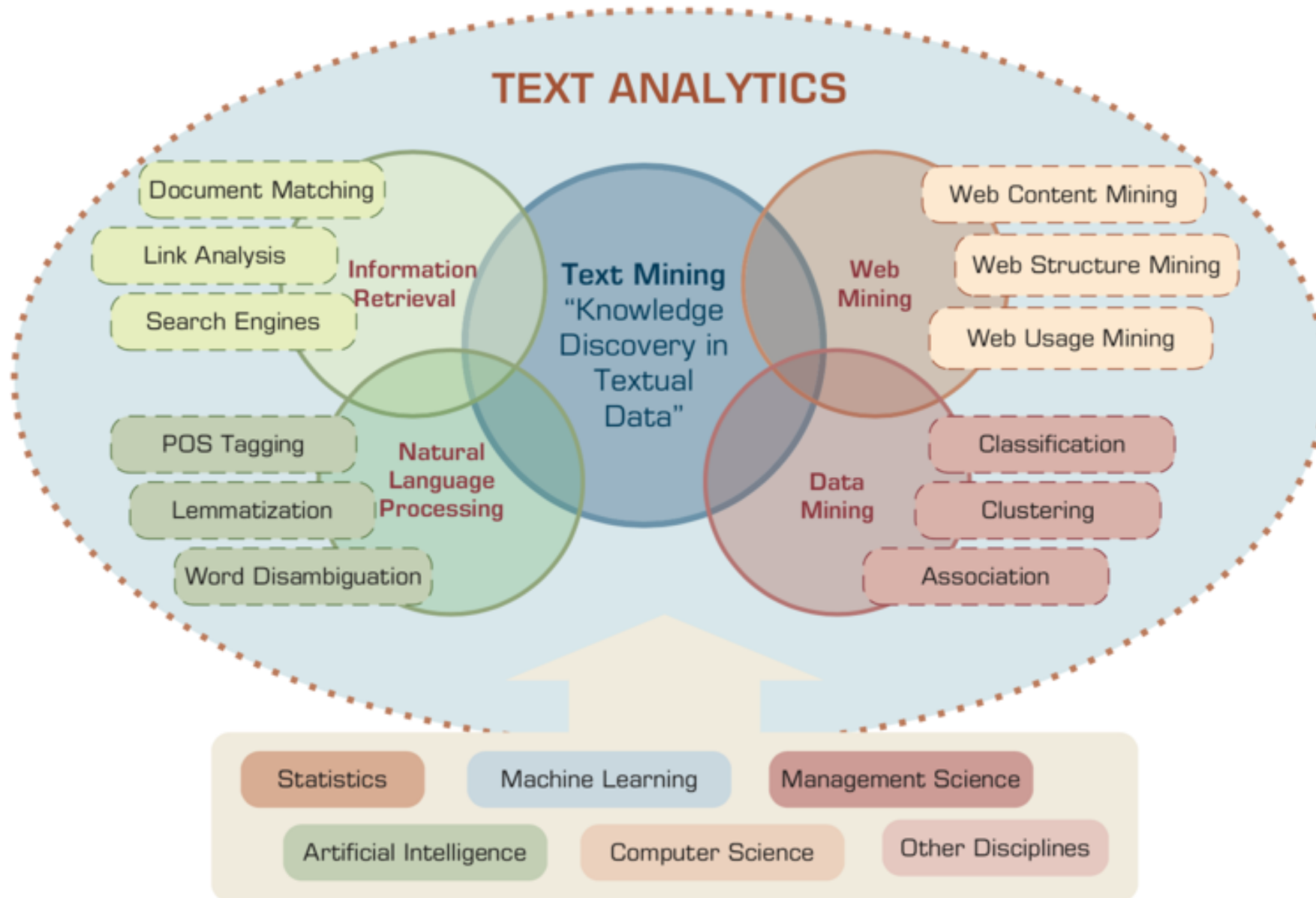
AI Acting Humanly: The Turing Test Approach

(Alan Turing, 1950)

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

Deep Learning for Natural Language Processing

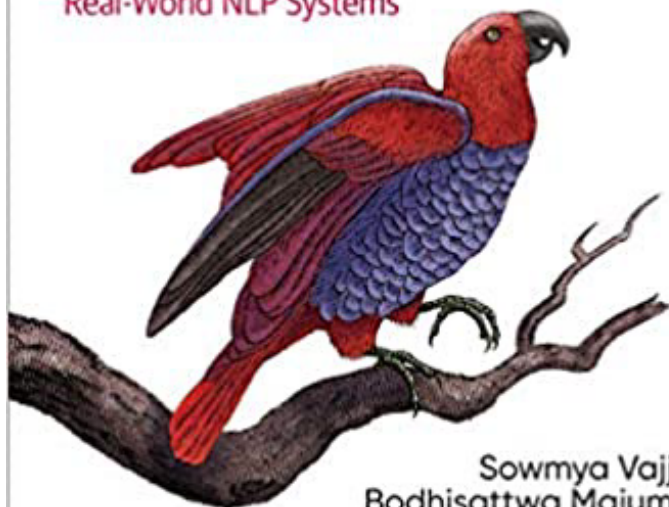
AI for Text Analytics



O'REILLY®

Practical Natural Language Processing

A Comprehensive Guide to Building Real-World NLP Systems



Sowmya Vajjala,
Bodhisattwa Majumder,
Anuj Gupta & Harshit Surana

FOUNDATIONS

*Covered in
Chapters 1 to 3*



ML for NLP



NLP Pipelines



Data
Gathering



Multilingual
NLP



Text
Representation

CORE TASKS

*Covered in
Chapters 3 to 7*



Text
Classification



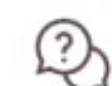
Information
Extraction



Conversational
Agents



Information
Retrieval



Question
Answering

GENERAL APPLICATIONS

*Covered in
Chapters 4 to 7*



Spam
Classification



Calendar Event
Extraction



Personal
Assistants



Search
Engines

JEOPARDY!

Jeopardy!

INDUSTRY SPECIFIC

*Covered in
Chapters 8 to 10*



Social Media
Analysis



Retail Data
Extraction



Health Records
Analysis



Financial
Analysis



Legal Entity
Extraction

AI PROJECT PLAYBOOK

*Covered in
Chapters 2 & 11*



Project
Processes



Best
Practices



Model
Iterations

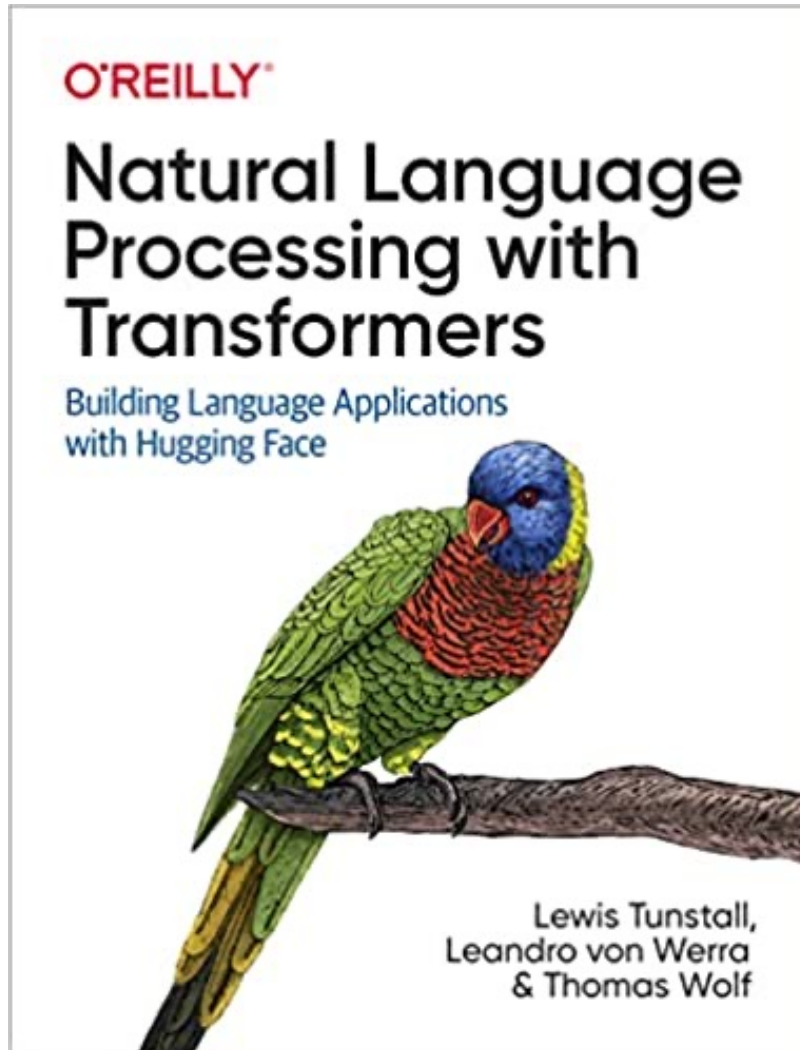


MLOps



AI Teams
& Hiring

NLP with Transformers Github Notebooks



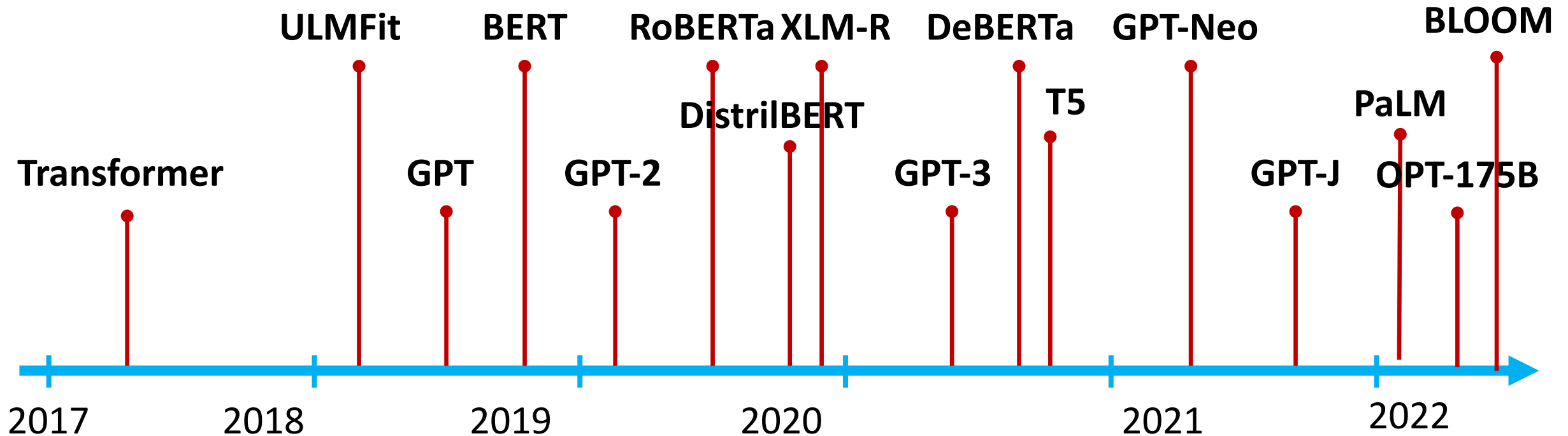
Running on a cloud platform

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

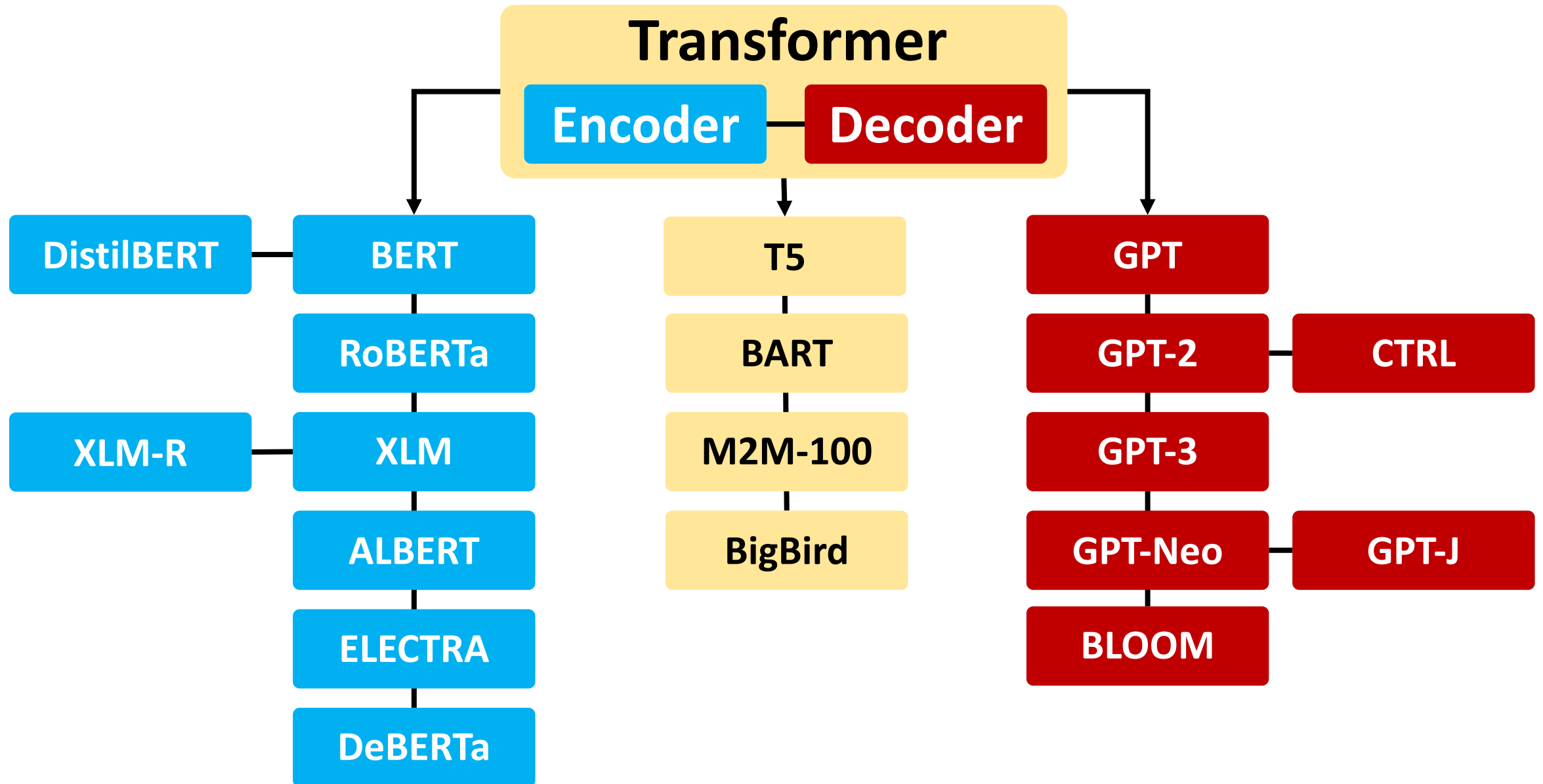
Chapter	Colab	Kaggle	Gradient	Studio Lab
Introduction	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Text Classification	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Transformer Anatomy	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Multilingual Named Entity 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!

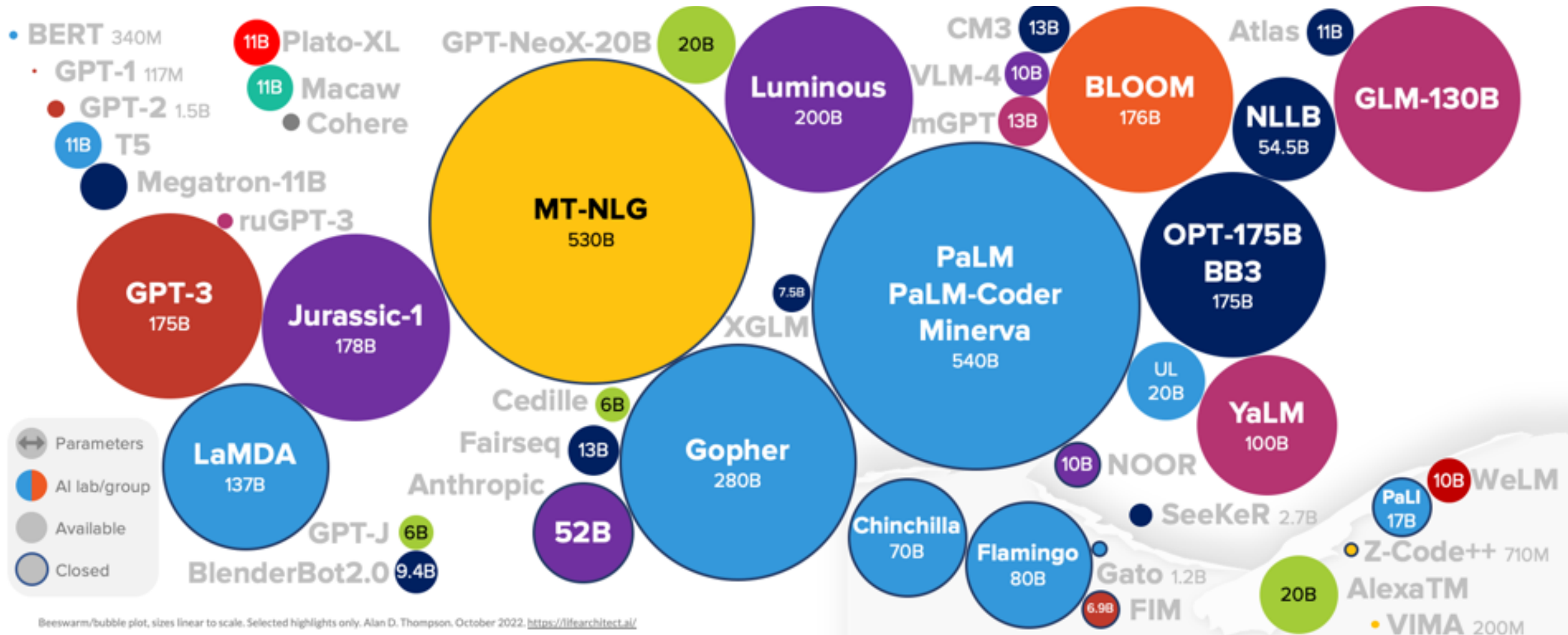
The Transformers Timeline



Transformer Models

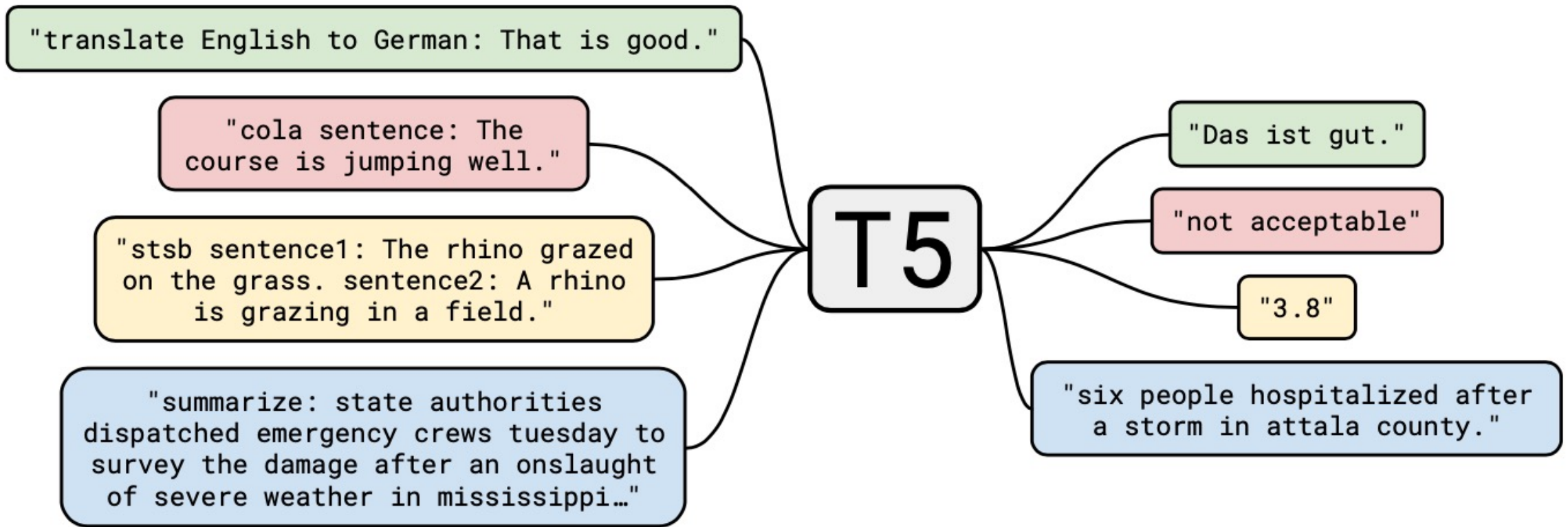


Language Models Sizes (GPT-3, PaLM, BLOOM)



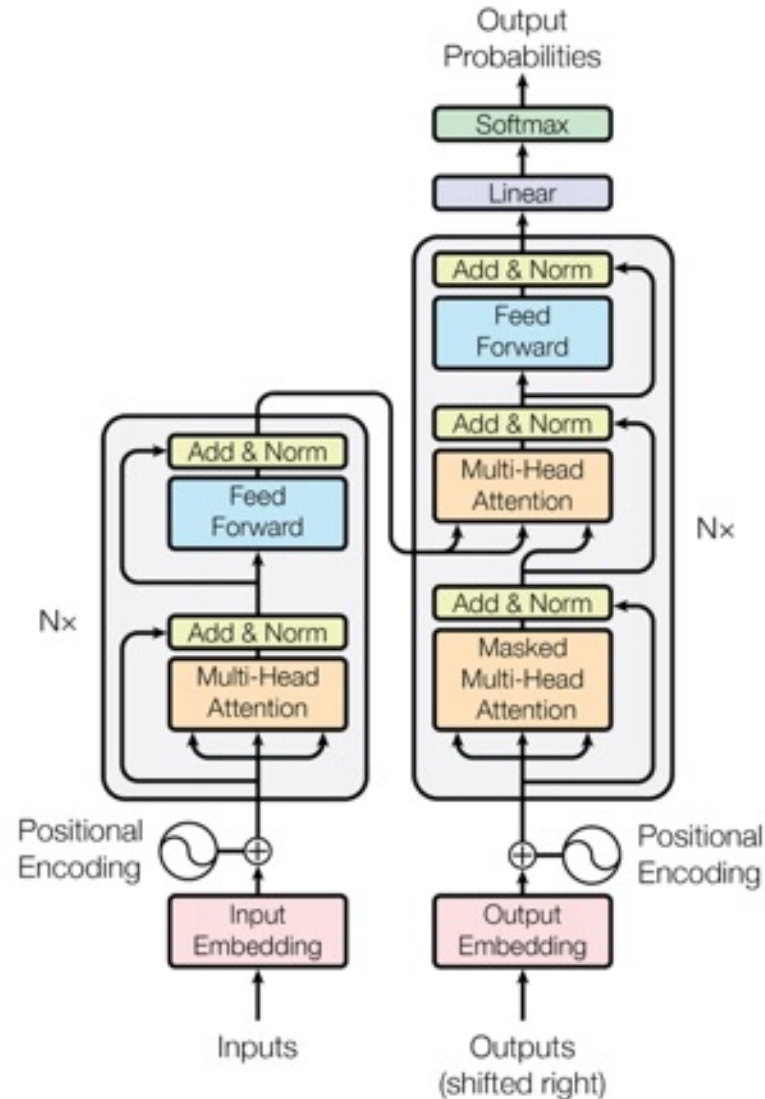
T5

Text-to-Text Transfer Transformer



Transformer (Attention is All You Need)

(Vaswani et al., 2017)

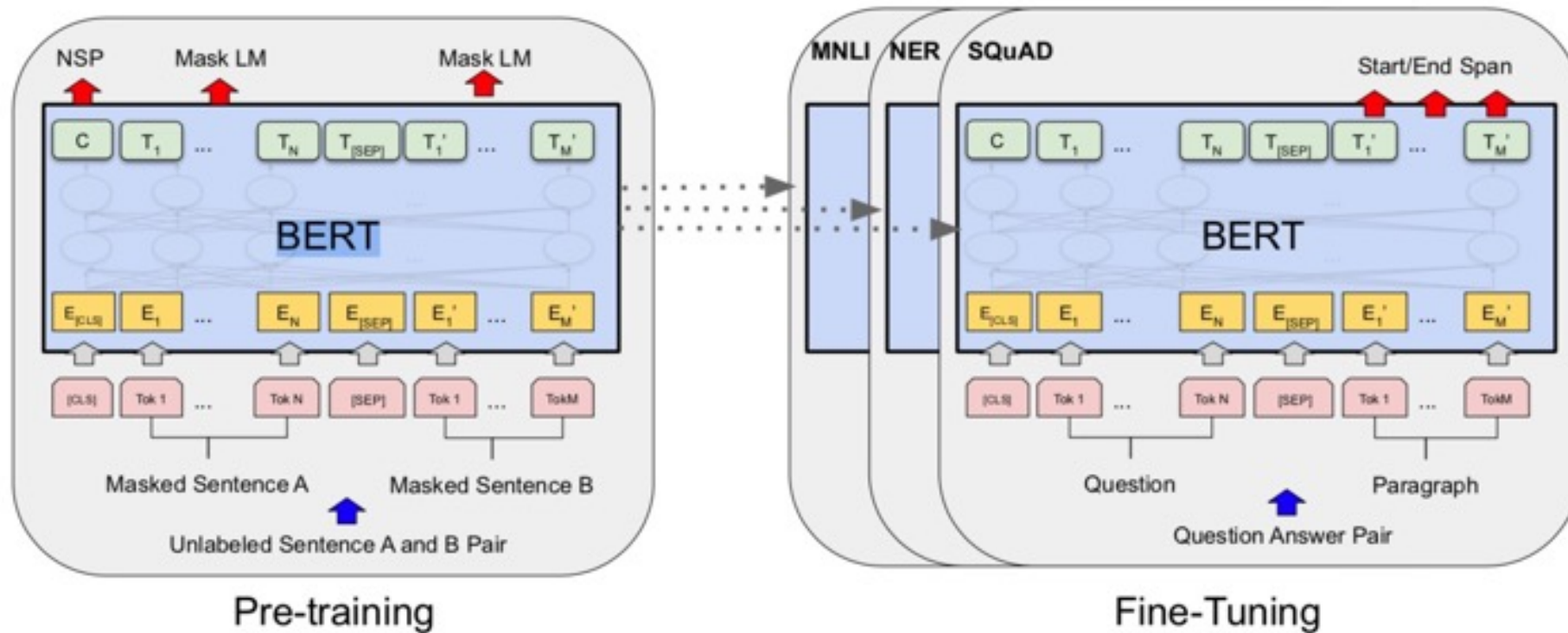


Source: Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In *Advances in neural information processing systems*, pp. 5998-6008. 2017.

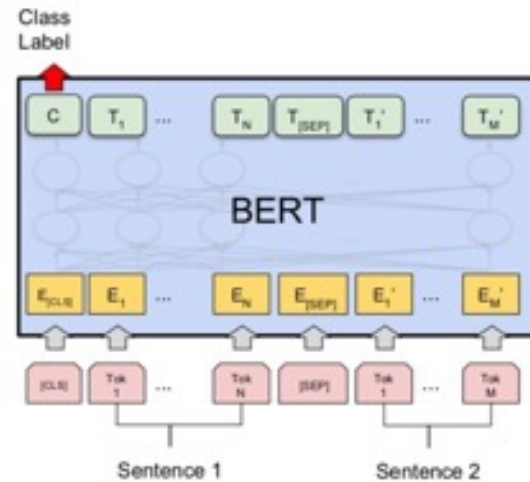
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT (Bidirectional Encoder Representations from Transformers)

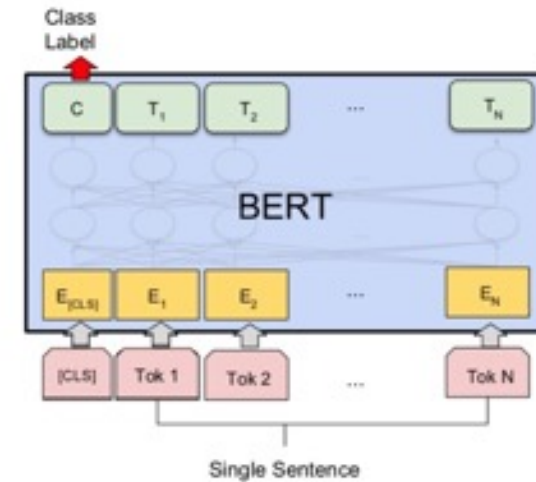
Overall pre-training and fine-tuning procedures for BERT



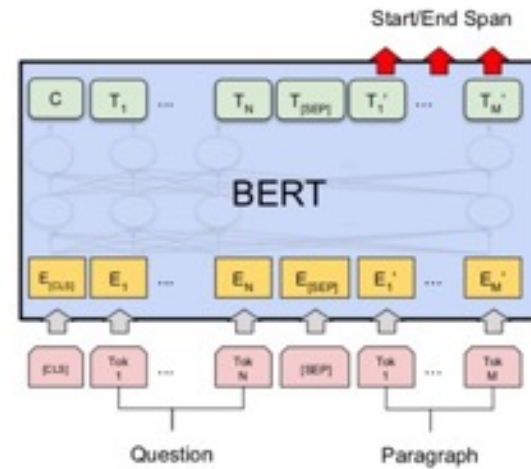
Fine-tuning BERT on Different Tasks



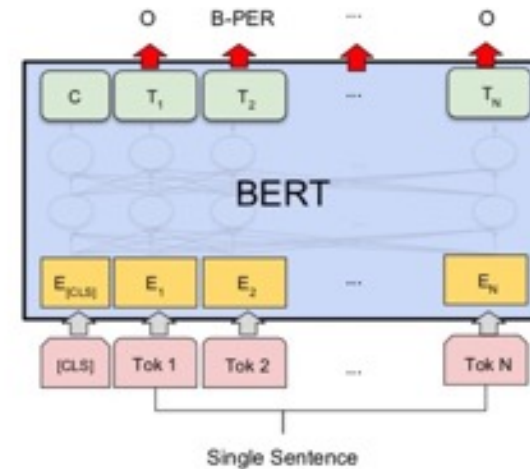
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



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



(c) Question Answering Tasks:
SQuAD v1.1



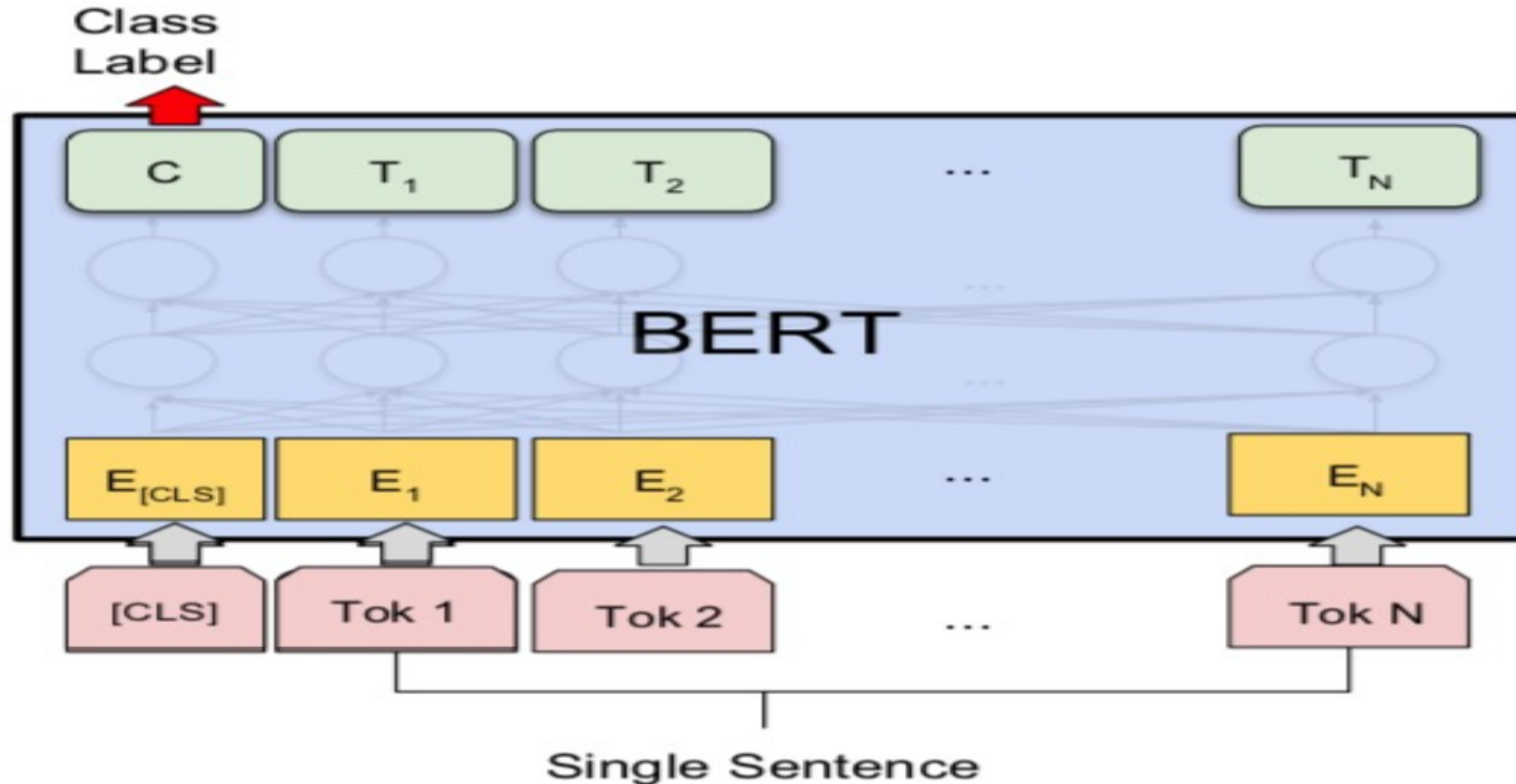
(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

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

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

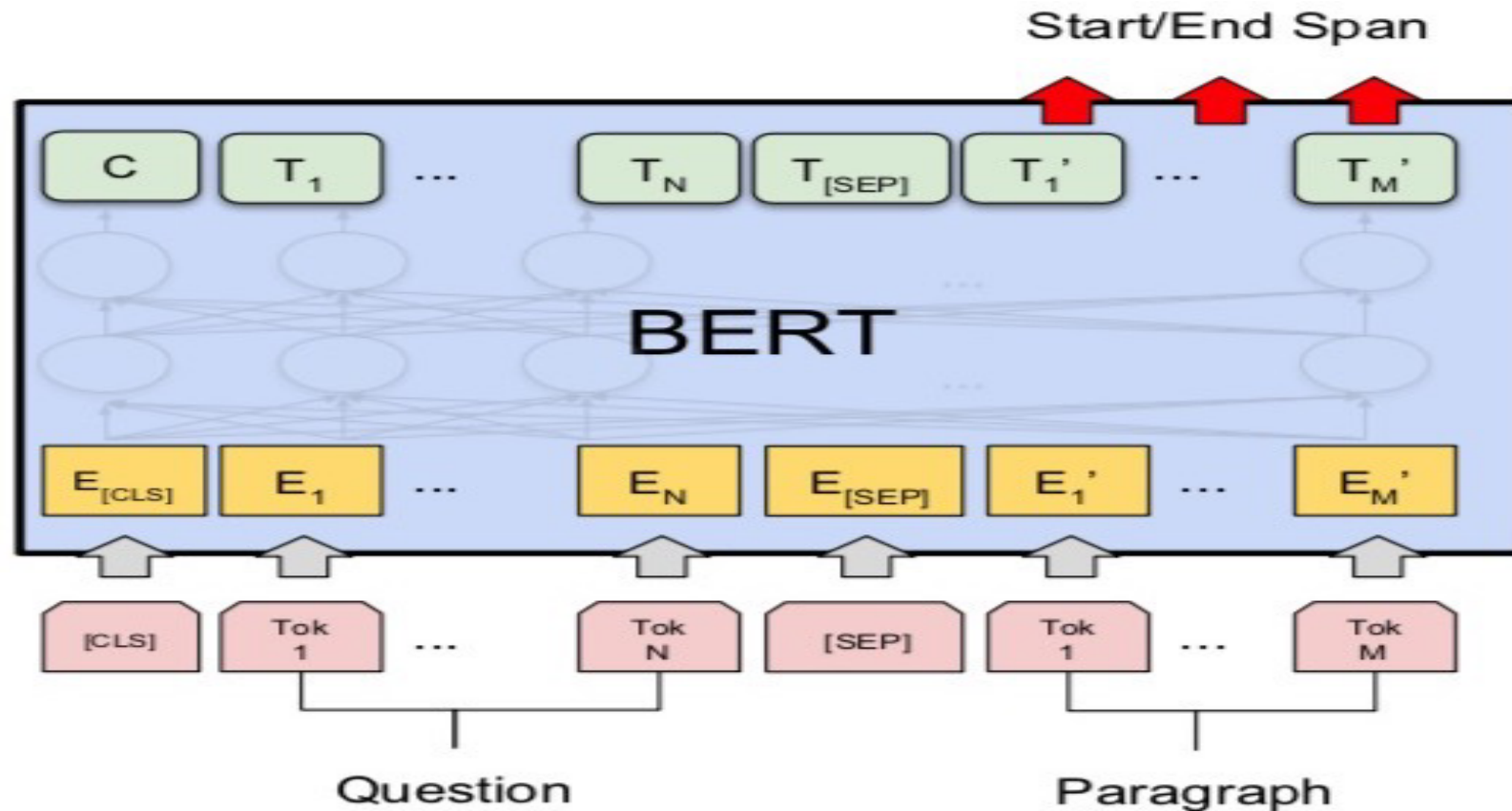
Sentiment Analysis:

Single Sentence Classification



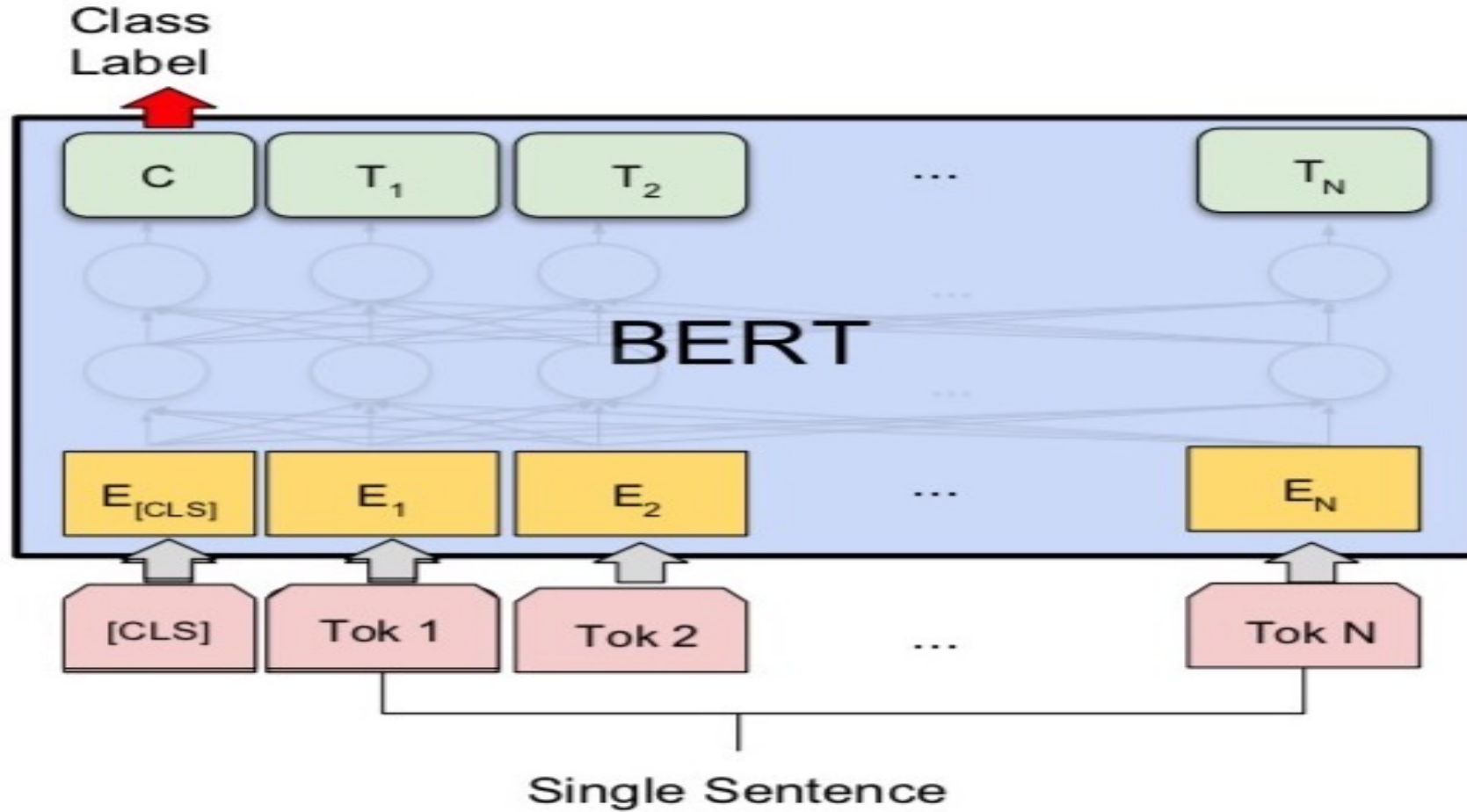
(b) Single Sentence Classification Tasks:
SST-2, CoLA

Fine-tuning BERT on Question Answering (QA)



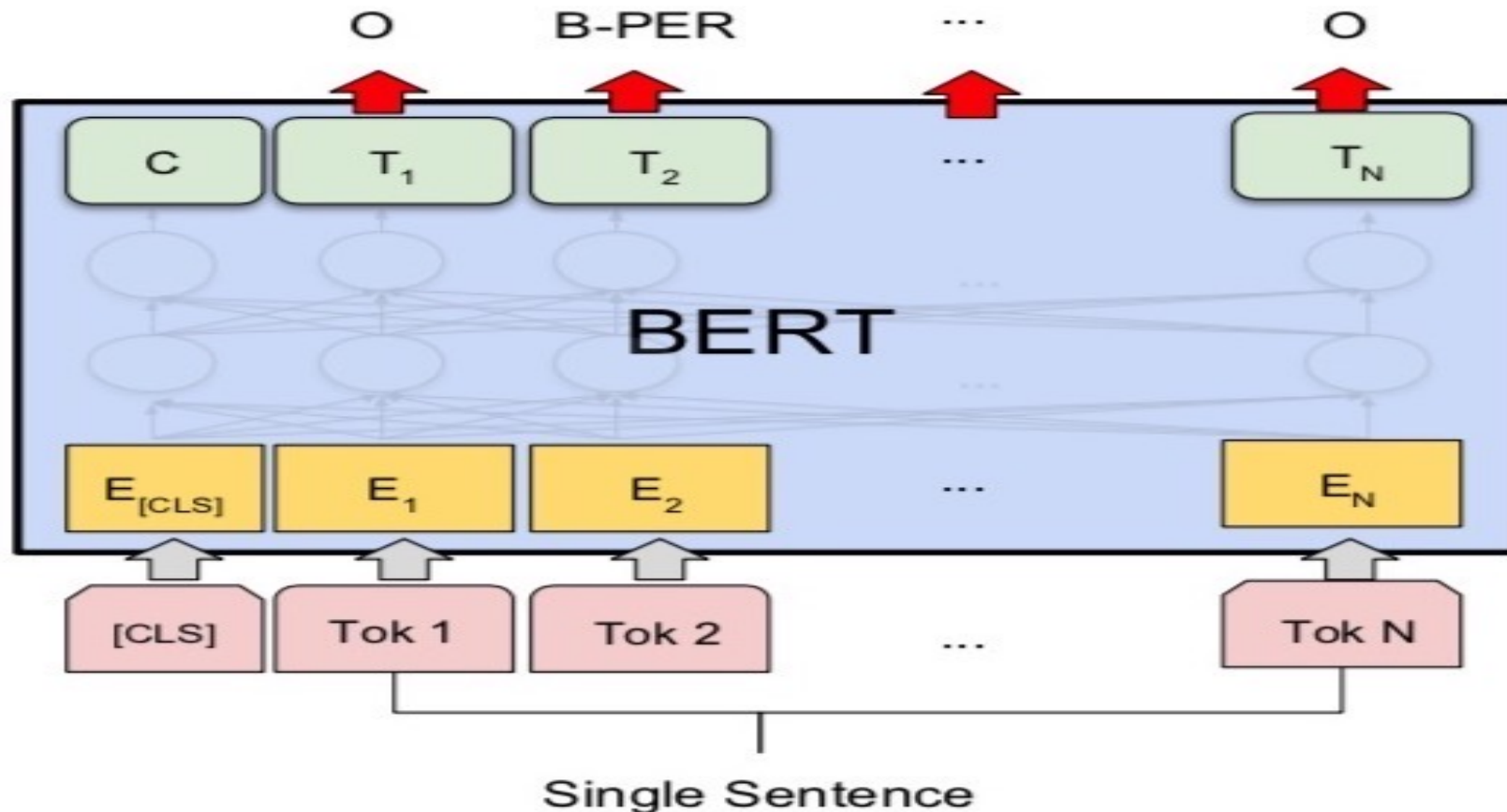
(c) Question Answering Tasks:
SQuAD v1.1

Fine-tuning BERT on Dialogue Intent Detection (ID; Classification)



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

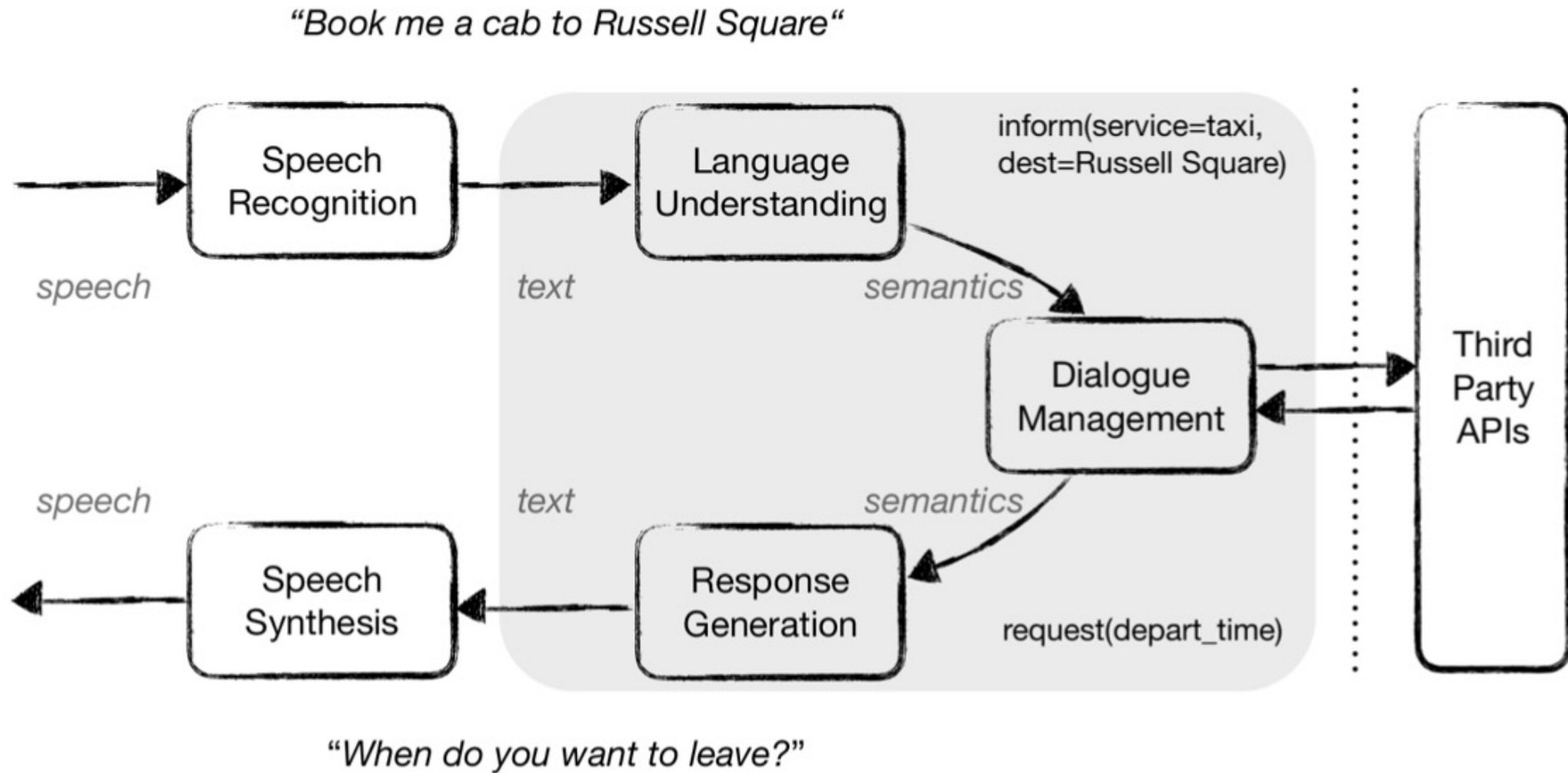
Fine-tuning BERT on Dialogue Slot Filling (SF)



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

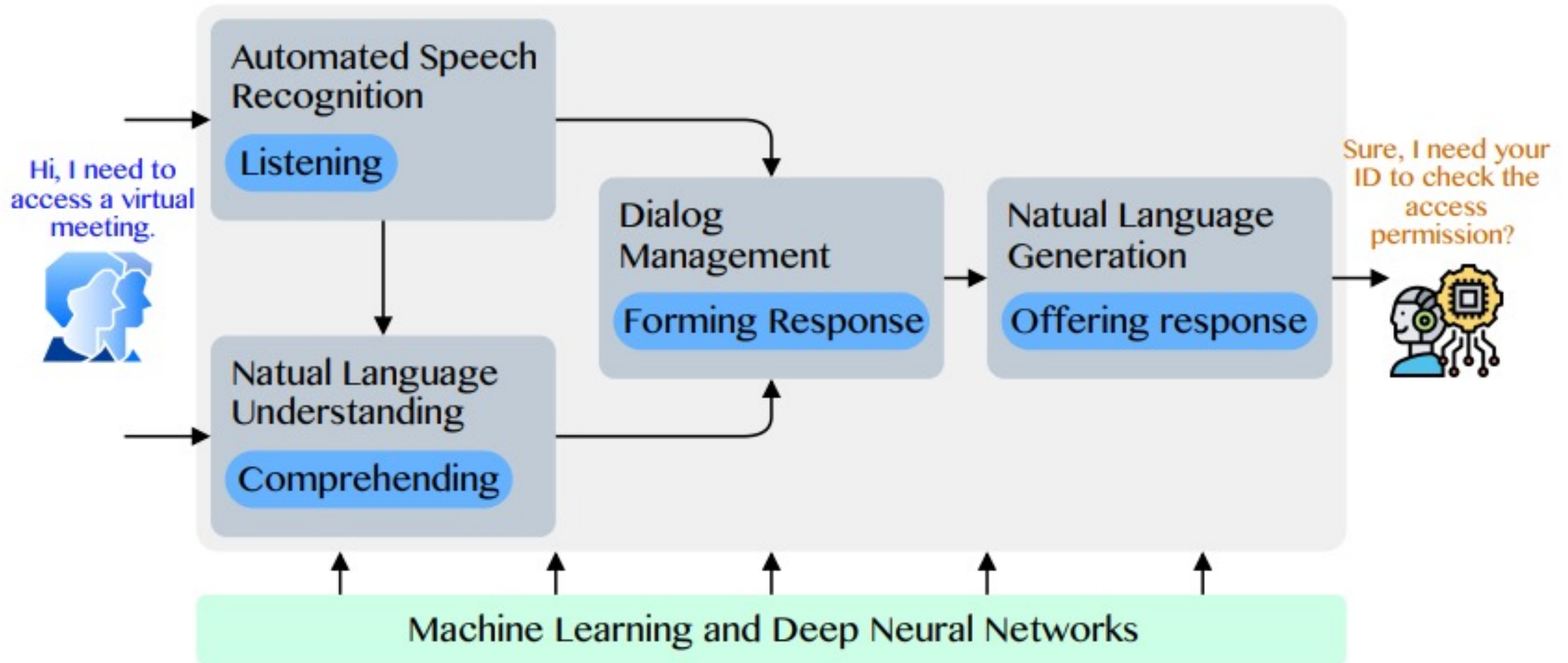
Task-Oriented Dialogue (ToD) System

Speech, Text, NLP



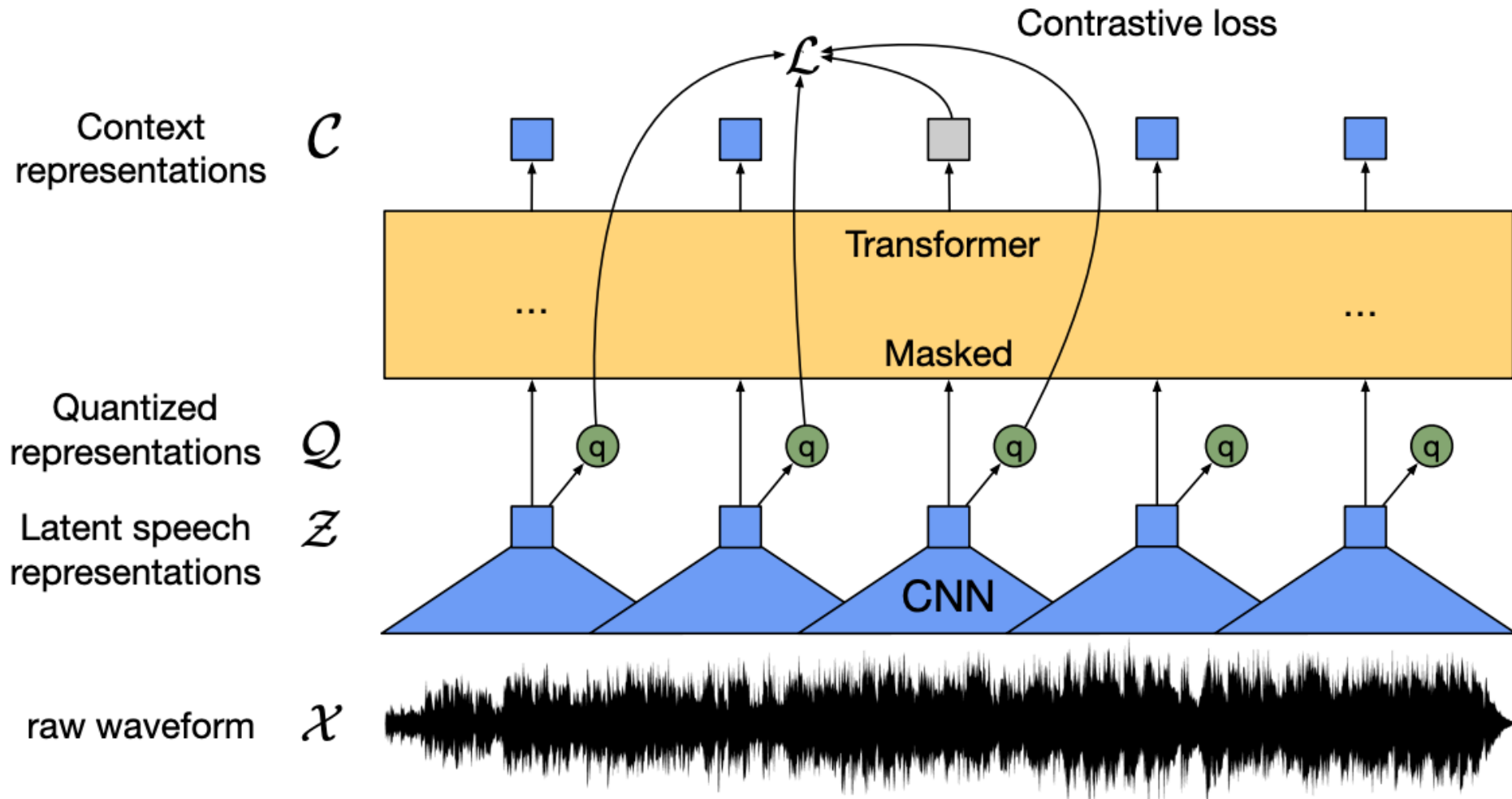
Conversational AI

to deliver contextual and personal experience to users



wav2vec 2.0:

A framework for self-supervised learning of speech representations

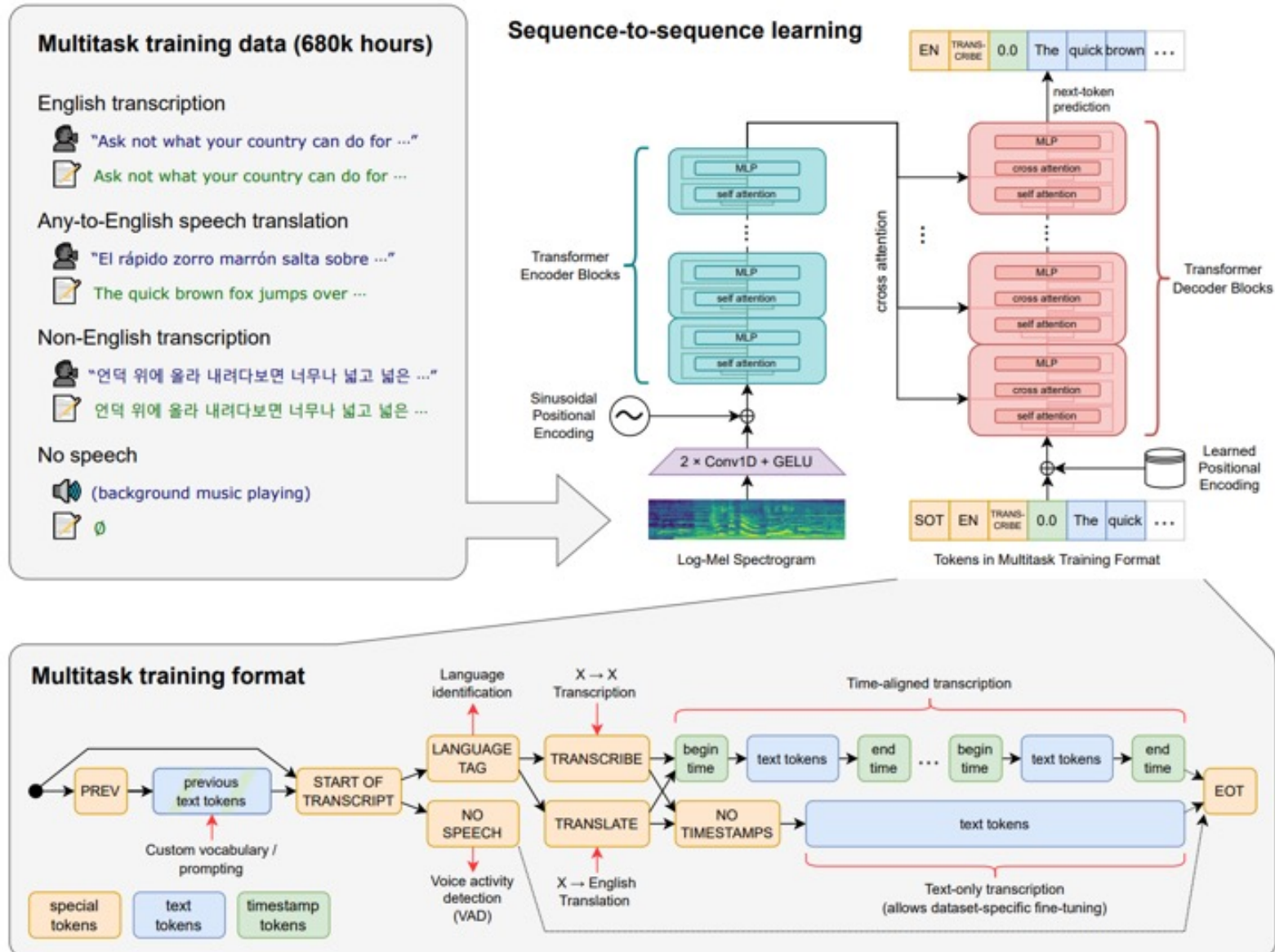


Source: Baevski, Alexei, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli.

"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



Microsoft Azure Text to Speech (TTS)

Text

SSML

You can replace this text with any text you wish. You can either write in this text box or paste your own text here.

Try different languages and voices. Change the speed and the pitch of the voice. You can even tweak the SSML (Speech Synthesis Markup Language) to control how the different sections of the text sound. Click on SSML above to give it a try!

Enjoy using Text to Speech!

Language

English (United States) ▾

Voice

Jenny (Neural) ▾

Speaking style

General ▾

Speaking speed: 1.00

Pitch: 0.00

Play

Hugging Face



Hugging Face

🔍 Search models, datasets, spaces



Models



Datasets



Spaces



Docs



Solutions

Pricing



Log In

Sign Up



The AI community building the future.

Build, train and deploy state of the art models powered by
the reference open source in machine learning.



Star

58,696

<https://huggingface.co/>

BLOOM

BigScience Large Open-science Open-access Multilingual Language Model



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



Hugging Face

Search models

Models

Datasets

Spaces

Docs

Solutions

Pricing



Spaces: openai/whisper



422

Running

App

Files



Community 49

Whisper

Whisper is a general-purpose speech recognition model. It is trained on a large dataset of diverse audio and is also a multi-task model that can perform multilingual speech recognition as well as speech translation and language identification. This demo cuts audio after around 30 secs.

You can skip the queue by using google colab for the space:



Open in Colab



0:05 / 0:05



Transcribe

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."""
```


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

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

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

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

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.

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.

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

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

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

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 }
```


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

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'}]]
```


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

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

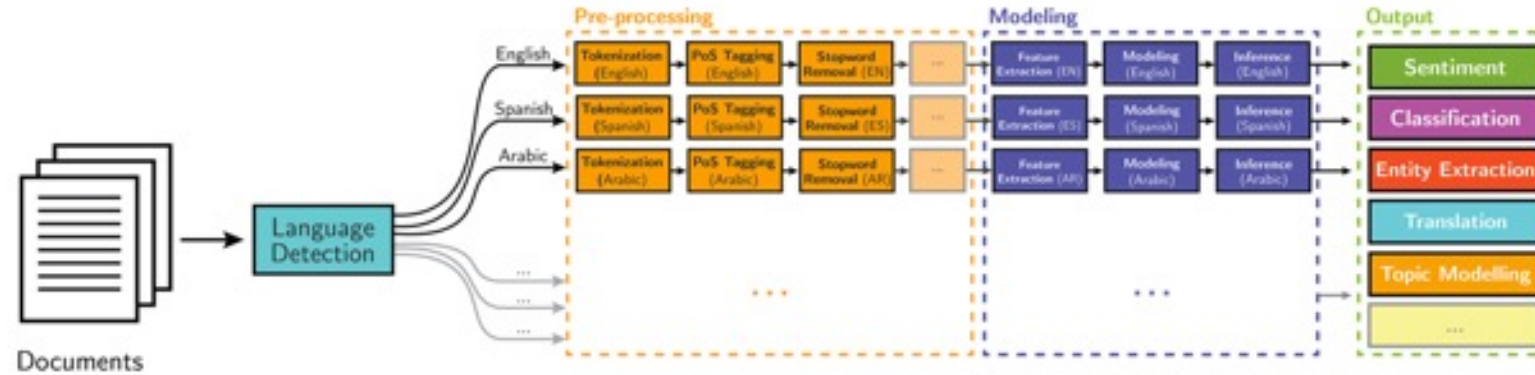
Text2Text Generation

```
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")
```

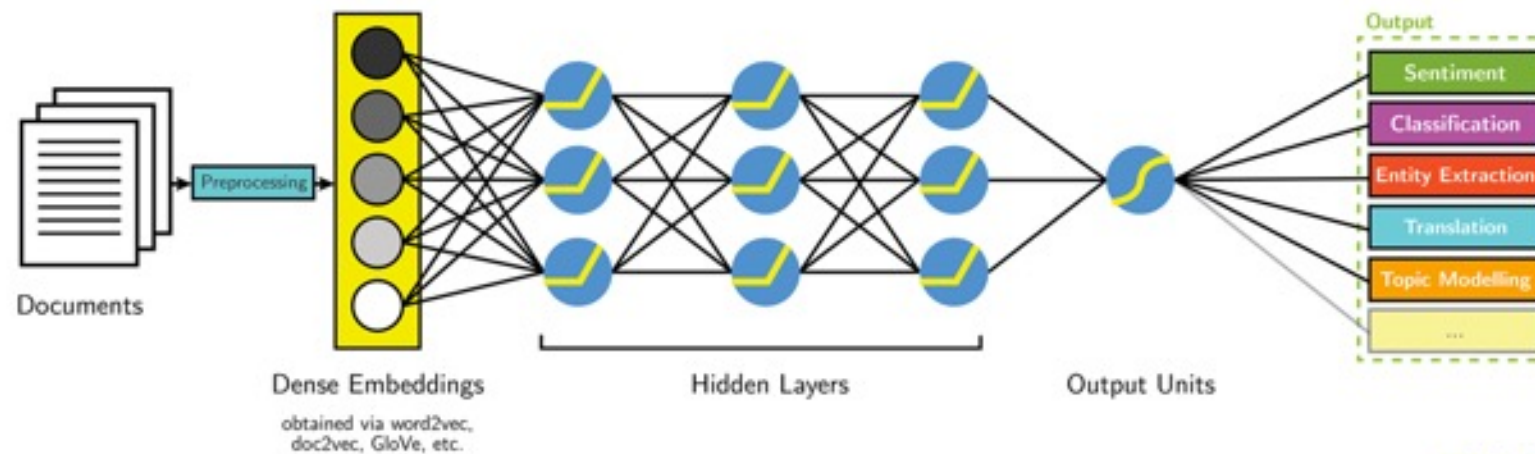
```
[{'generated_text': 'the answer to life, the universe and everything'}]
```


NLP

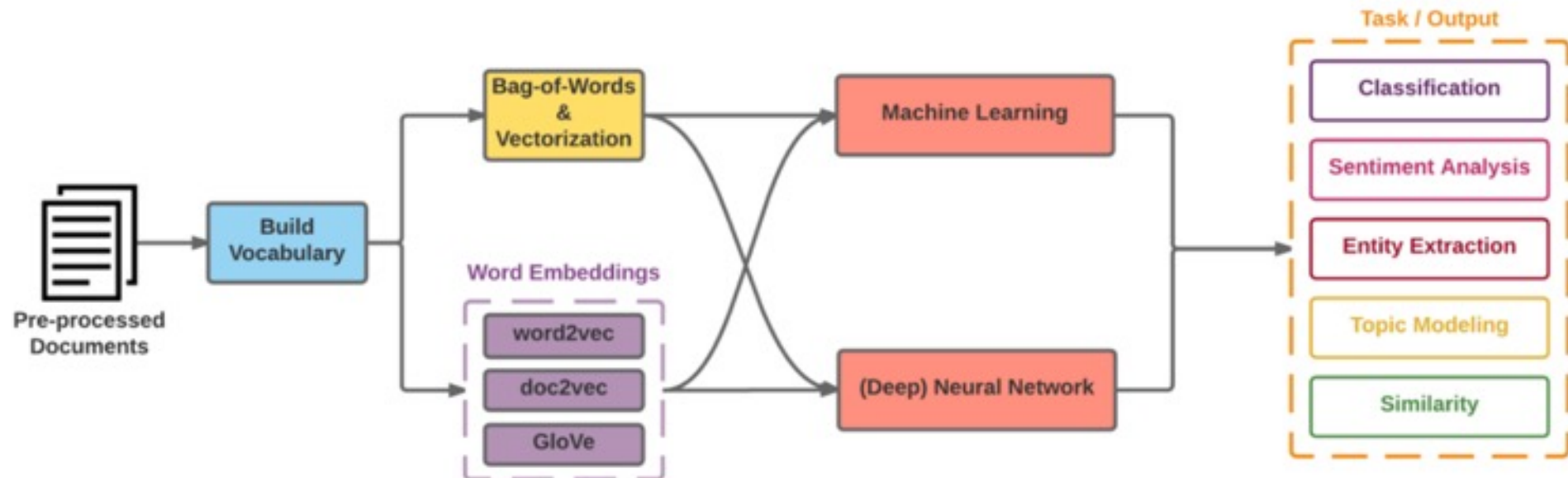
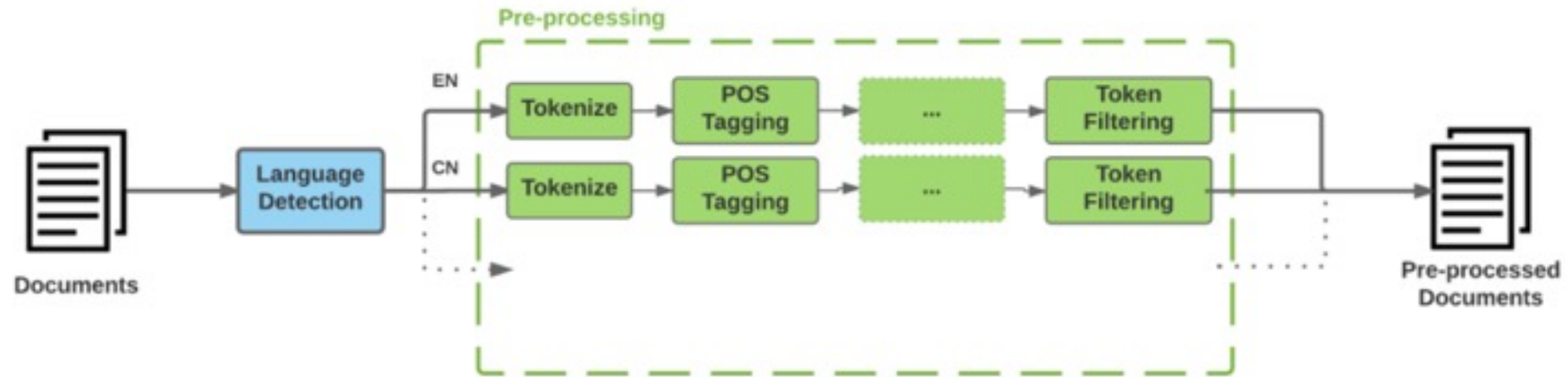
Classical NLP



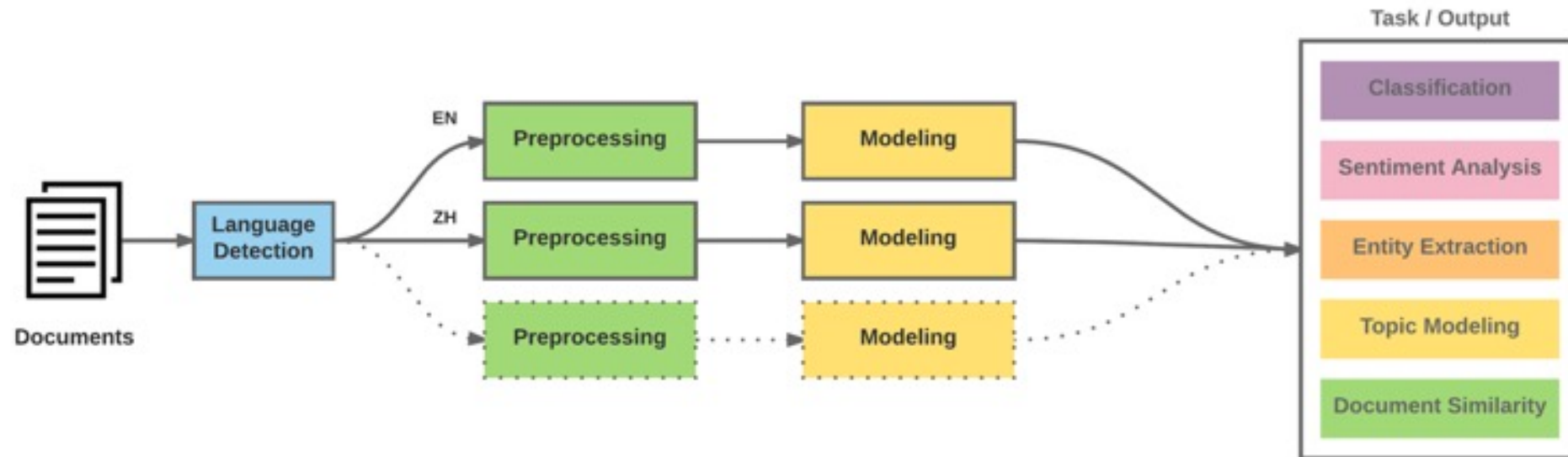
Deep Learning-based NLP



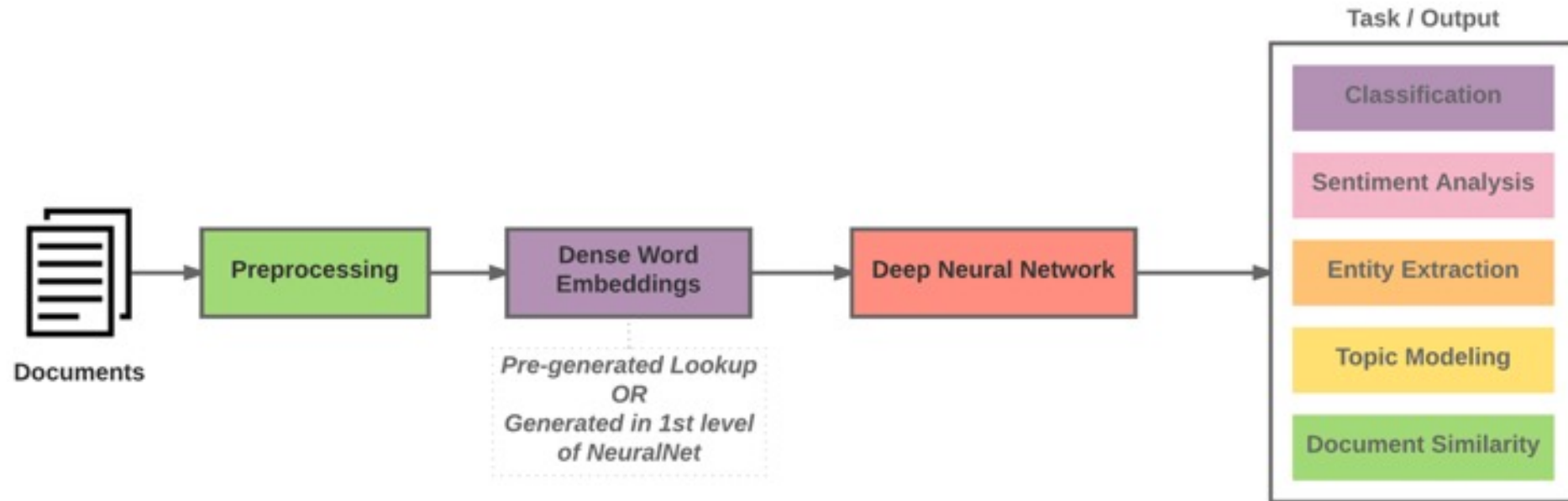
Modern NLP Pipeline



Modern NLP Pipeline



Deep Learning NLP



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

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' =

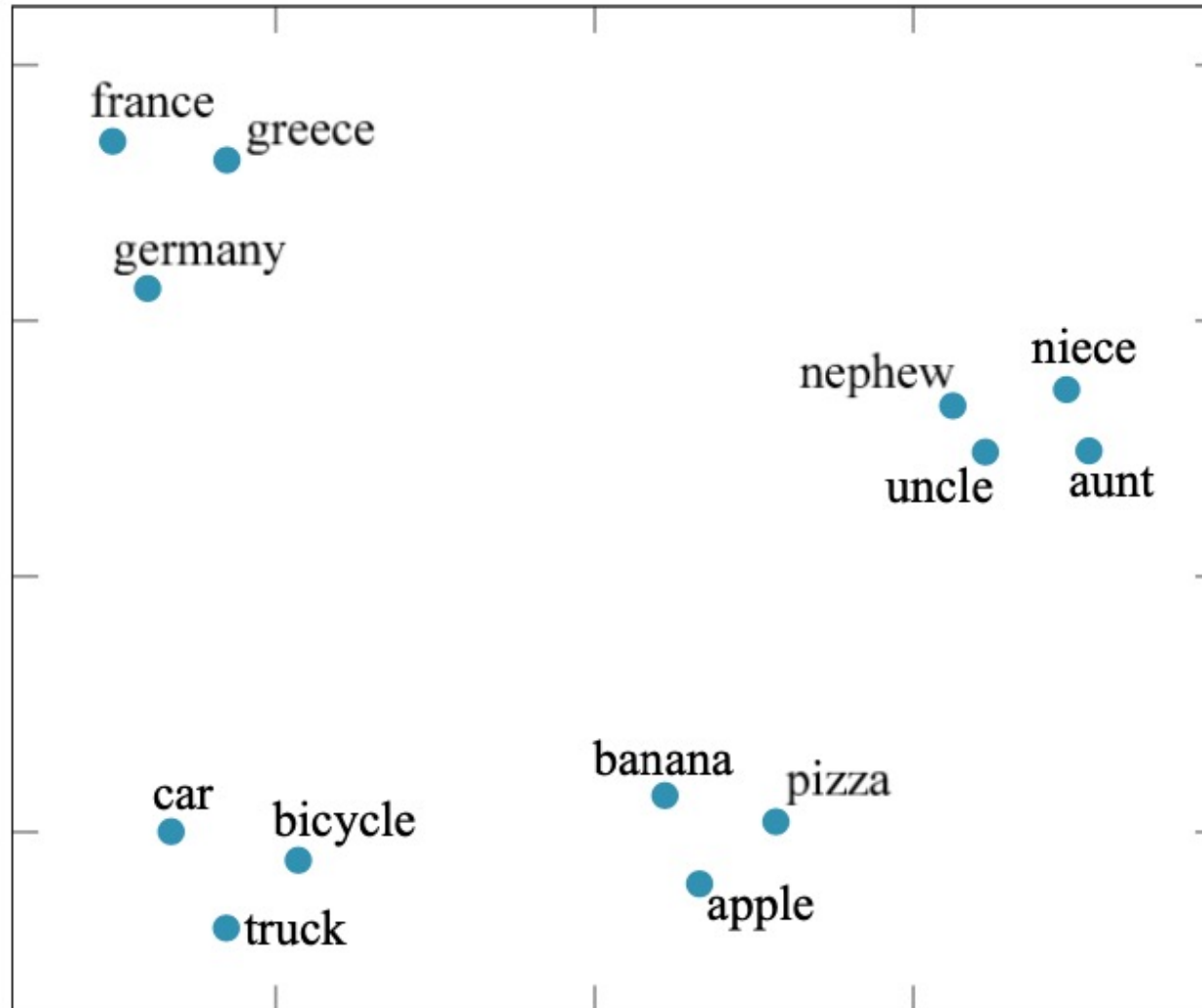
The	1	[[0, 1, 0, 0, 0, 0, 0],
mouse	2		[0, 0, 1, 0, 0, 0, 0],
ran	3		[0, 0, 0, 1, 0, 0, 0],
up	4		[0, 0, 0, 0, 1, 0, 0],
the	1		[0, 1, 0, 0, 0, 0, 0],
clock	5		[0, 0, 0, 0, 0, 1, 0]]

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

Word embedding

GloVe (trained on 6 billion words of text)

100-dimensional word vectors are projected down onto two dimensions

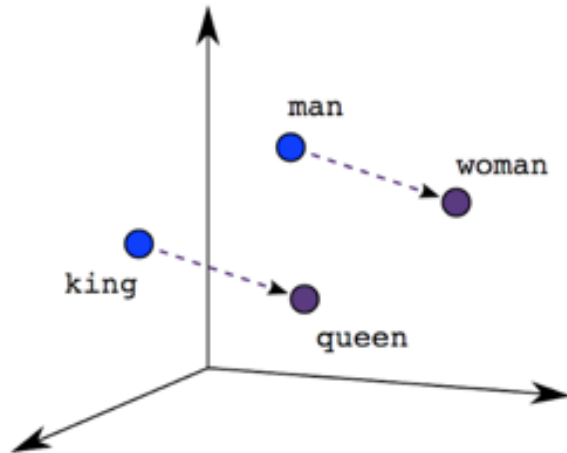


Word Embedding model

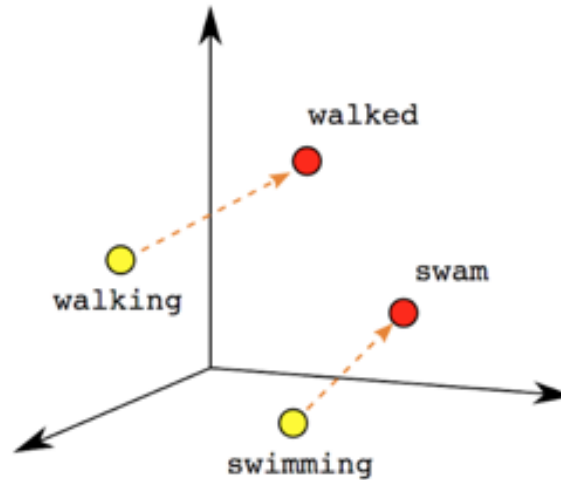
answer the question “A is to B as C is to [what]?”

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

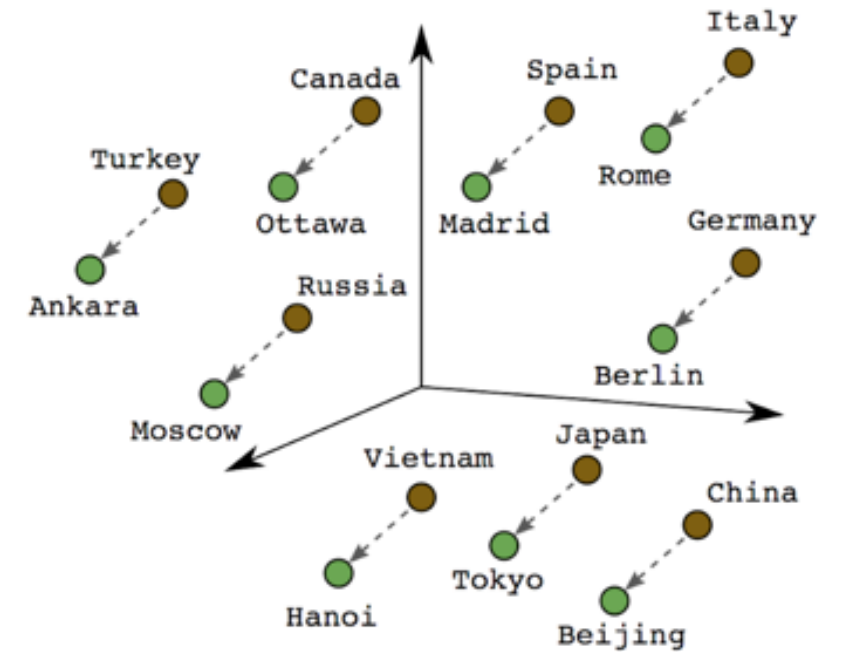
Word embeddings



Male-Female

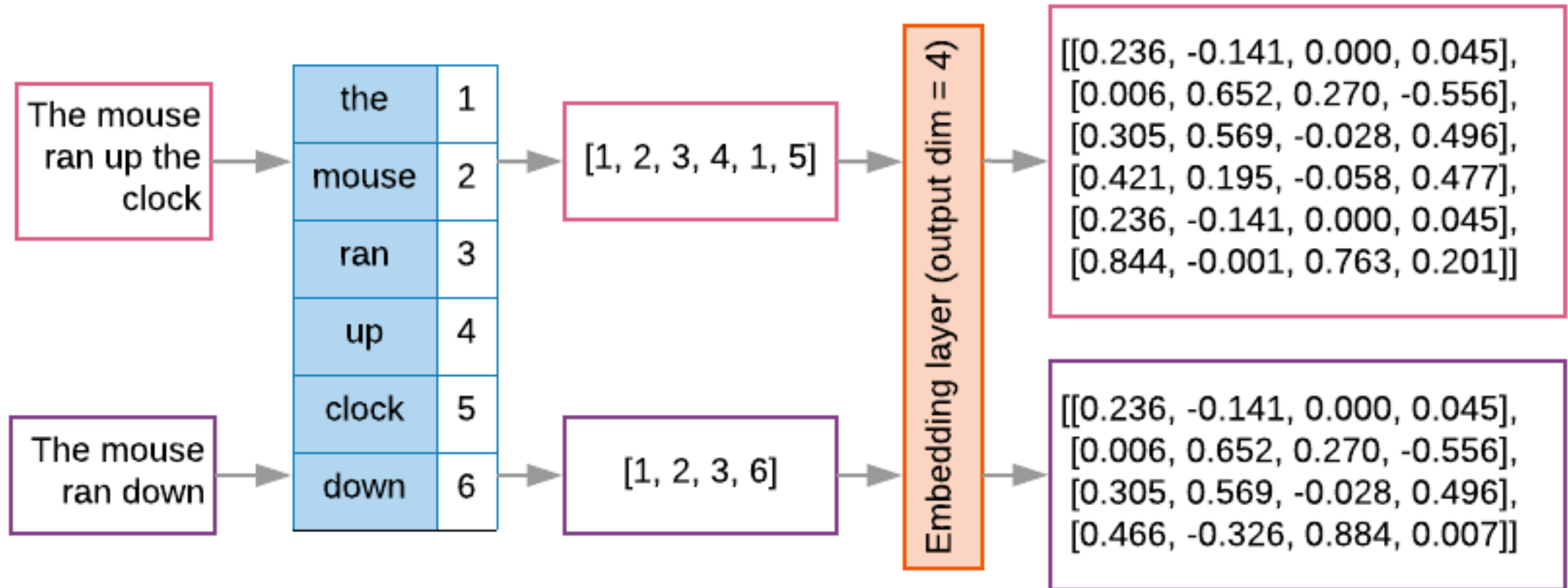


Verb Tense

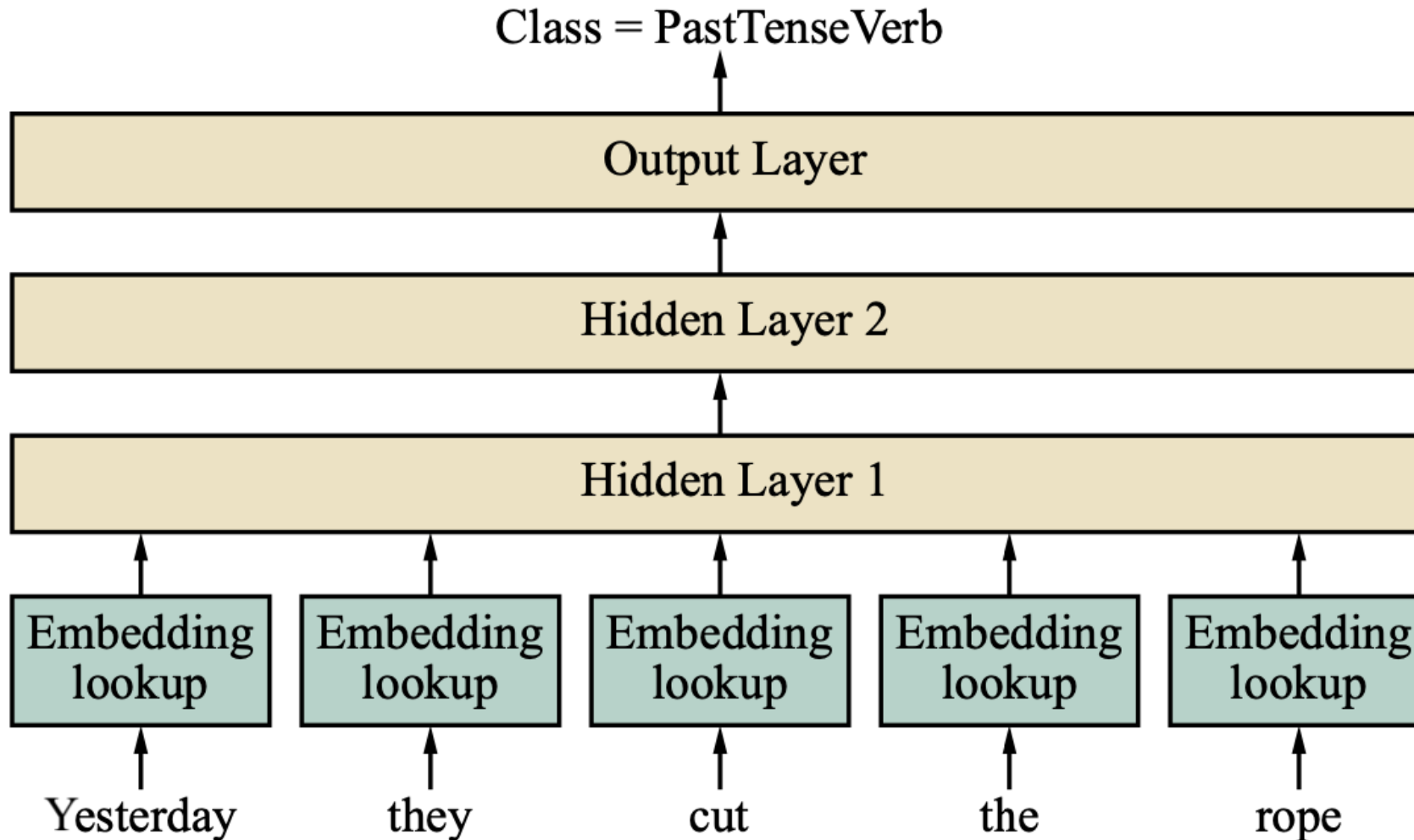


Country-Capital

Word embeddings



Feedforward part-of-speech (POS) tagging model

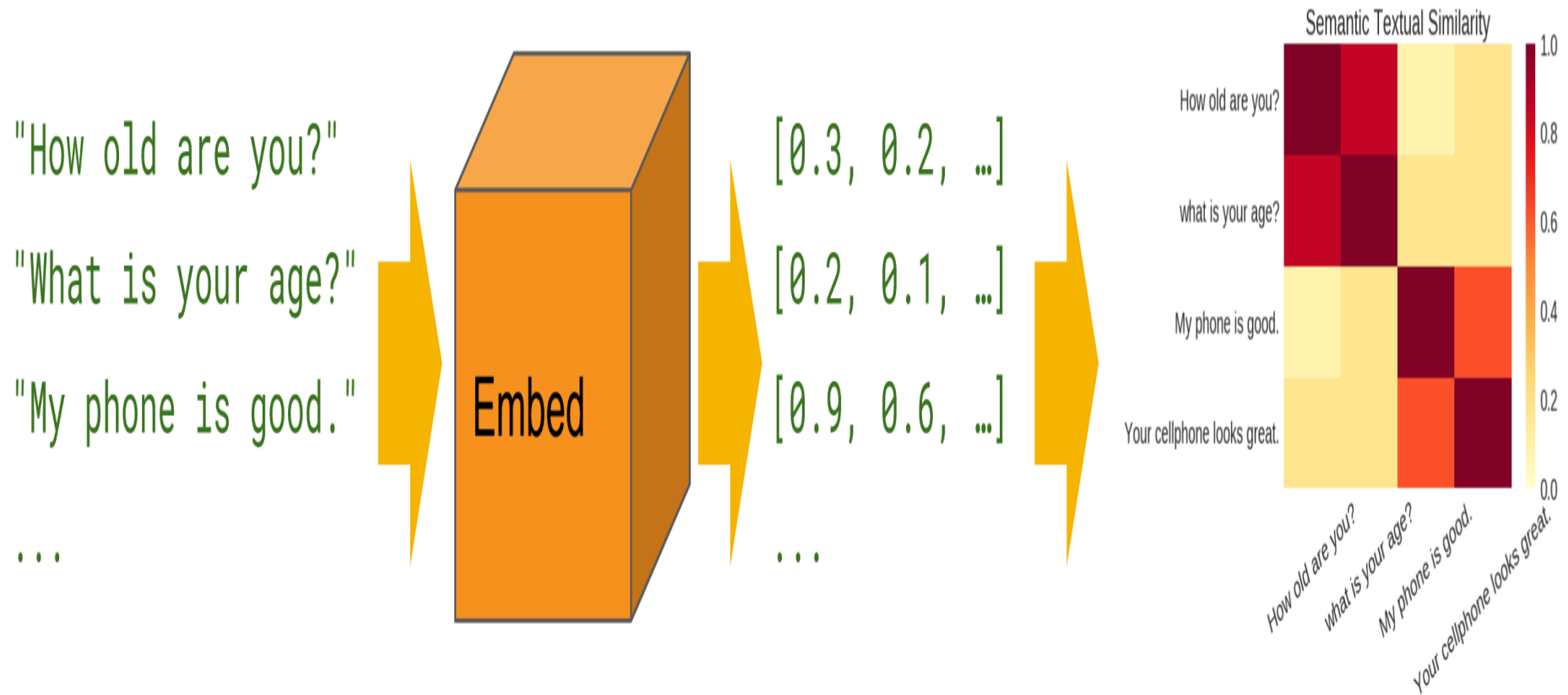


Universal Sentence Encoder (USE)

- 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.

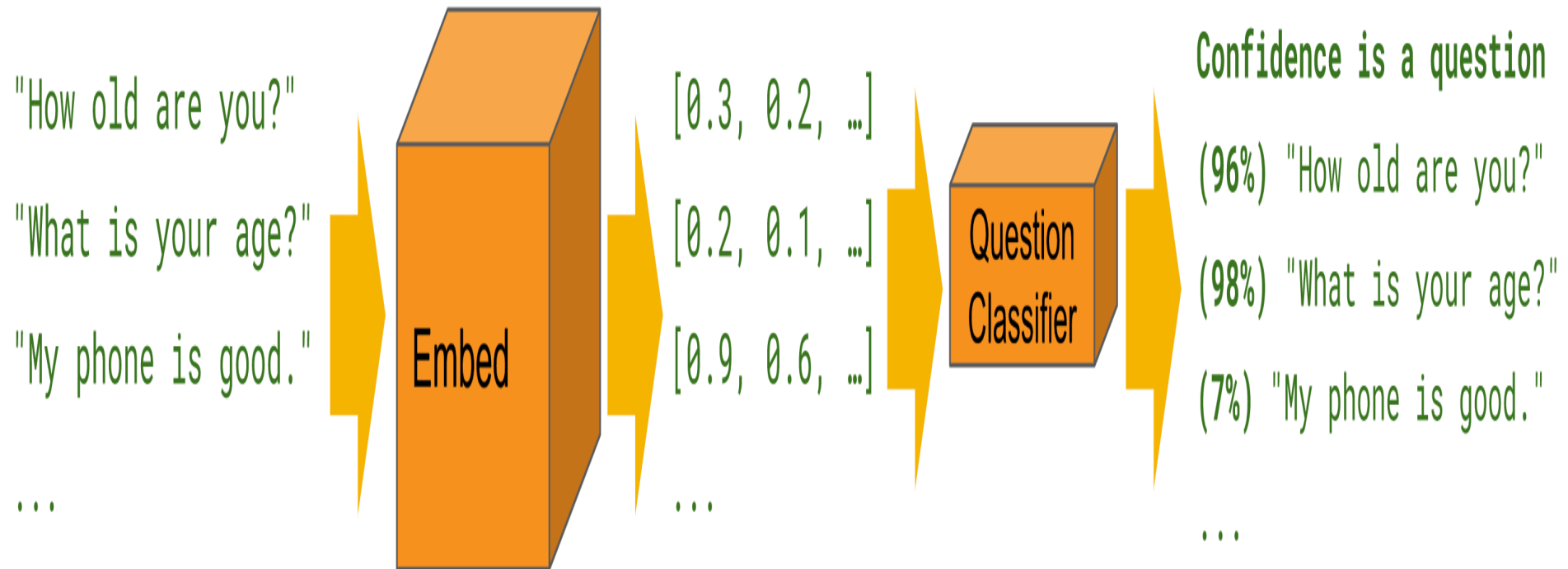
Universal Sentence Encoder (USE)

Semantic Similarity



Universal Sentence Encoder (USE)

Classification



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."])
```


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"])
```


Python in Google Colab (Python101)

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

python101.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings A

RAM Disk Editing

Table of contents

- Text Clustering
- Semantic Analysis and Named Entity Recognition (NER)
 - Semantic Analysis
 - Named Entity Recognition (NER)
- Sentiment Analysis
 - Sentiment Analysis - Unsupervised Lexical
 - Sentiment Analysis - Supervised Machine Learning
 - Sentiment Analysis - Supervised Deep Learning Models
 - Sentiment Analysis - Advanced Deep Learning
- Deep Learning and Universal Sentence-Embedding Models
 - Universal Sentence Encoder (USE)**
 - Universal Sentence Encoder Multilingual (USEM)
- Data Visualization
- Section

Deep Learning and Universal Sentence-Embedding Models

Universal Sentence Encoder (USE)

- Source: Universal Sentence Encoder: <https://tfhub.dev/google/universal-sentence-encoder/4>

```
[ ] 1 import tensorflow as tf
    2 import tensorflow_hub as hub
    3 import numpy as np
    4 import pandas as pd
    5 import os
    6 import re
    7 import matplotlib.pyplot as plt
    8 import seaborn as sns
    9
   10 module_url = "https://tfhub.dev/google/universal-sentence-encoder/4"
   11 # "https://tfhub.dev/google/universal-sentence-encoder-large/5"
   12 model = hub.load(module_url)
   13 print ("module %s loaded" % module_url)
   14 def embed(input):
   15     return model(input)
```

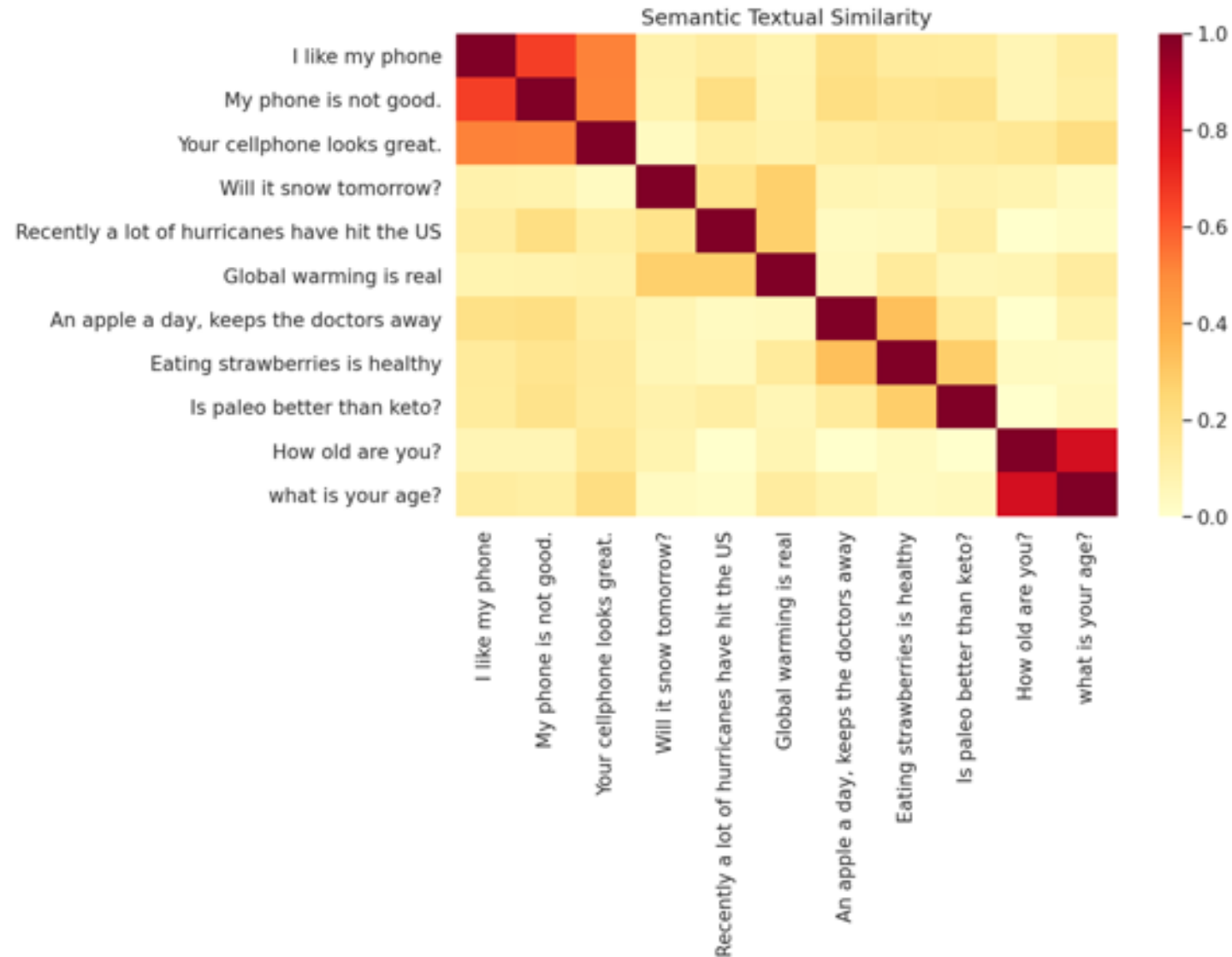
module <https://tfhub.dev/google/universal-sentence-encoder/4> loaded

```
[ ] 1 word = "Elephant"
    2 sentence = "I am a sentence for which I would like to get its embedding."
```

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

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

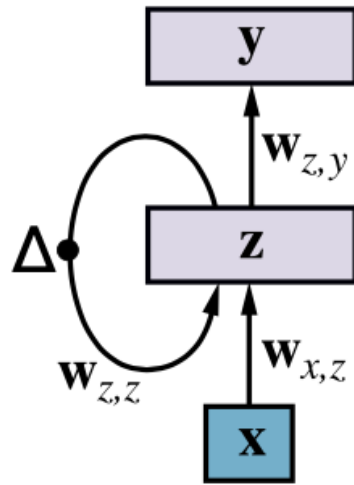


<https://tinyurl.com/aintpupython101>

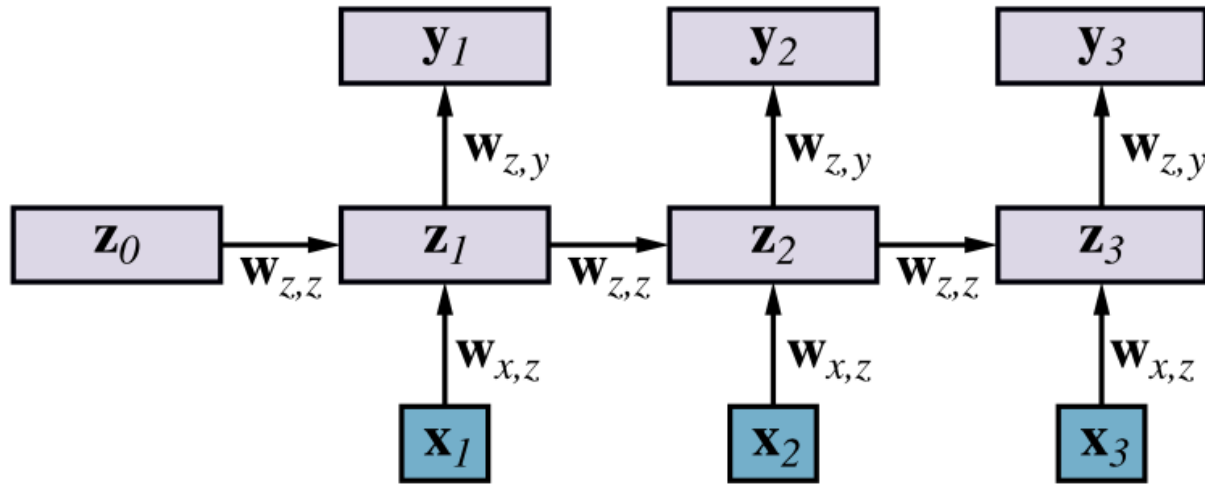
Outline

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

RNN

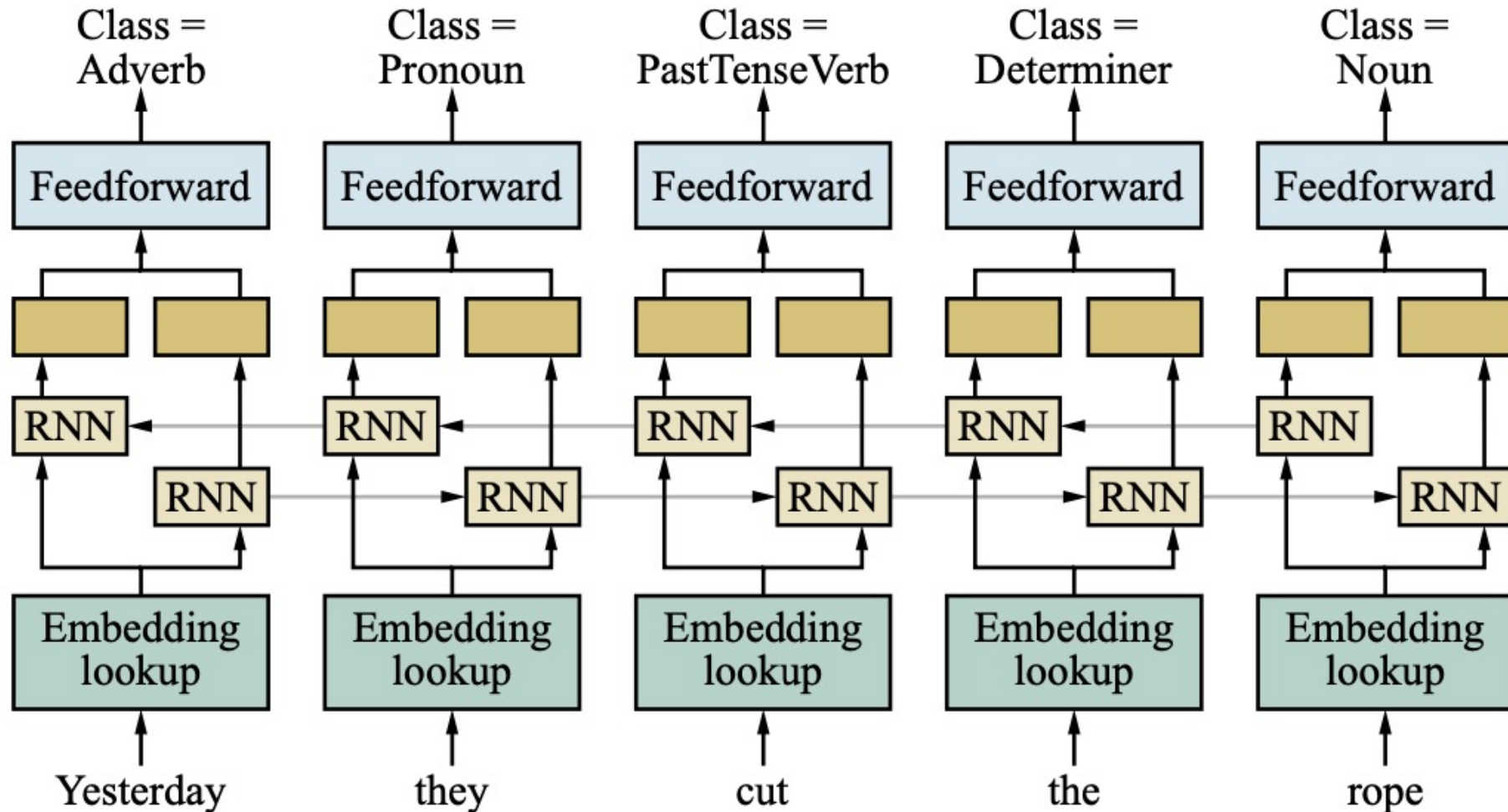


(a)



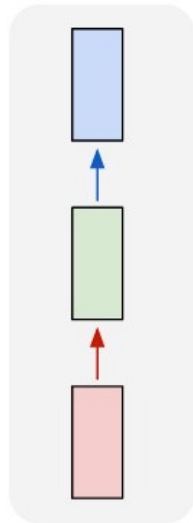
(b)

Bidirectional RNN network for POS tagging



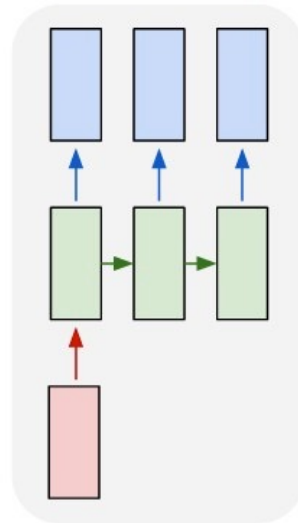
LSTM Recurrent Neural Network

one to one



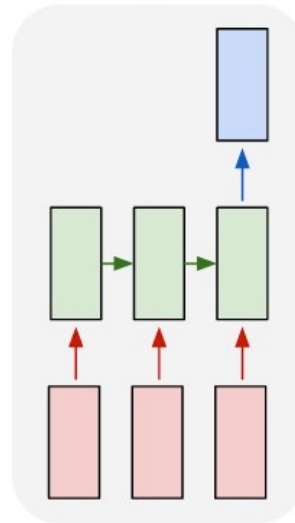
**Traditional
Neural
Network**

one to many



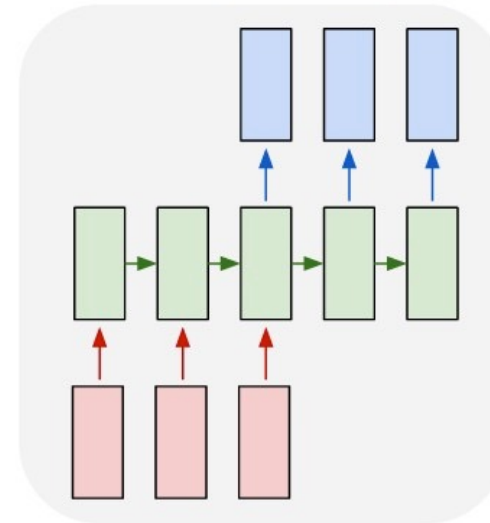
**Music
Generation**

many to one



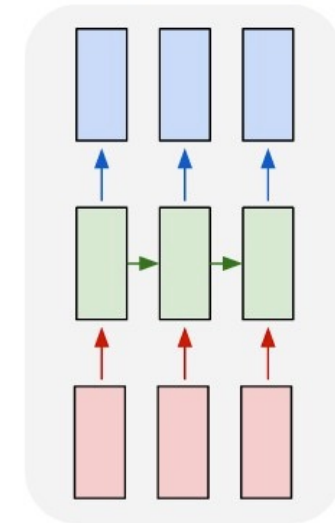
**Sentiment
Classification**

many to many



**Name
Entity
Recognition**

many to many

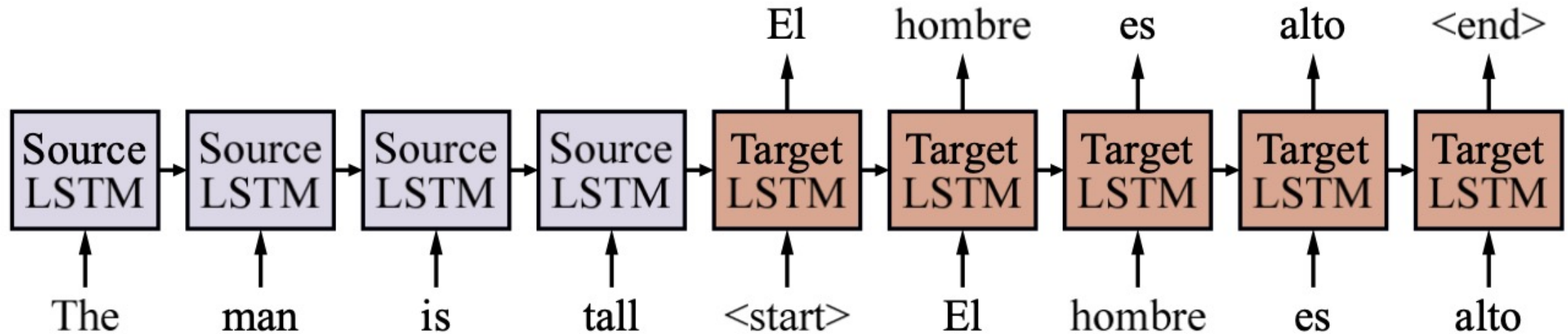


**Machine
Translation**

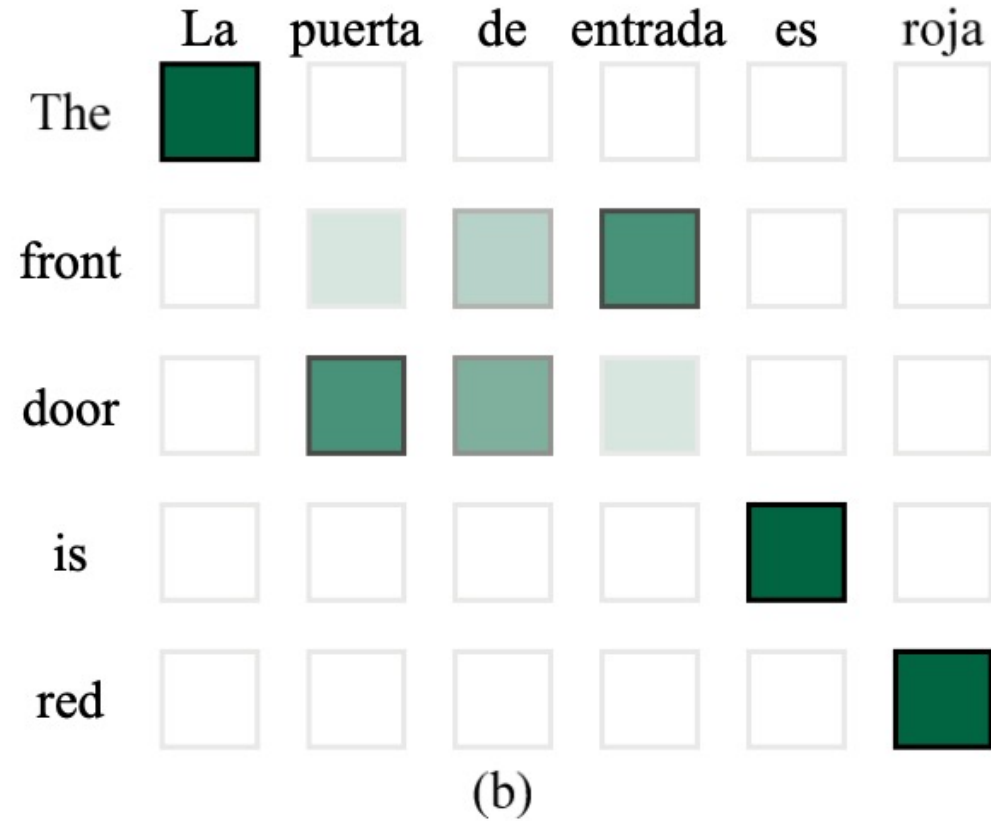
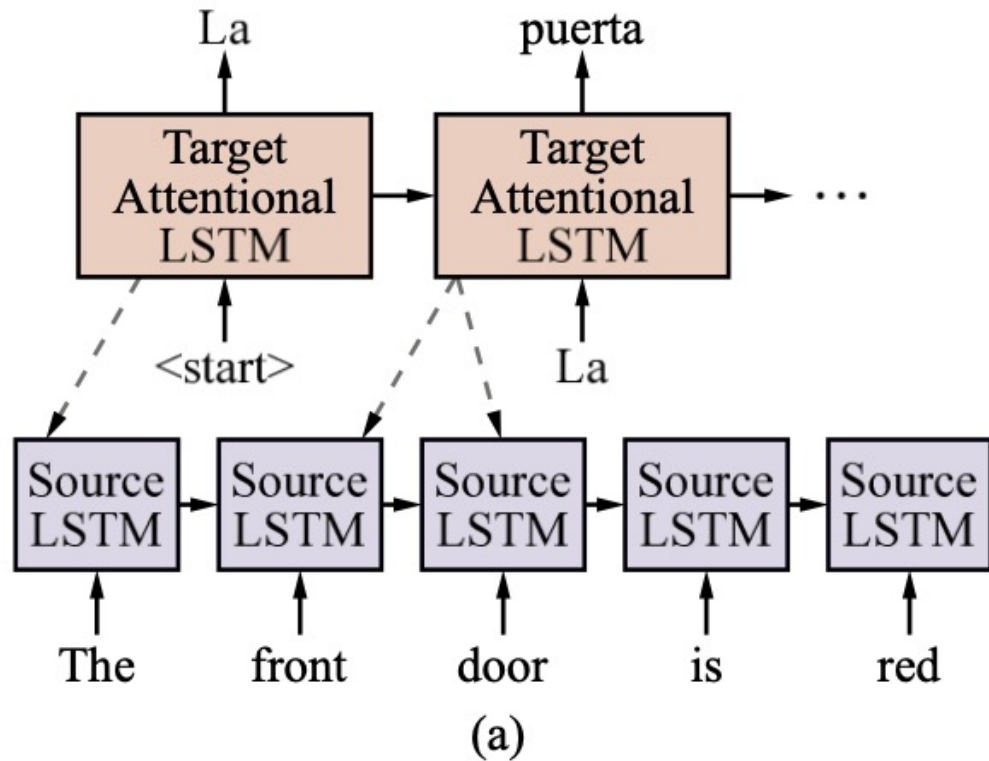
Outline

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

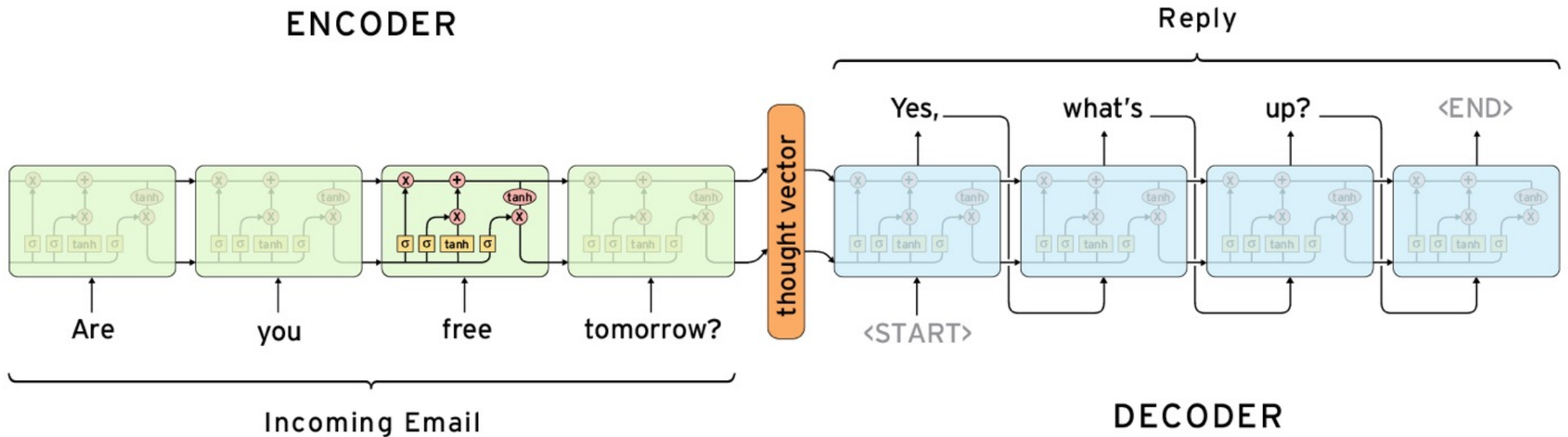
Sequence-to-Sequence model



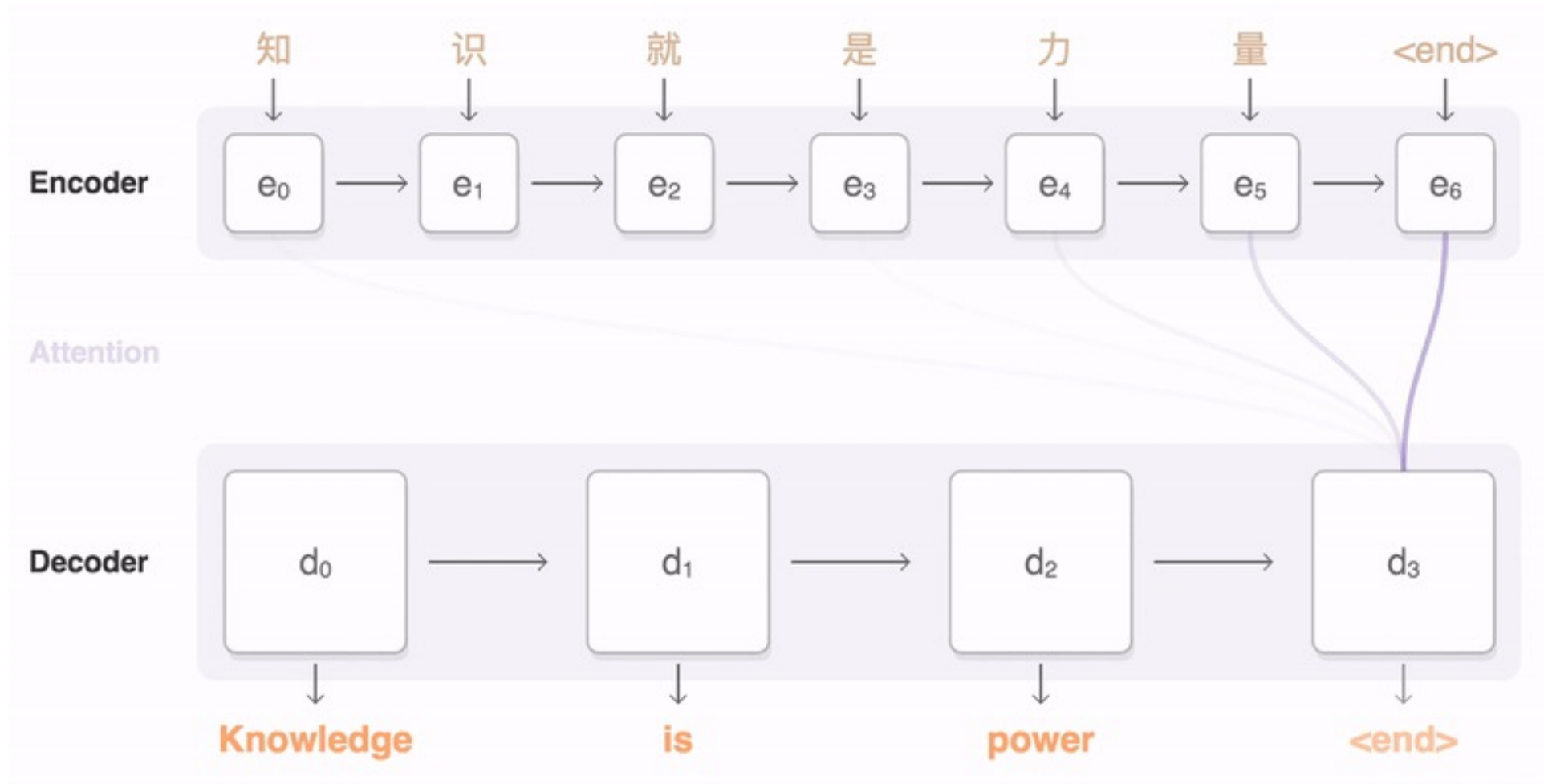
Attentional Sequence-to-Sequence model for English-to-Spanish translation



The Sequence to Sequence model (seq2seq)



Sequence to Sequence (Seq2Seq)

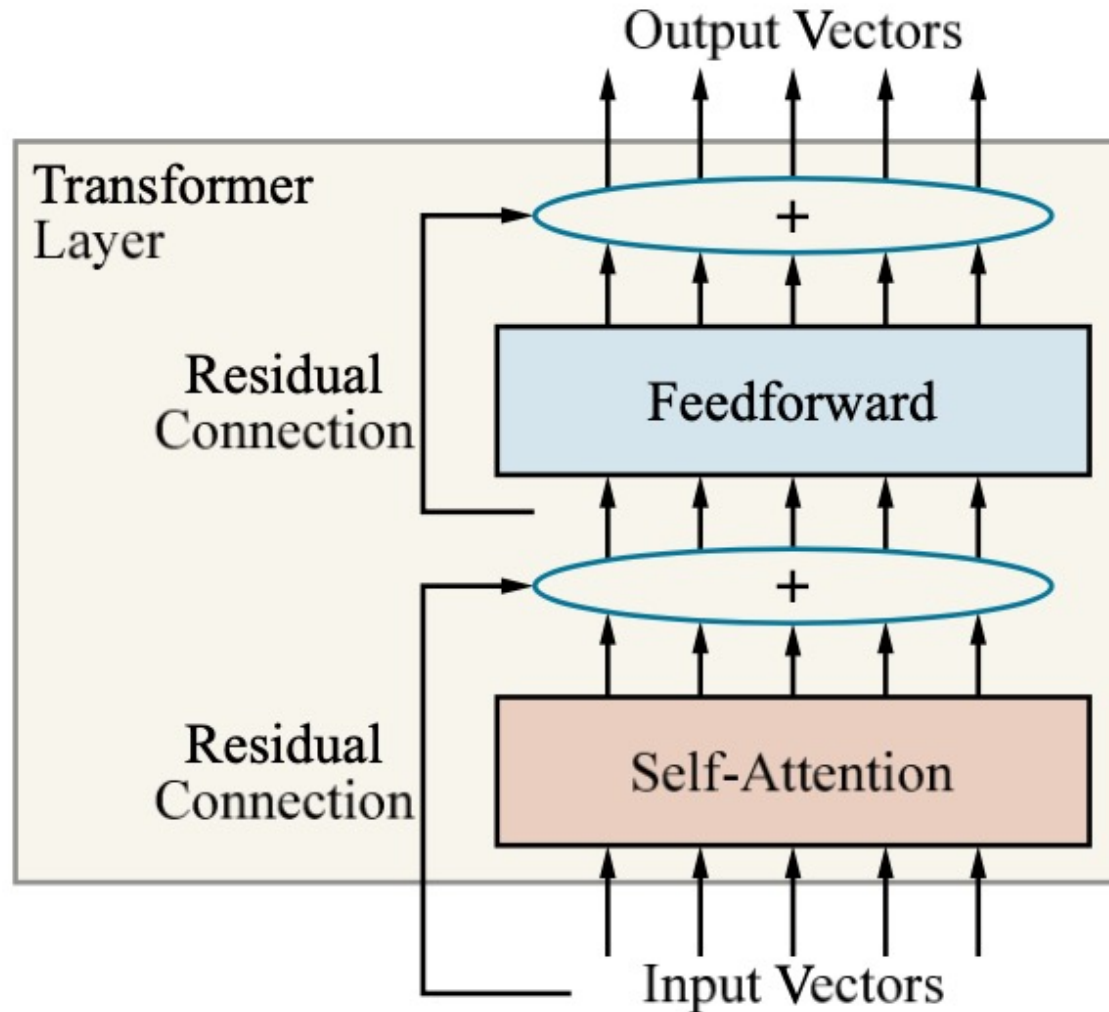


Outline

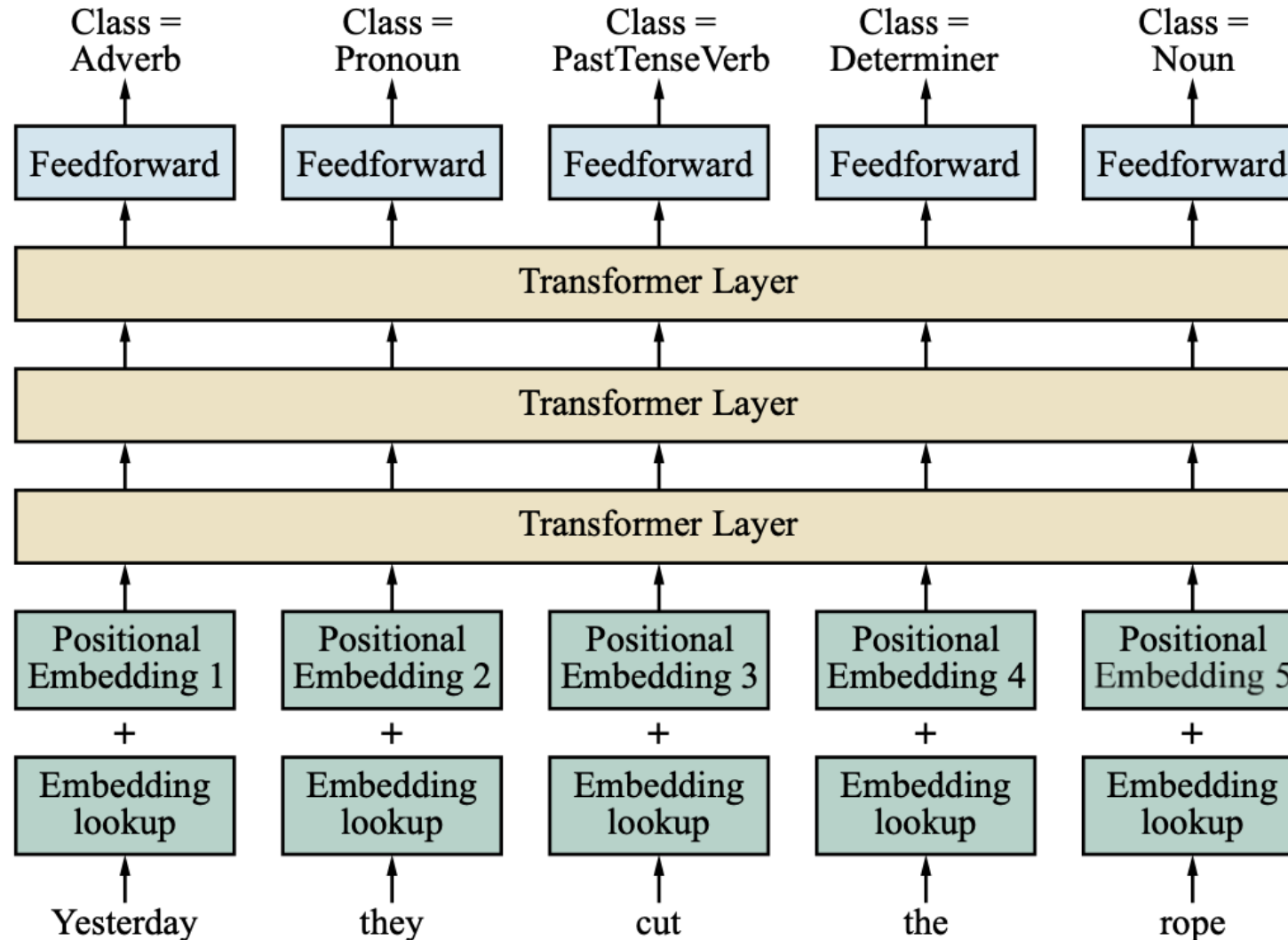
- Word Embeddings
- Recurrent Neural Networks for NLP
- Sequence-to-Sequence Models
- **The Transformer Architecture**
- Pretraining and Transfer Learning
- State of the art (SOTA)

Single-layer Transformer

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

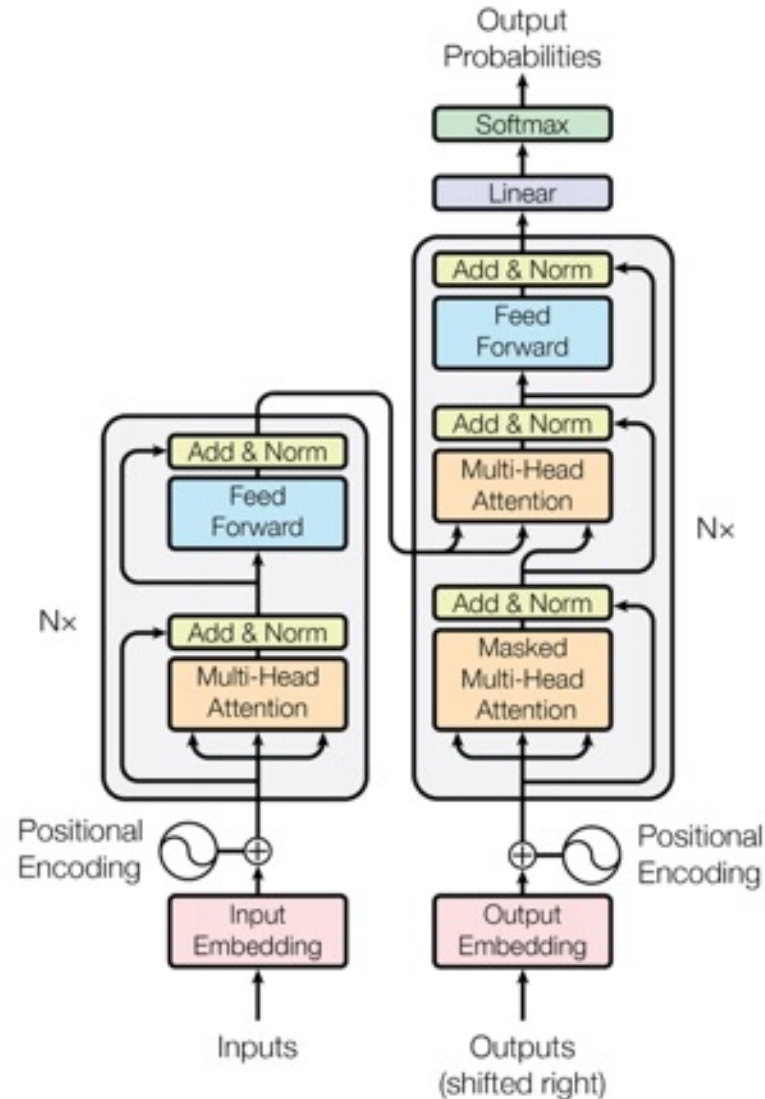


Transformer Architecture for POS Tagging



Transformer (Attention is All You Need)

(Vaswani et al., 2017)



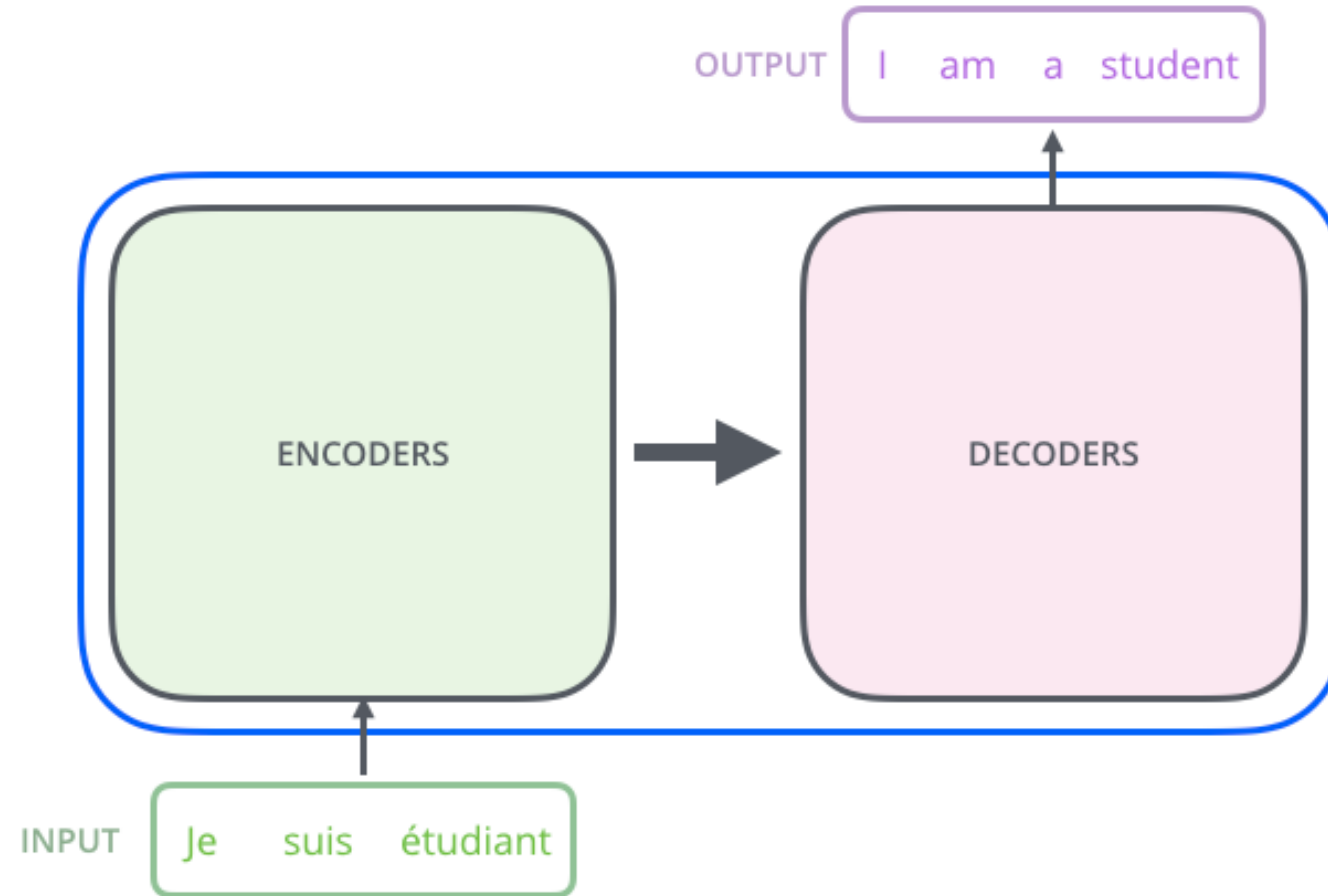
Source: Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In *Advances in neural information processing systems*, pp. 5998-6008. 2017.

Transformer



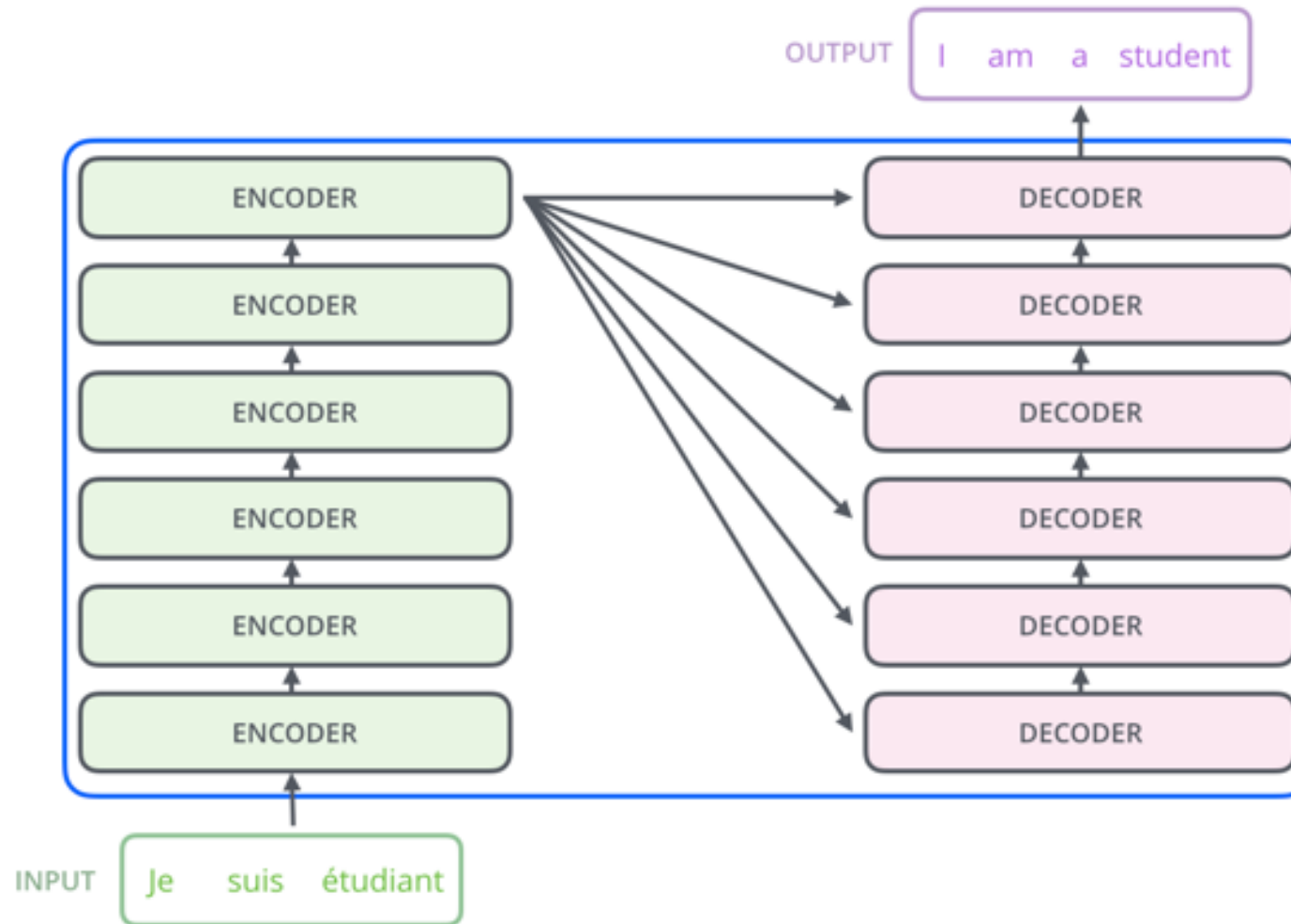
Transformer

Encoder Decoder



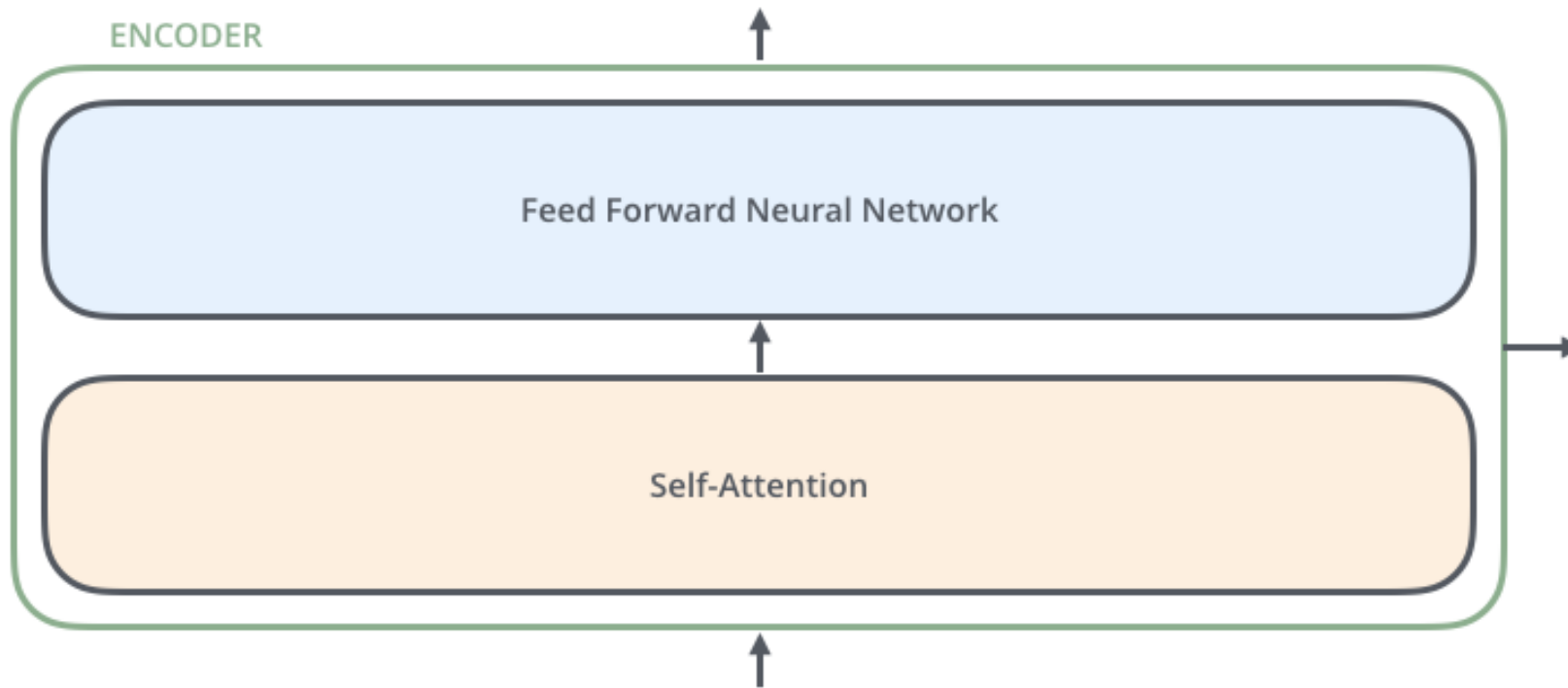
Transformer

Encoder Decoder Stack

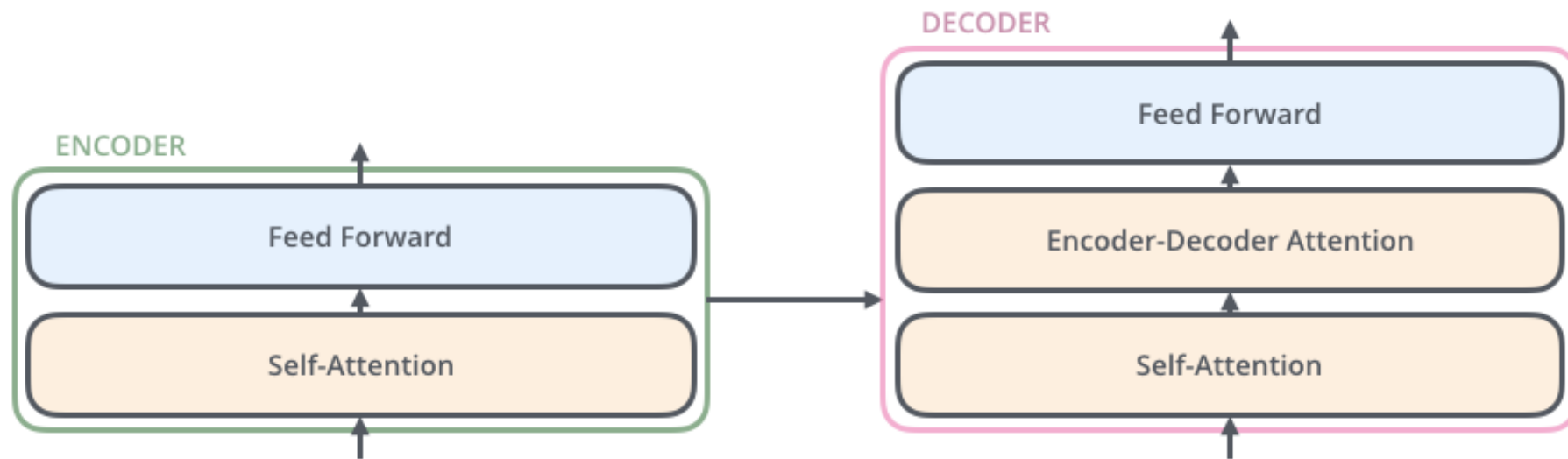


Transformer

Encoder Self-Attention



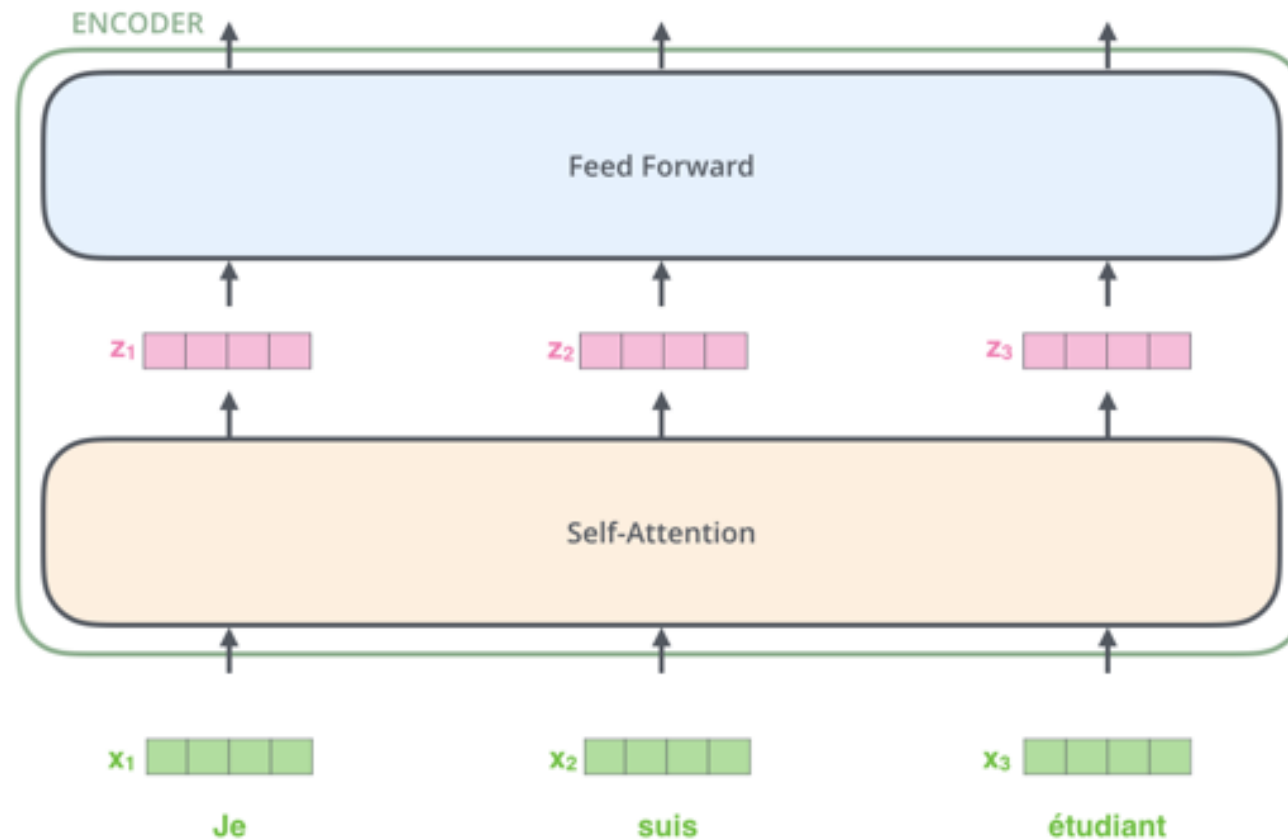
Transformer Decoder



Transformer

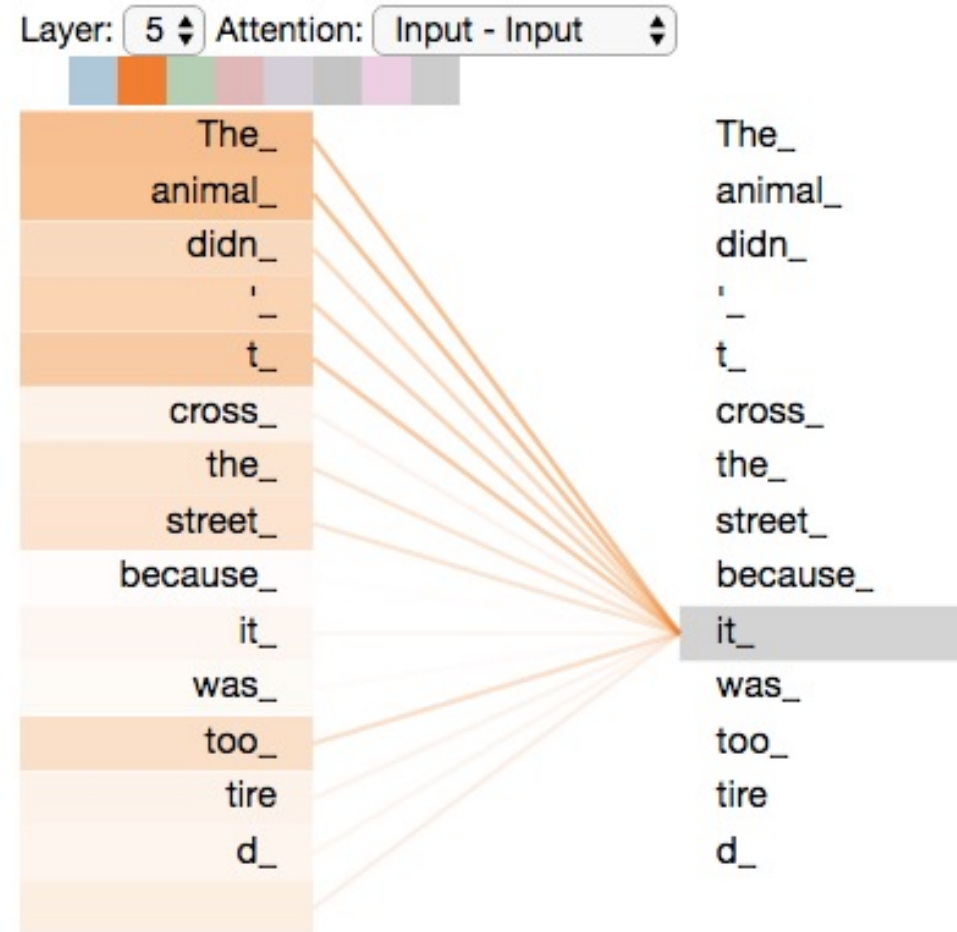
Encoder with Tensors

Word Embeddings



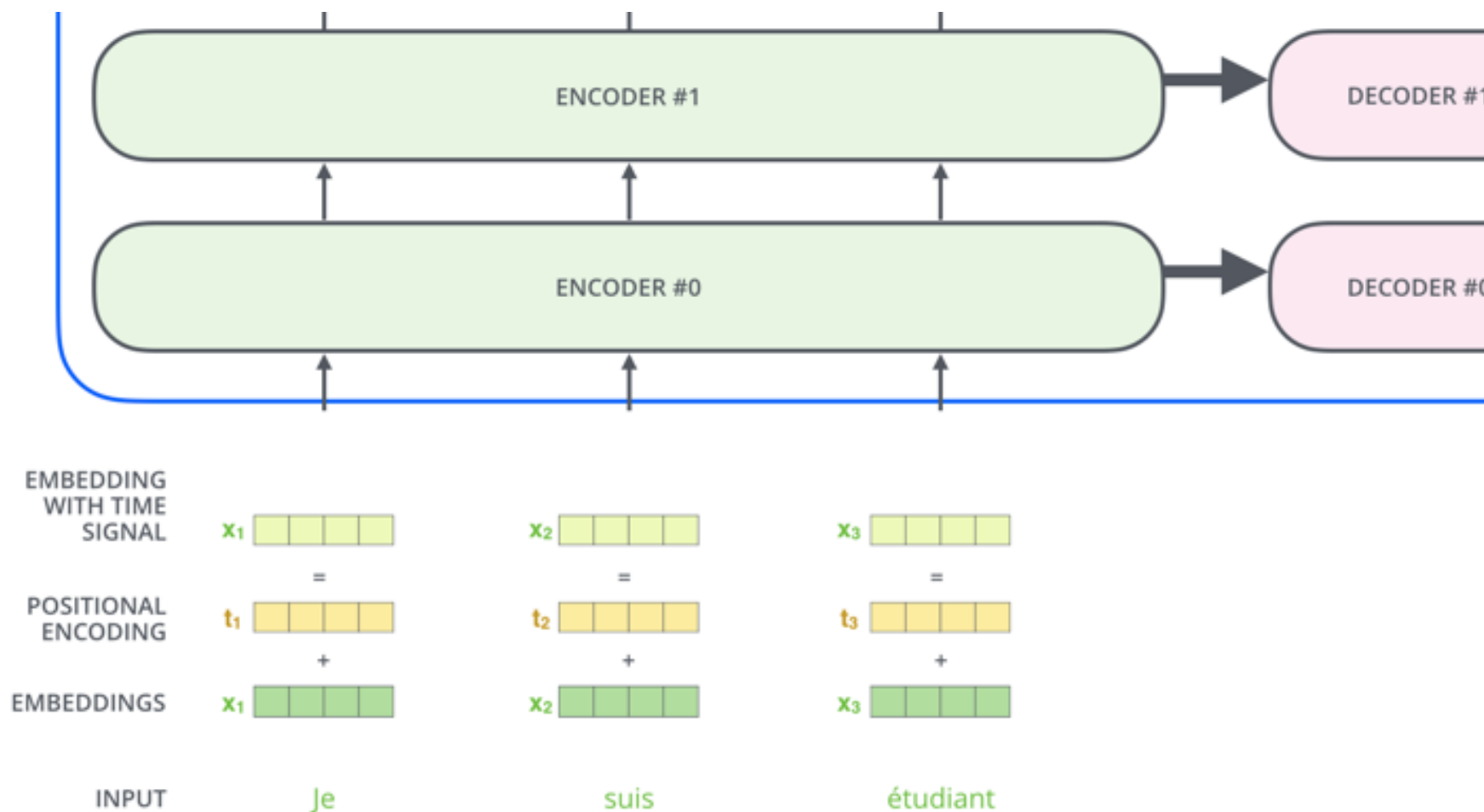
Transformer

Self-Attention Visualization



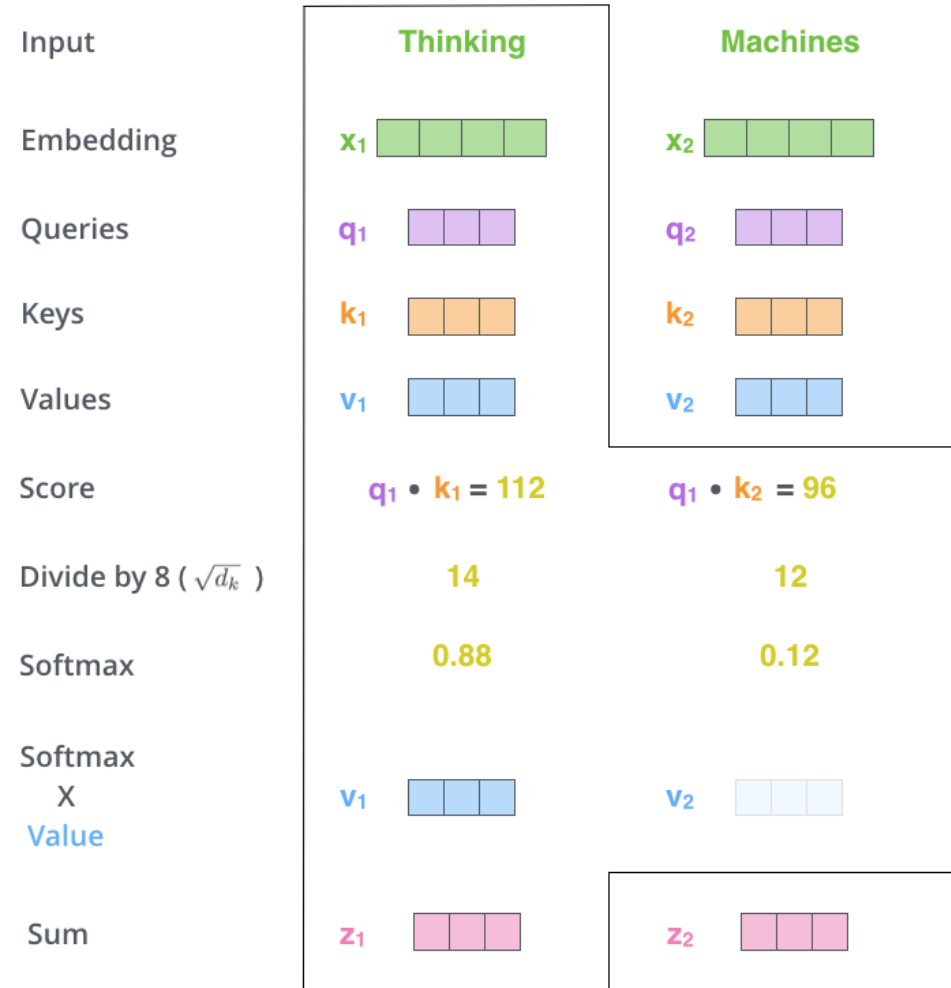
Transformer

Positional Encoding Vectors



Transformer

Self-Attention Softmax Output



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

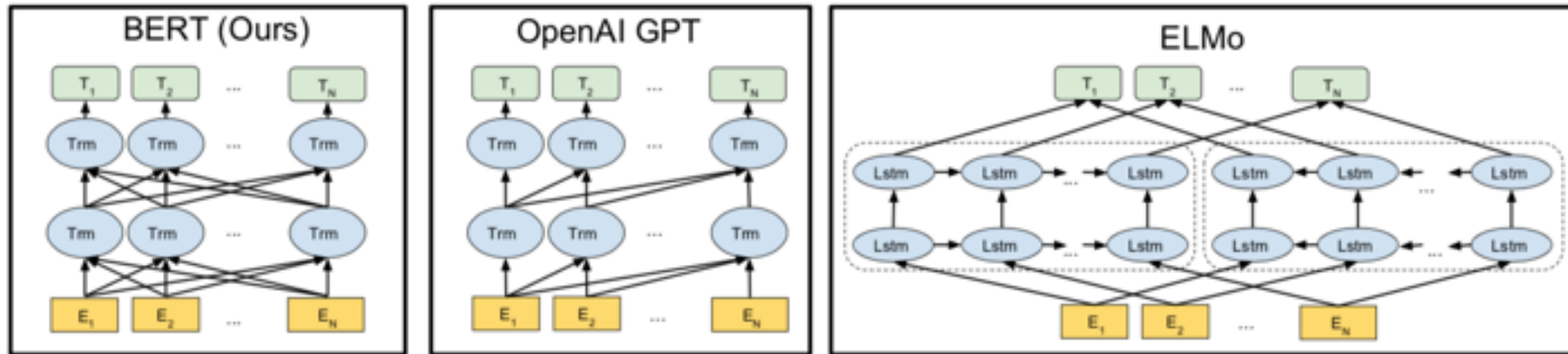
**BERT: Pre-training of Deep Bidirectional Transformers for
Language Understanding**

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova
Google AI Language

`{jacobdevlin, mingweichang, kentonl, kristout}@google.com`

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.

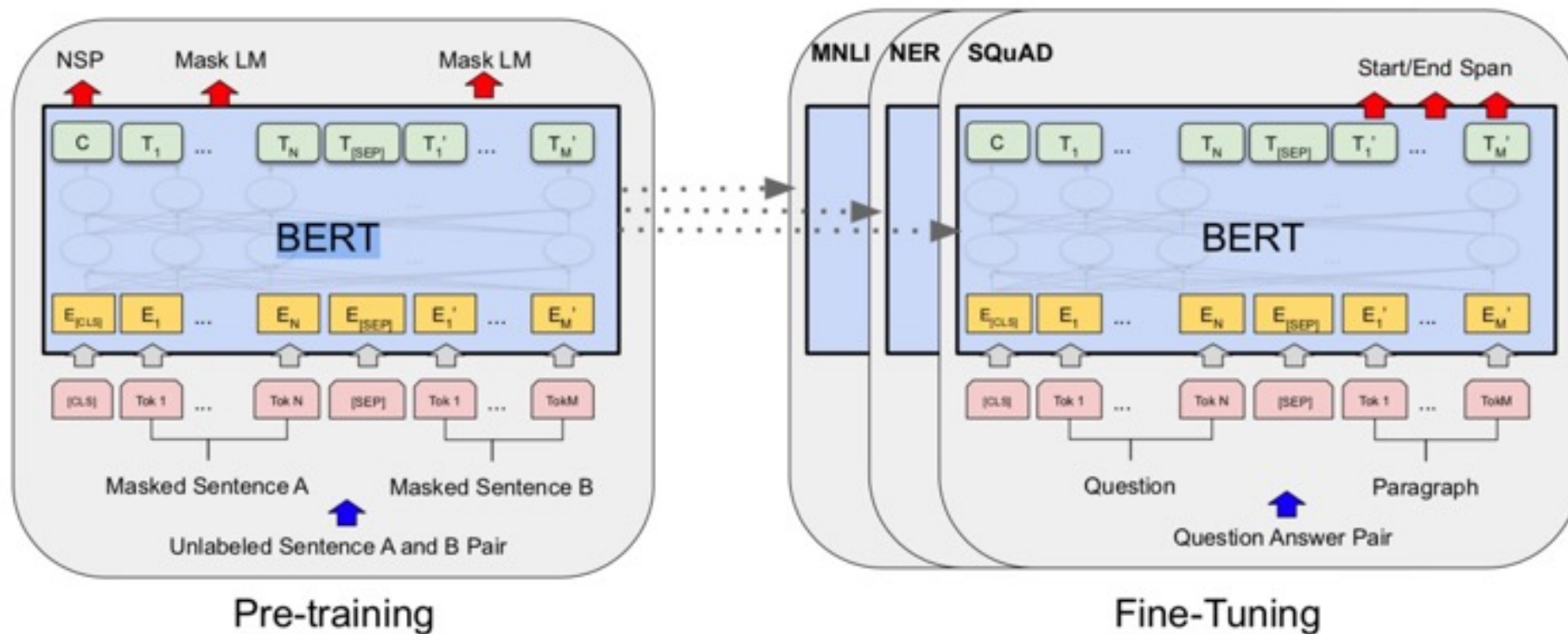
Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

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

BERT

(Bidirectional Encoder Representations from Transformers)

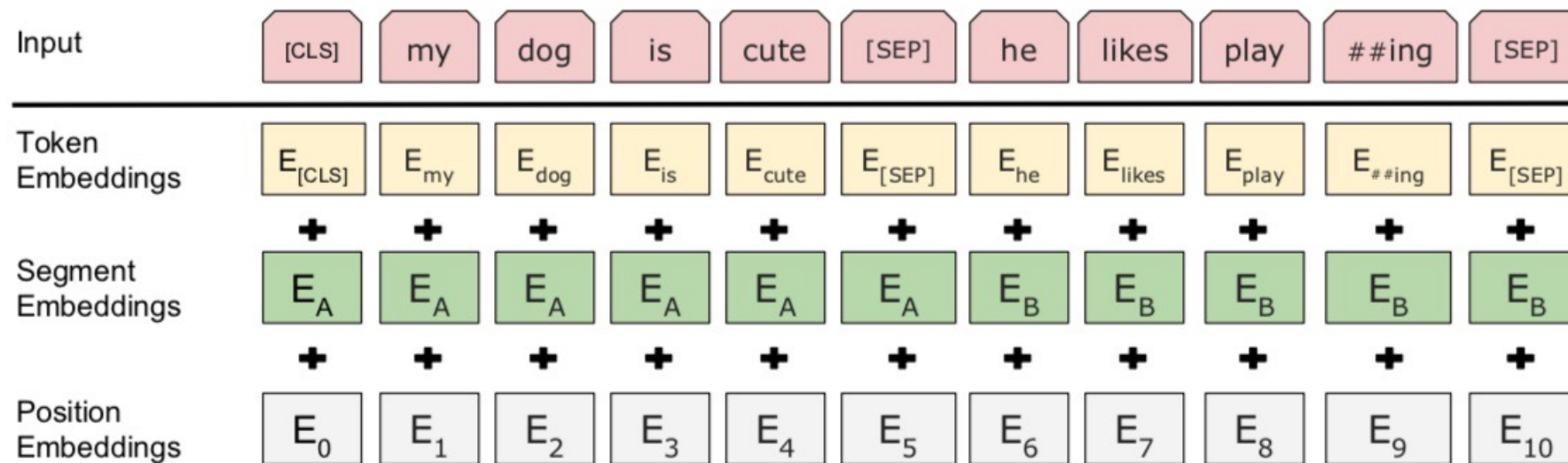
Overall pre-training and fine-tuning procedures for BERT



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

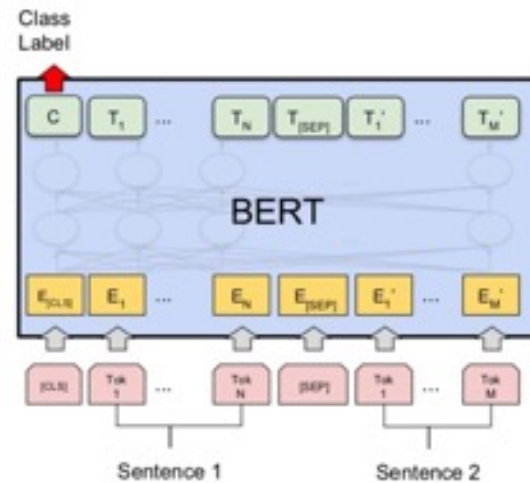
BERT (Bidirectional Encoder Representations from Transformers)

BERT input representation

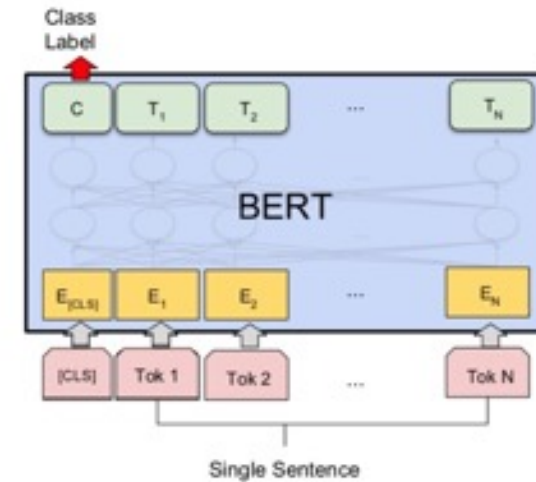


The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

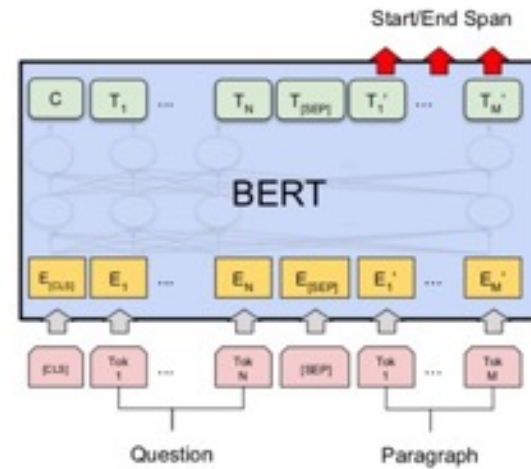
Fine-tuning BERT on Different Tasks



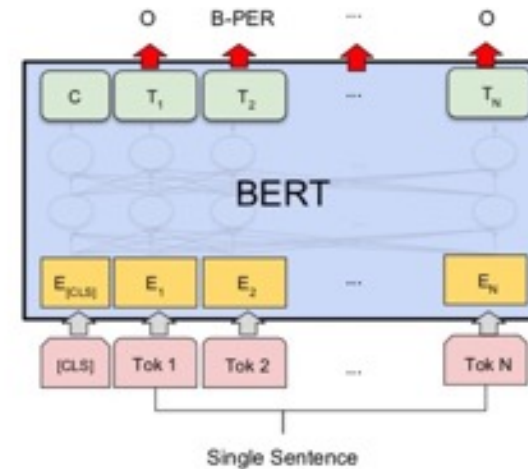
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



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



(c) Question Answering Tasks:
SQuAD v1.1

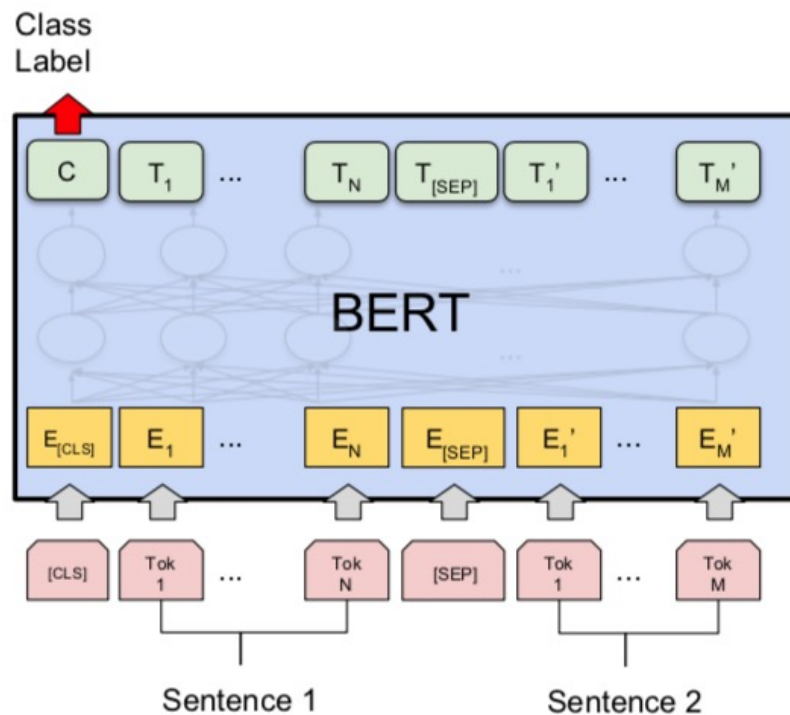


(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

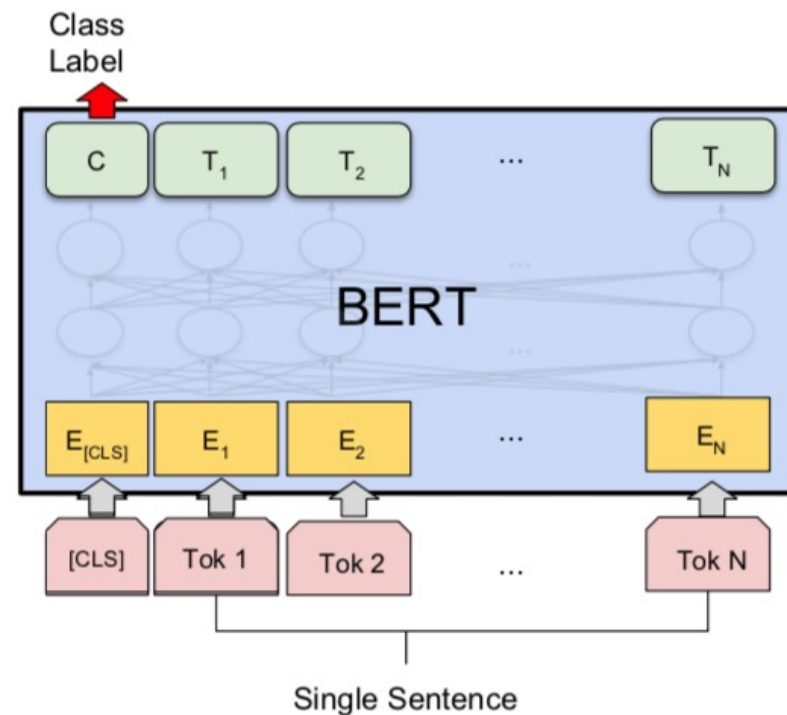
Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).

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

BERT Sequence-level tasks

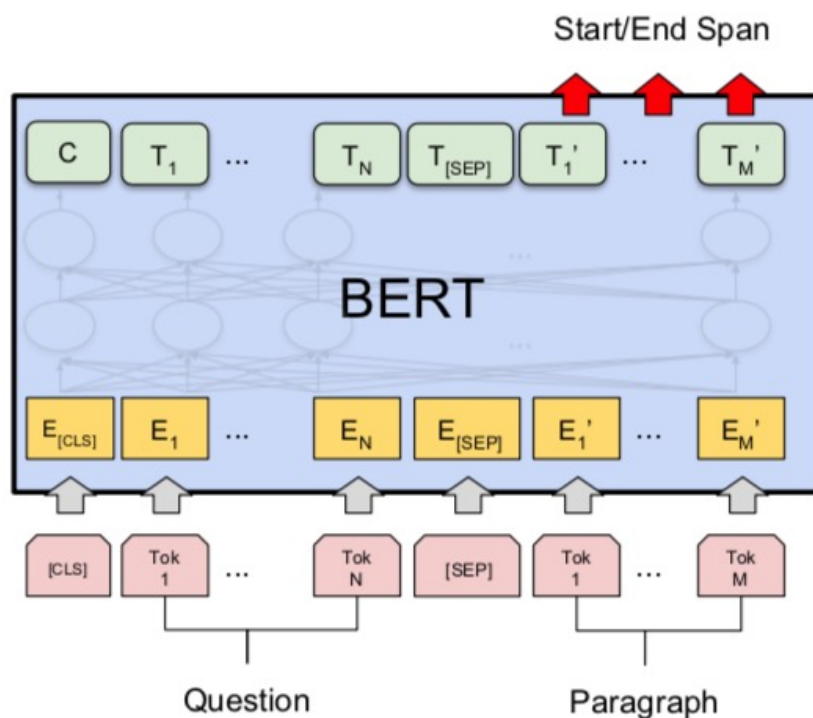


(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG

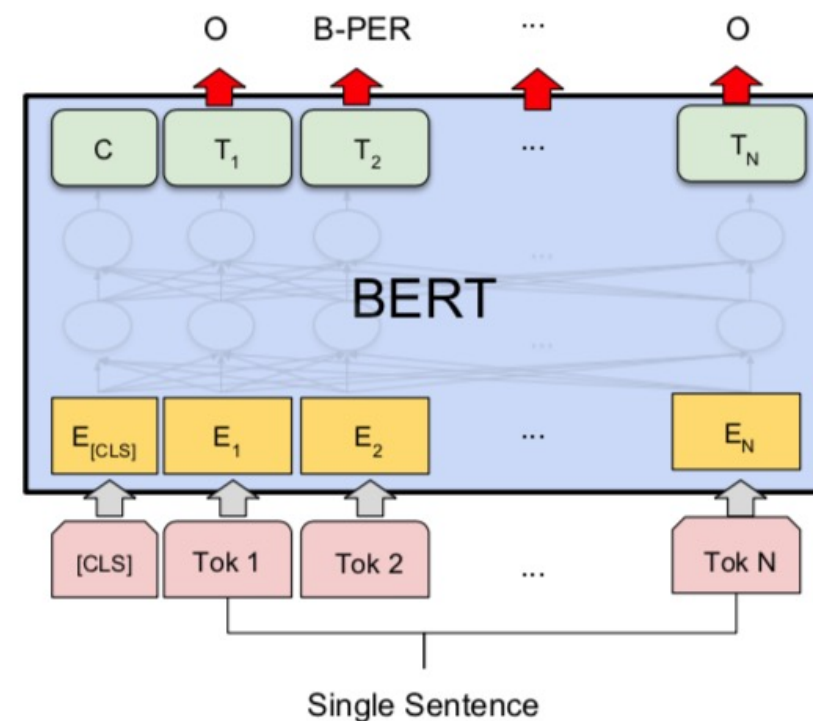


(b) Single Sentence Classification Tasks:
SST-2, CoLA

BERT Token-level tasks

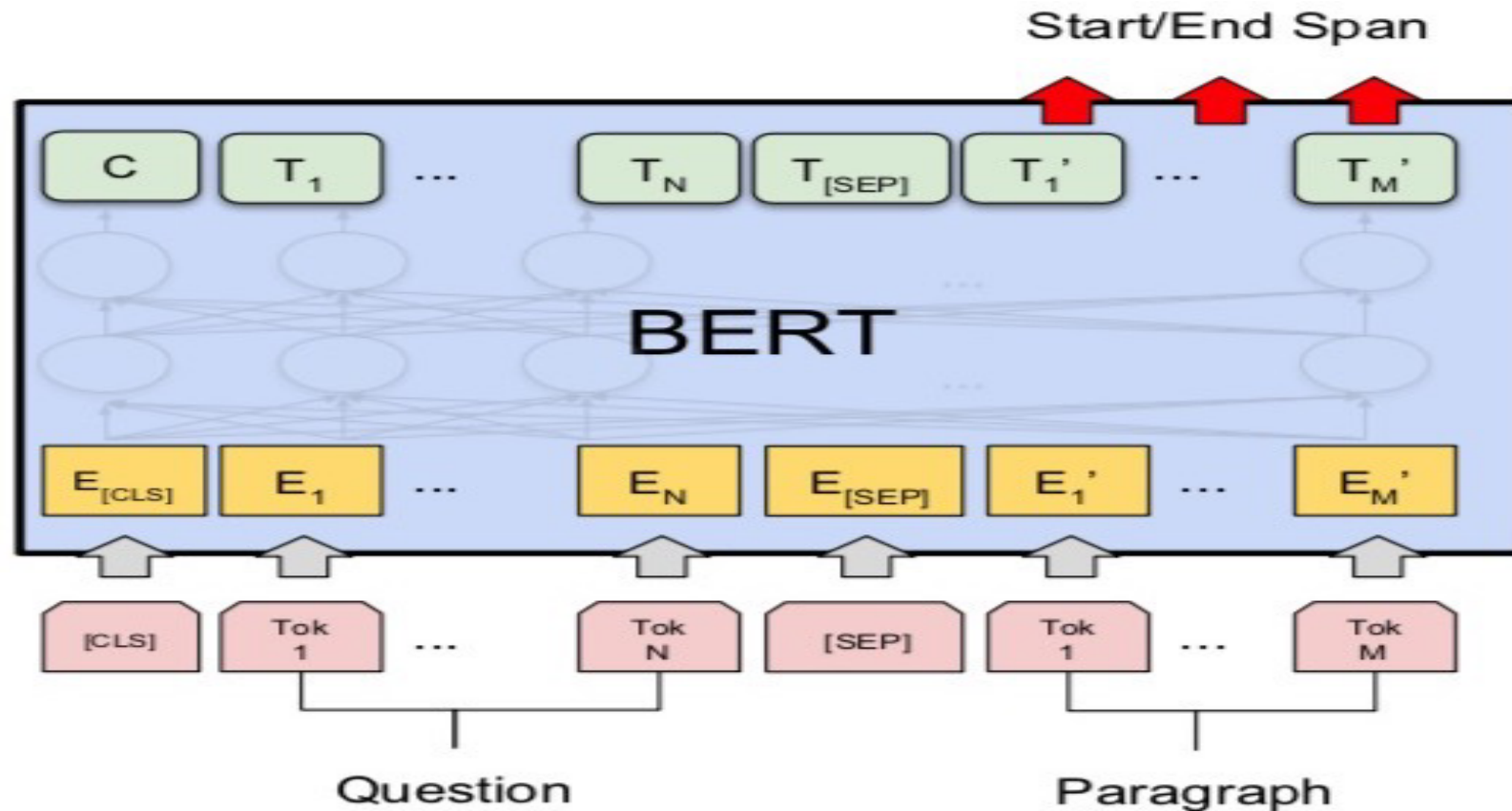


(c) Question Answering Tasks:
SQuAD v1.1



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

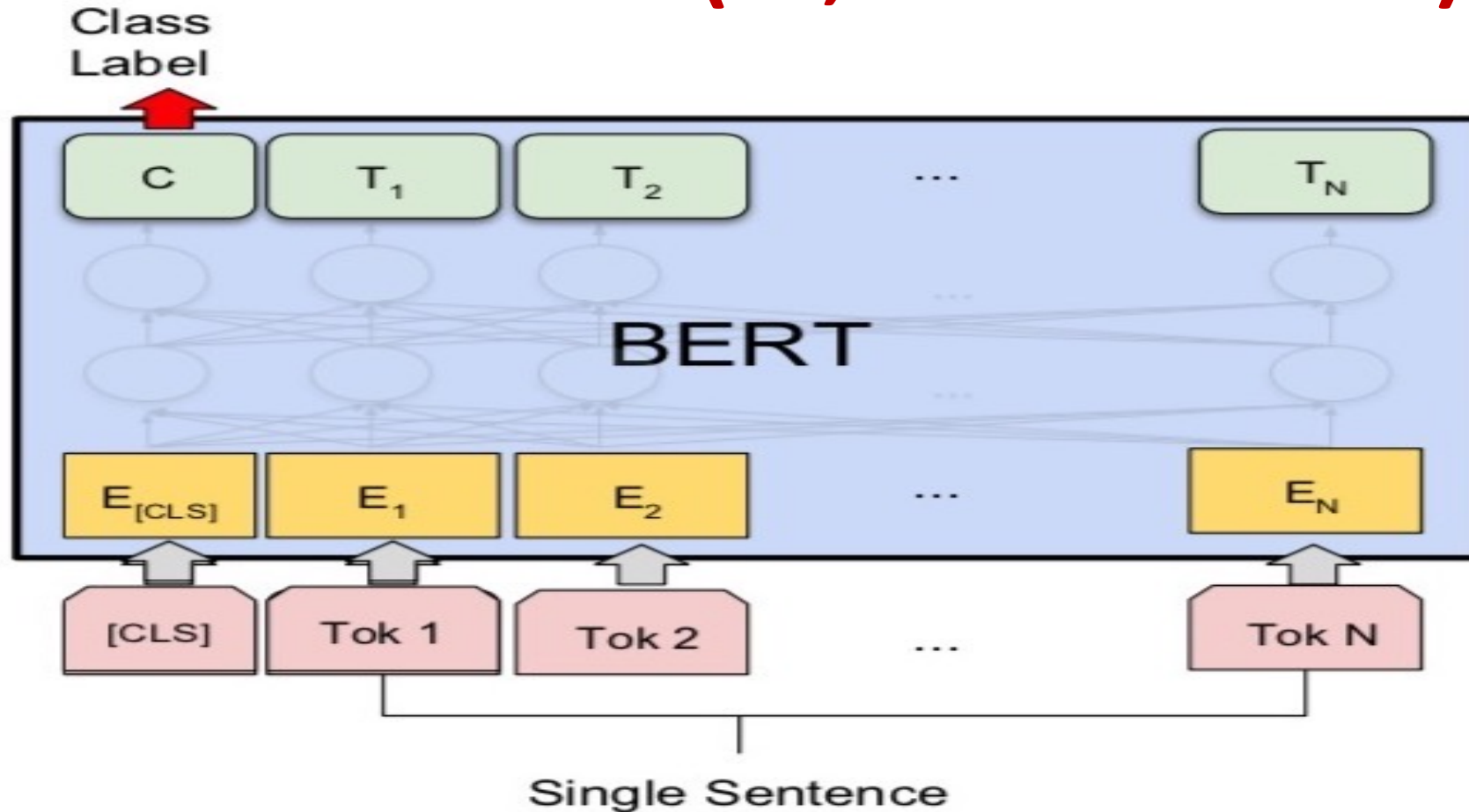
Fine-tuning BERT on Question Answering (QA)



(c) Question Answering Tasks:
SQuAD v1.1

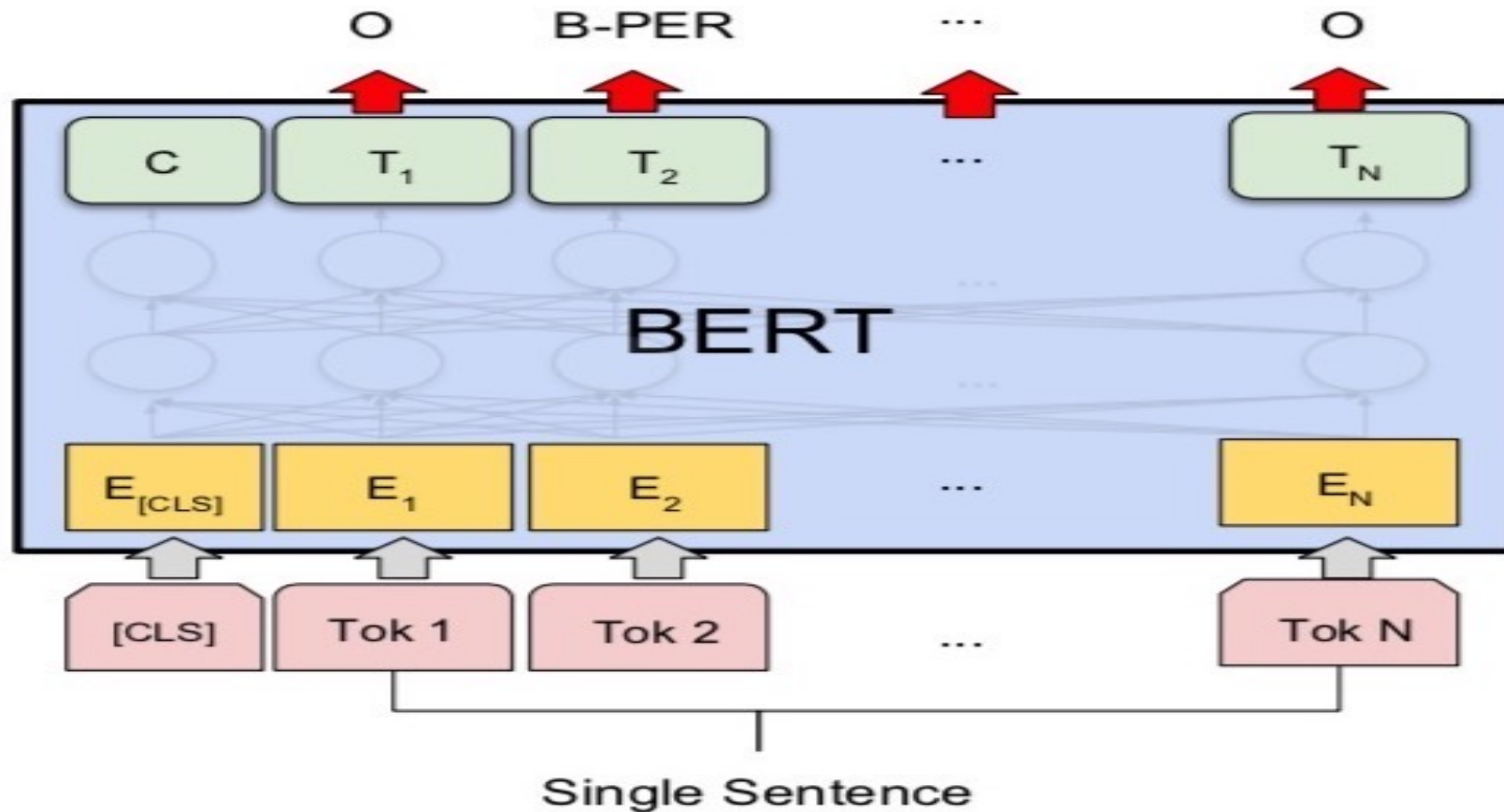
Fine-tuning BERT on Dialogue

Intent Detection (ID; Classification)



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

Fine-tuning BERT on Dialogue Slot Filling (SF)



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

General Language Understanding Evaluation (GLUE) benchmark

GLUE Test results

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

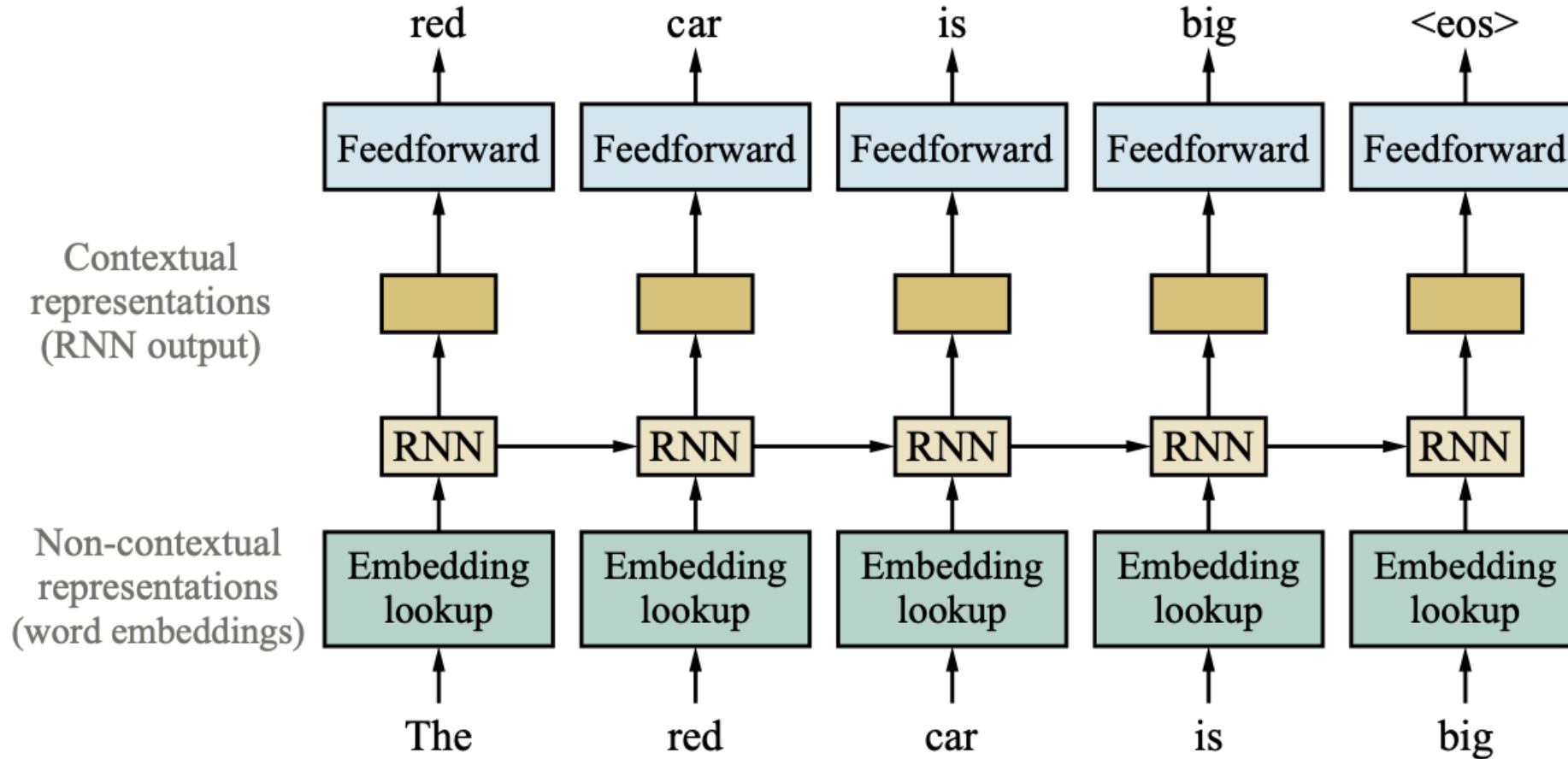
RTE: Recognizing Textual Entailment

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

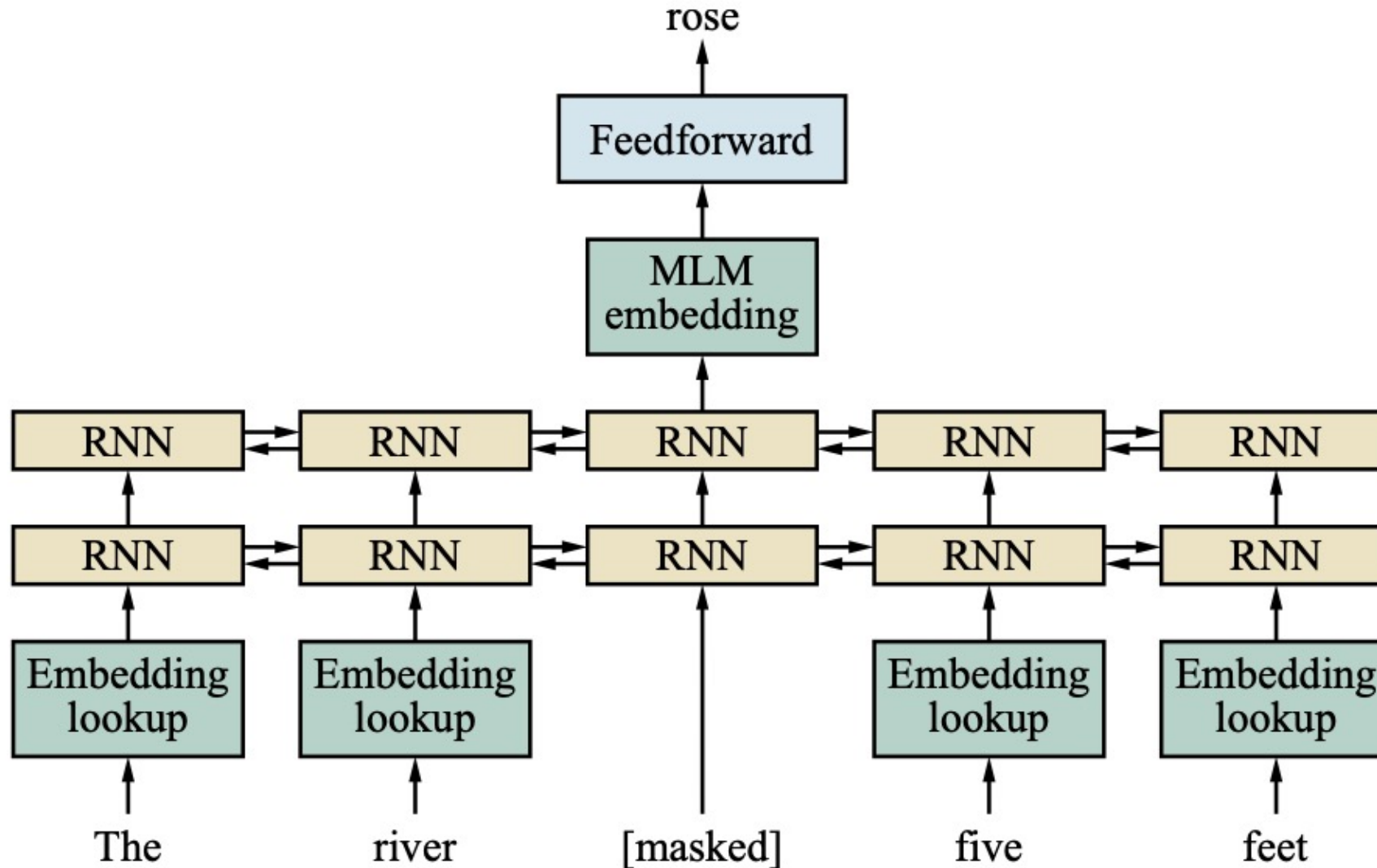
"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

Training Contextual Representations

using a left-to-right Language Model



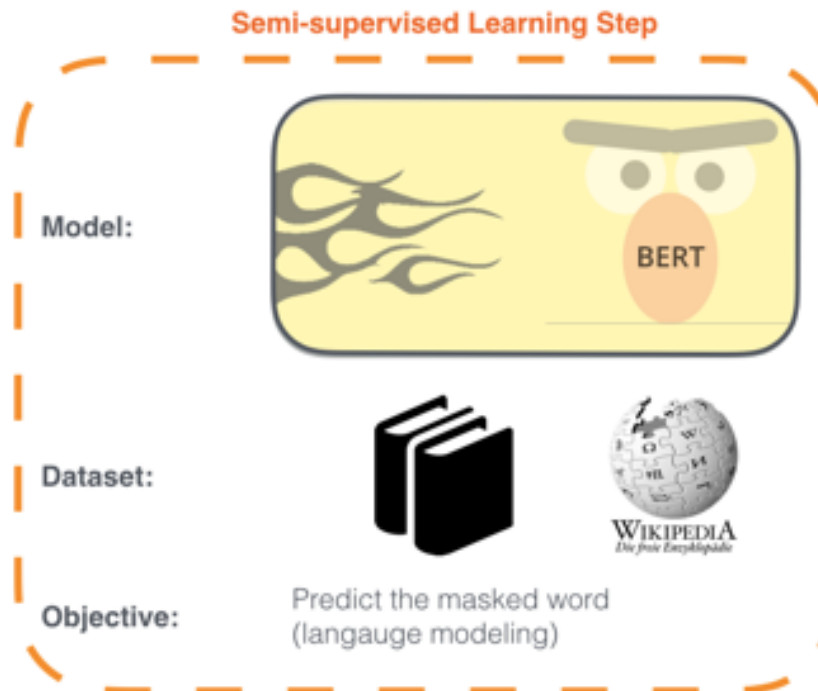
Masked Language Modeling: Pretrain a Bidirectional Model



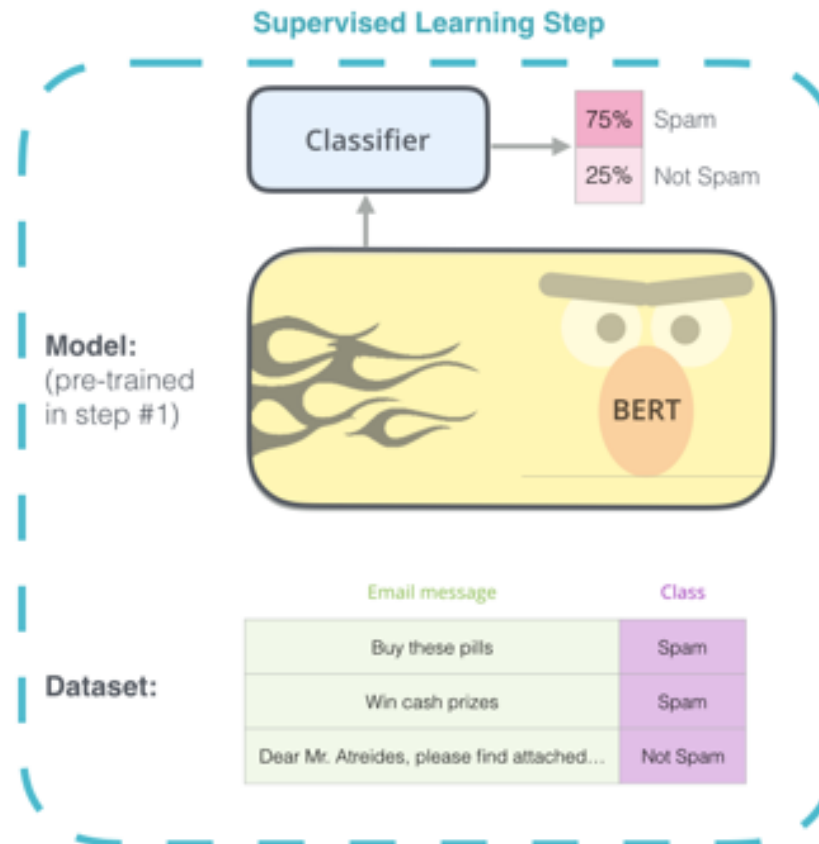
Illustrated BERT

1 - **Semi-supervised** training on large amounts of text (books, wikipedia..etc).

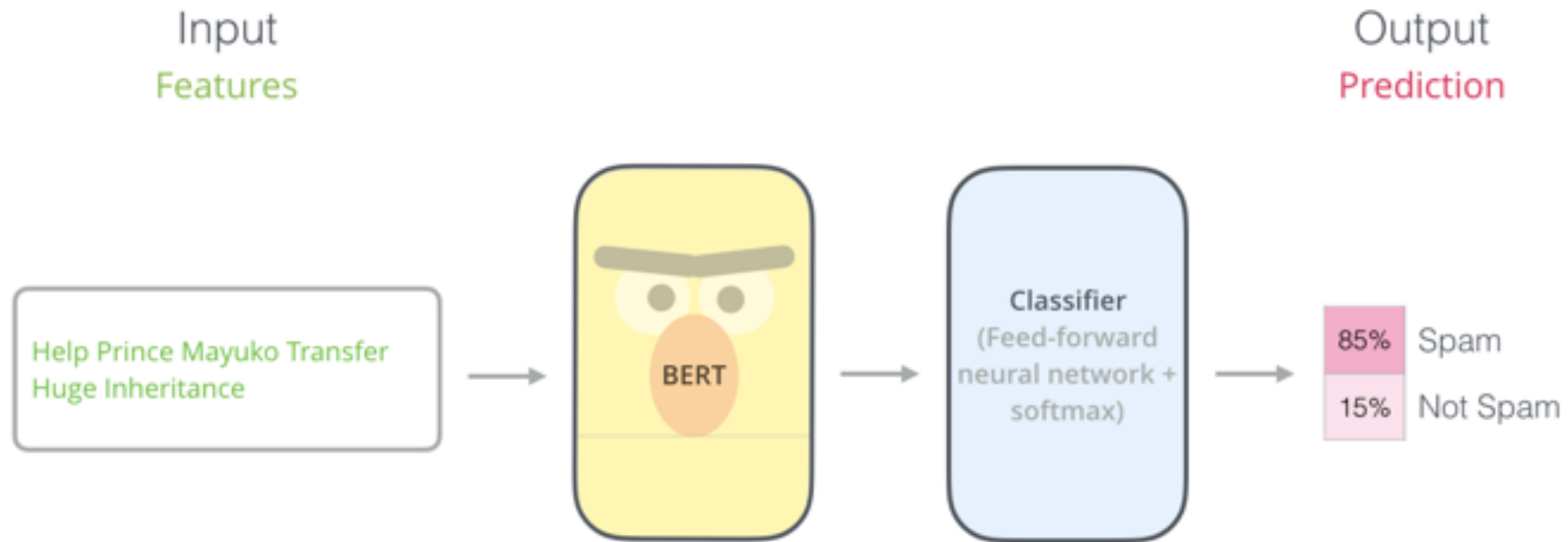
The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.



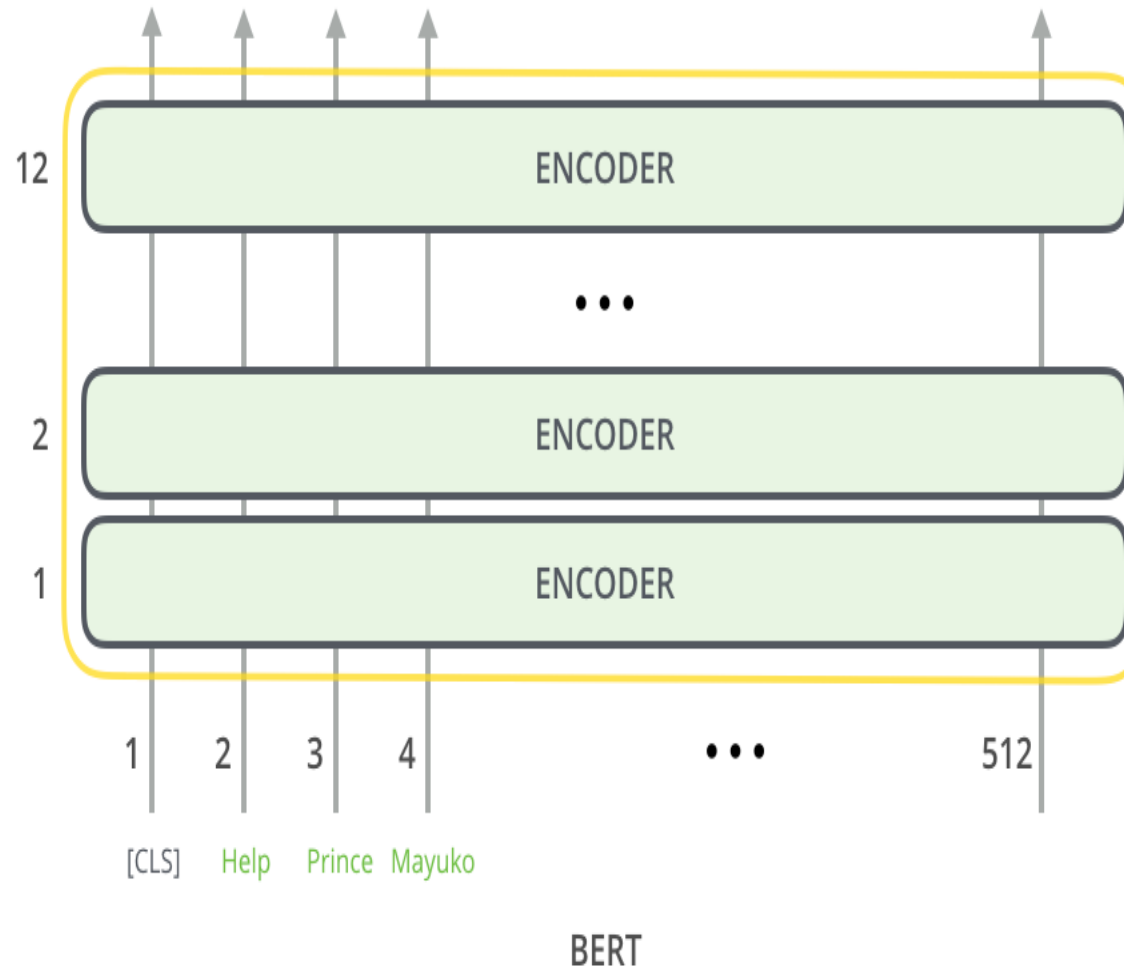
2 - **Supervised** training on a specific task with a labeled dataset.



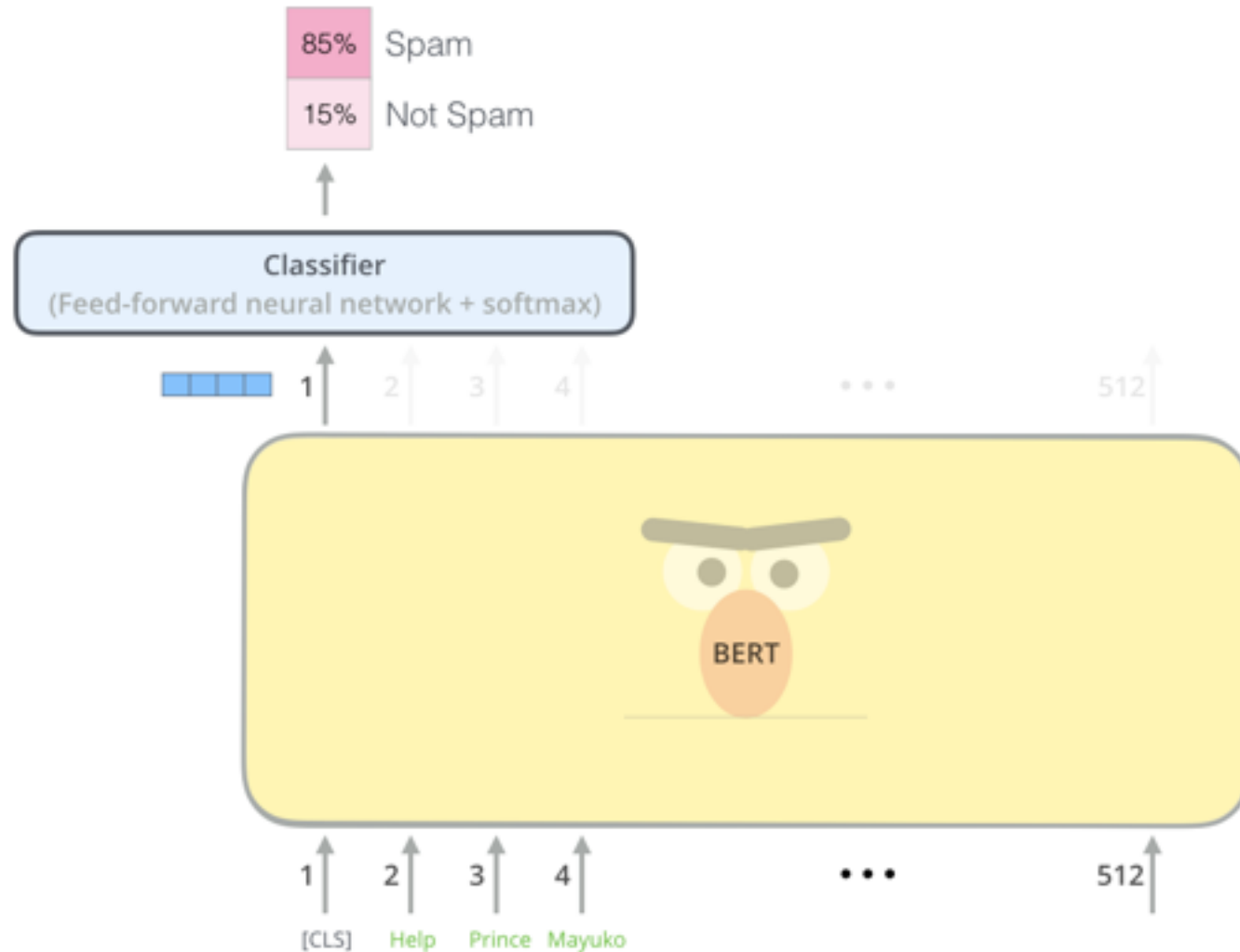
BERT Classification Input Output



BERT Encoder Input



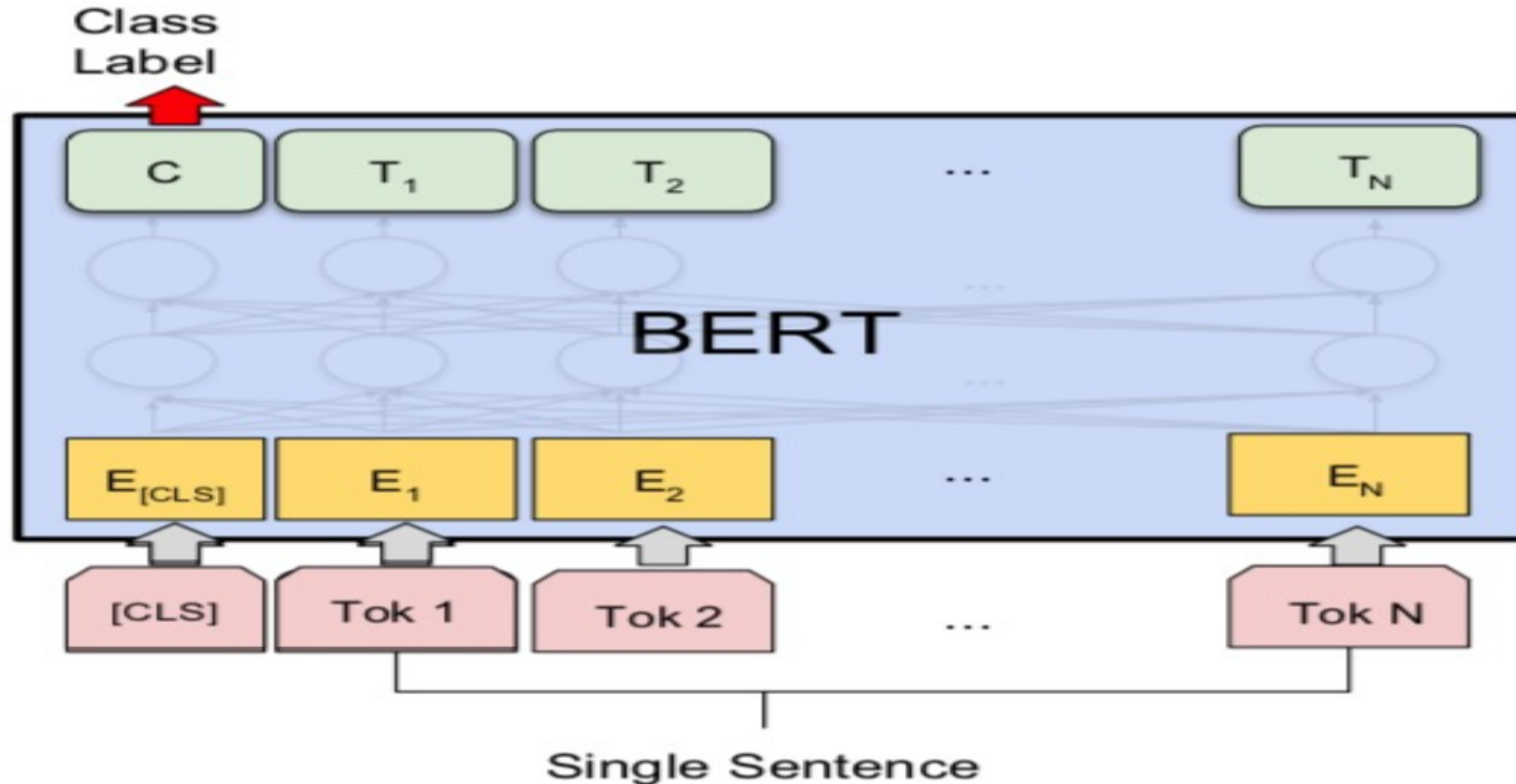
BERT Classifier



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

A Visual Guide to Using BERT for the First Time

(Jay Alammar, 2019)



Sentiment Classification: SST2

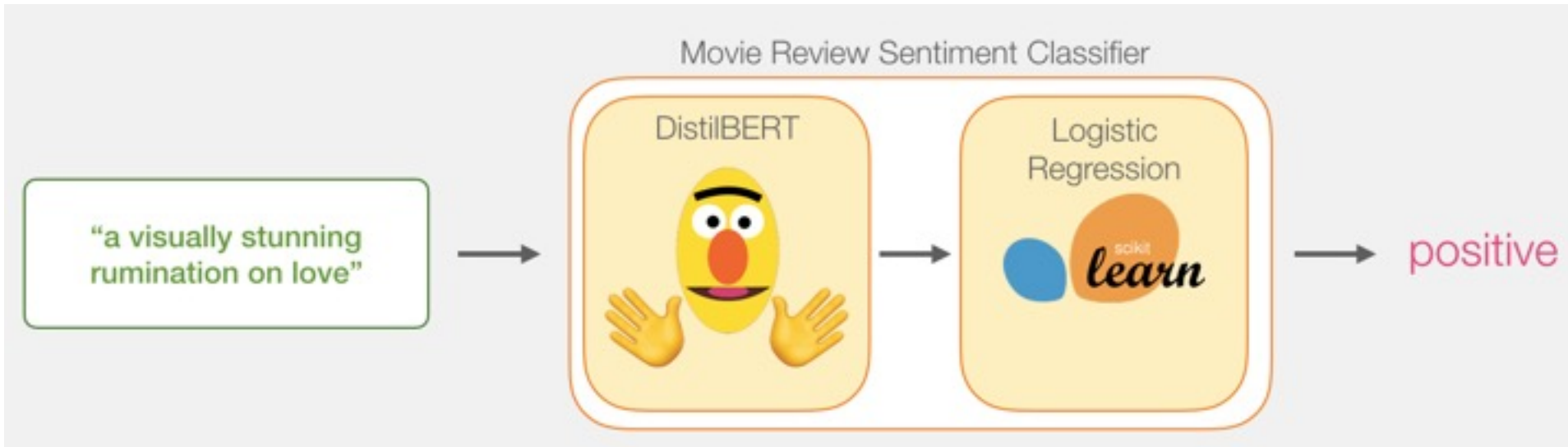
Sentences from movie reviews

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

Movie Review Sentiment Classifier

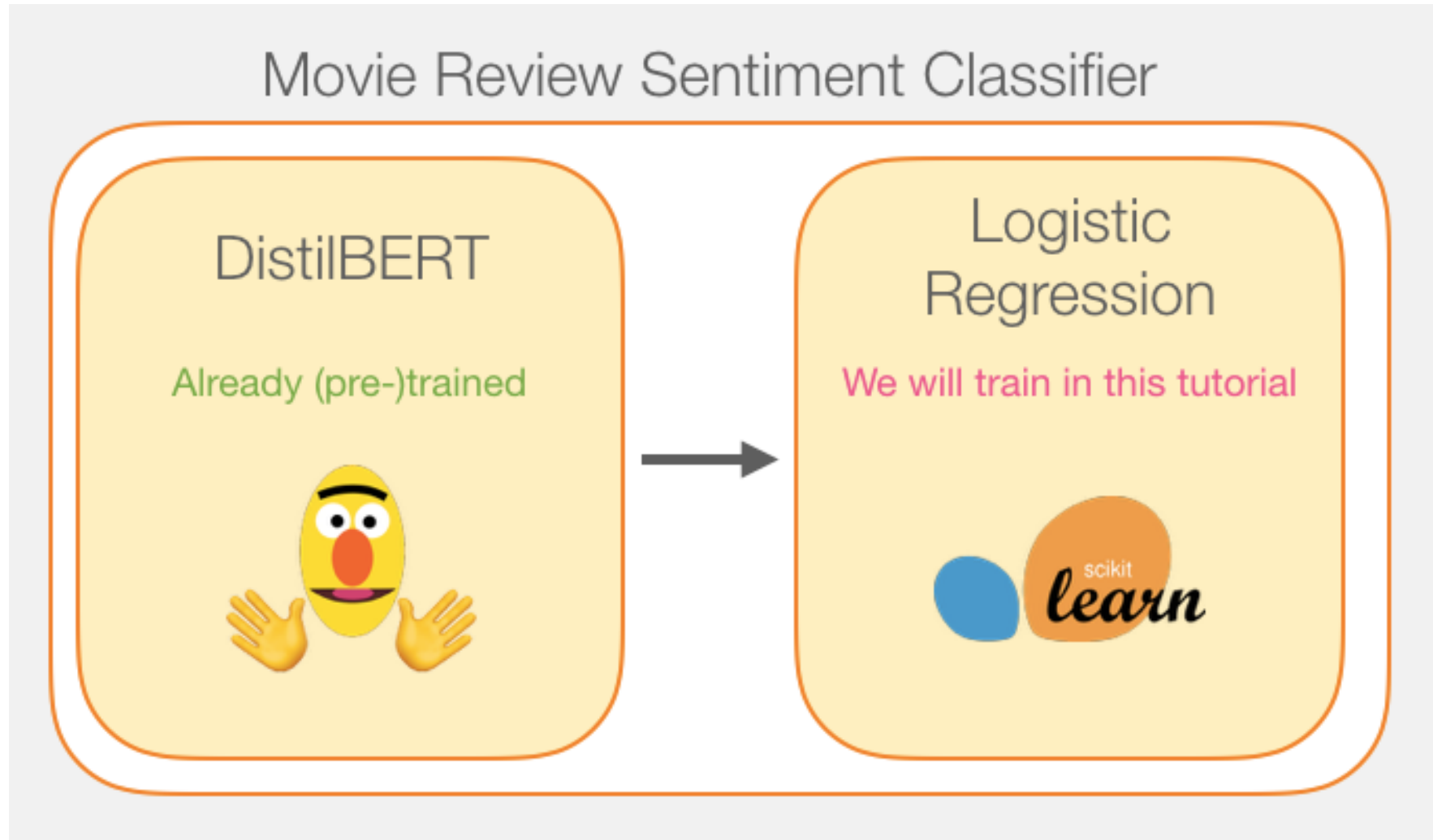


Movie Review Sentiment Classifier



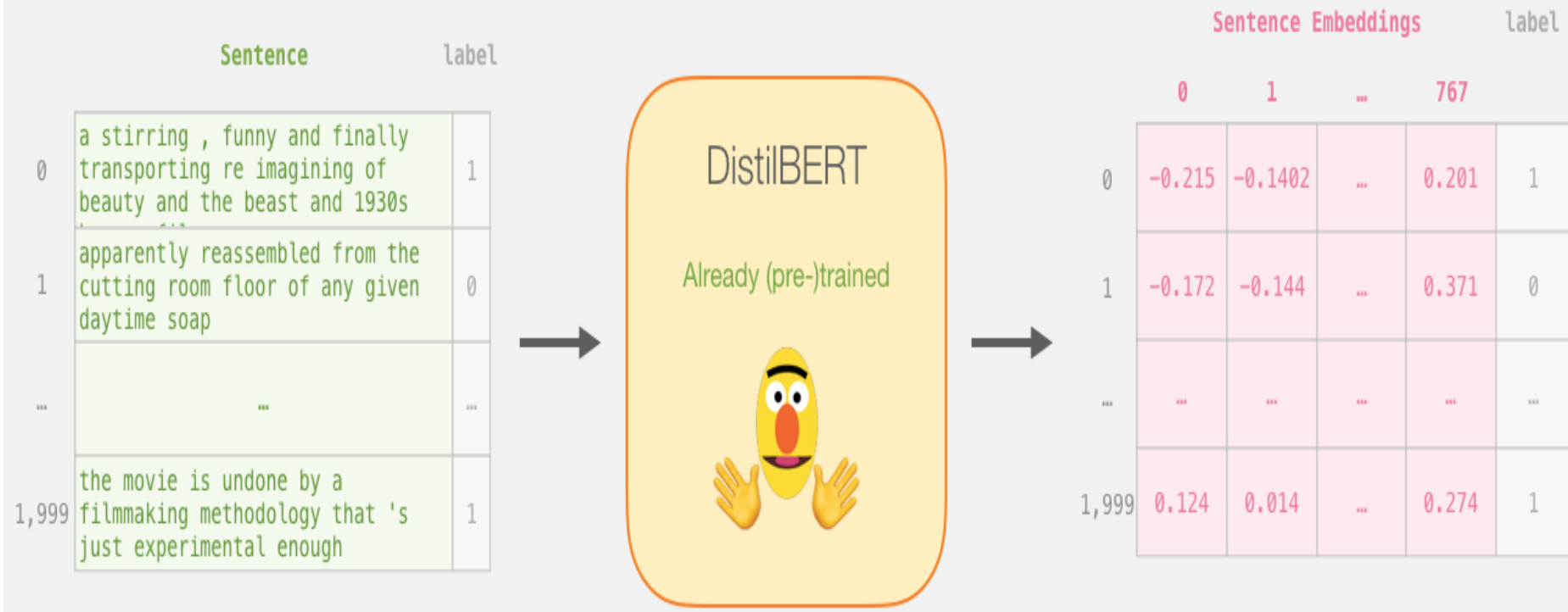
Movie Review Sentiment Classifier

Model Training



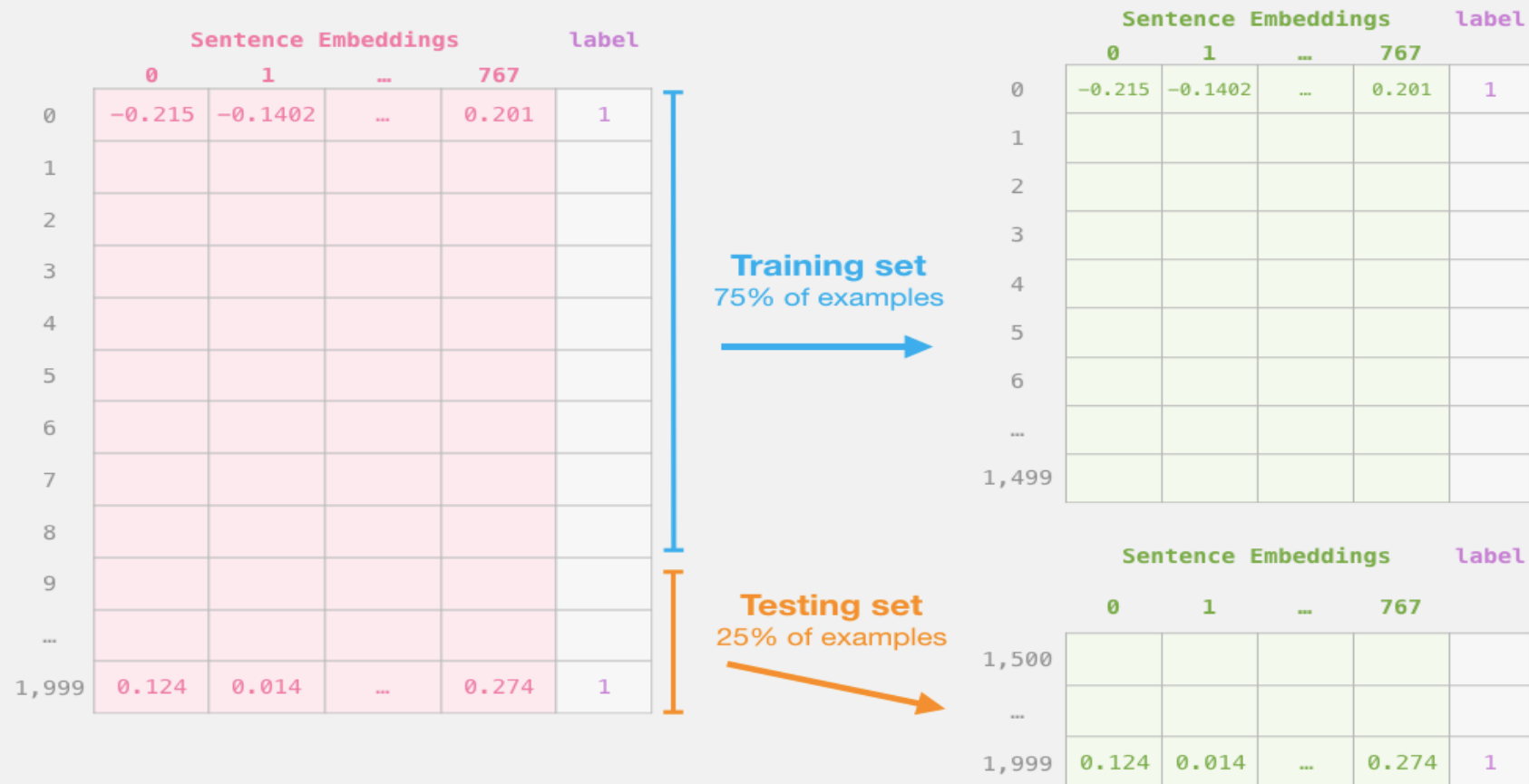
Step # 1 Use distilBERT to Generate Sentence Embeddings

Step #1: Use DistilBERT to embed all the sentences



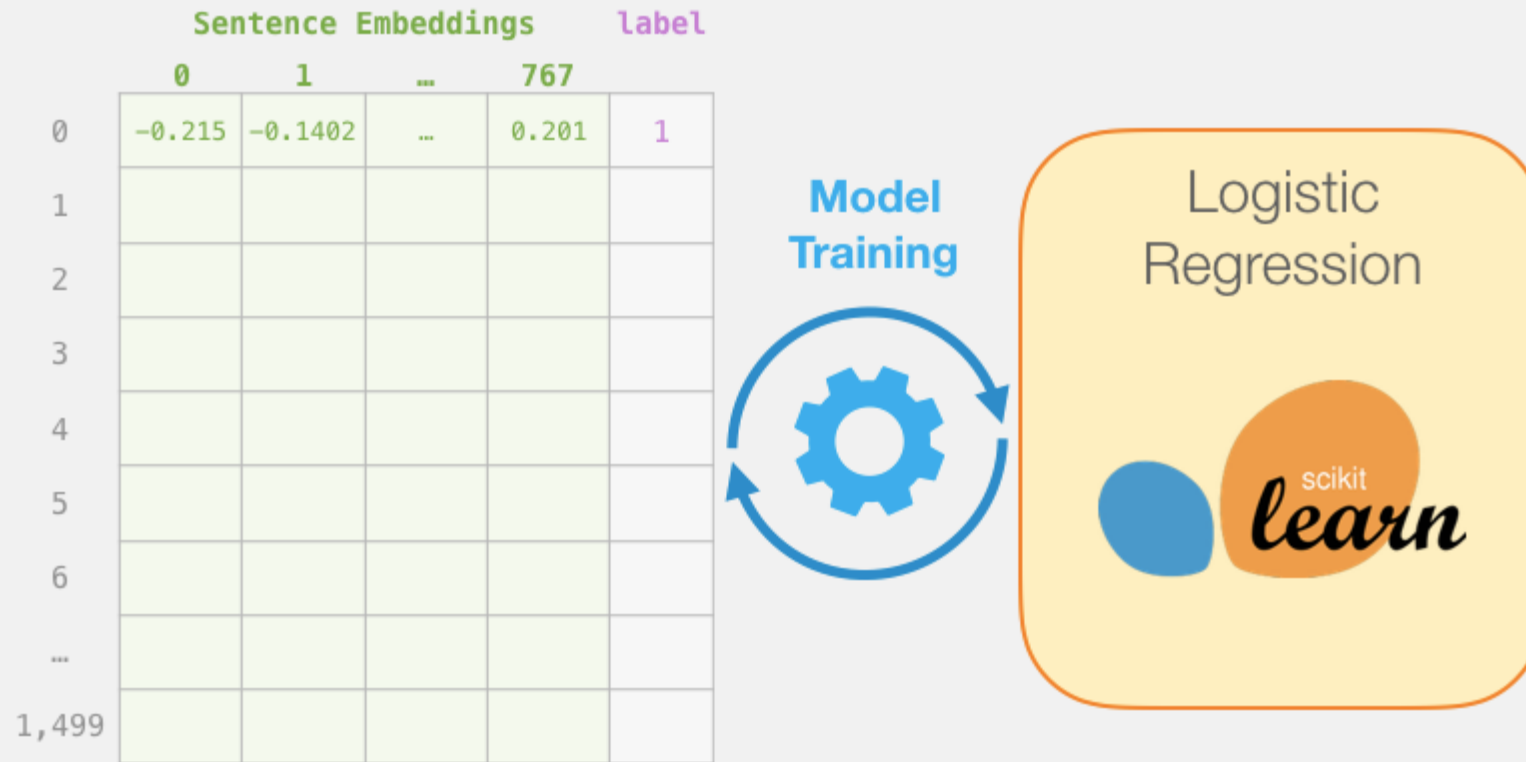
Step #2: Test/Train Split for Model #2, Logistic Regression

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



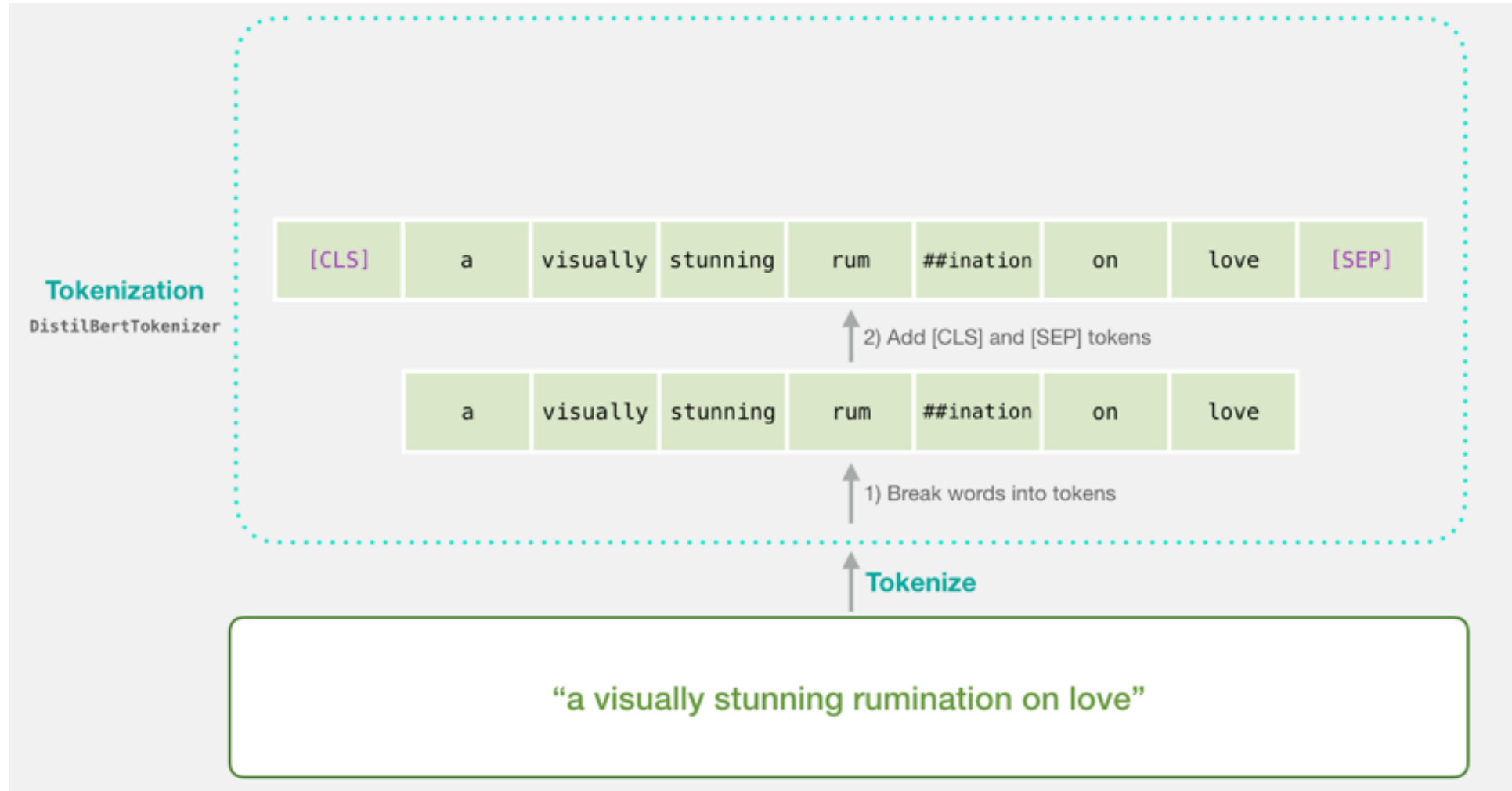
Step #3 Train the logistic regression model using the training set

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



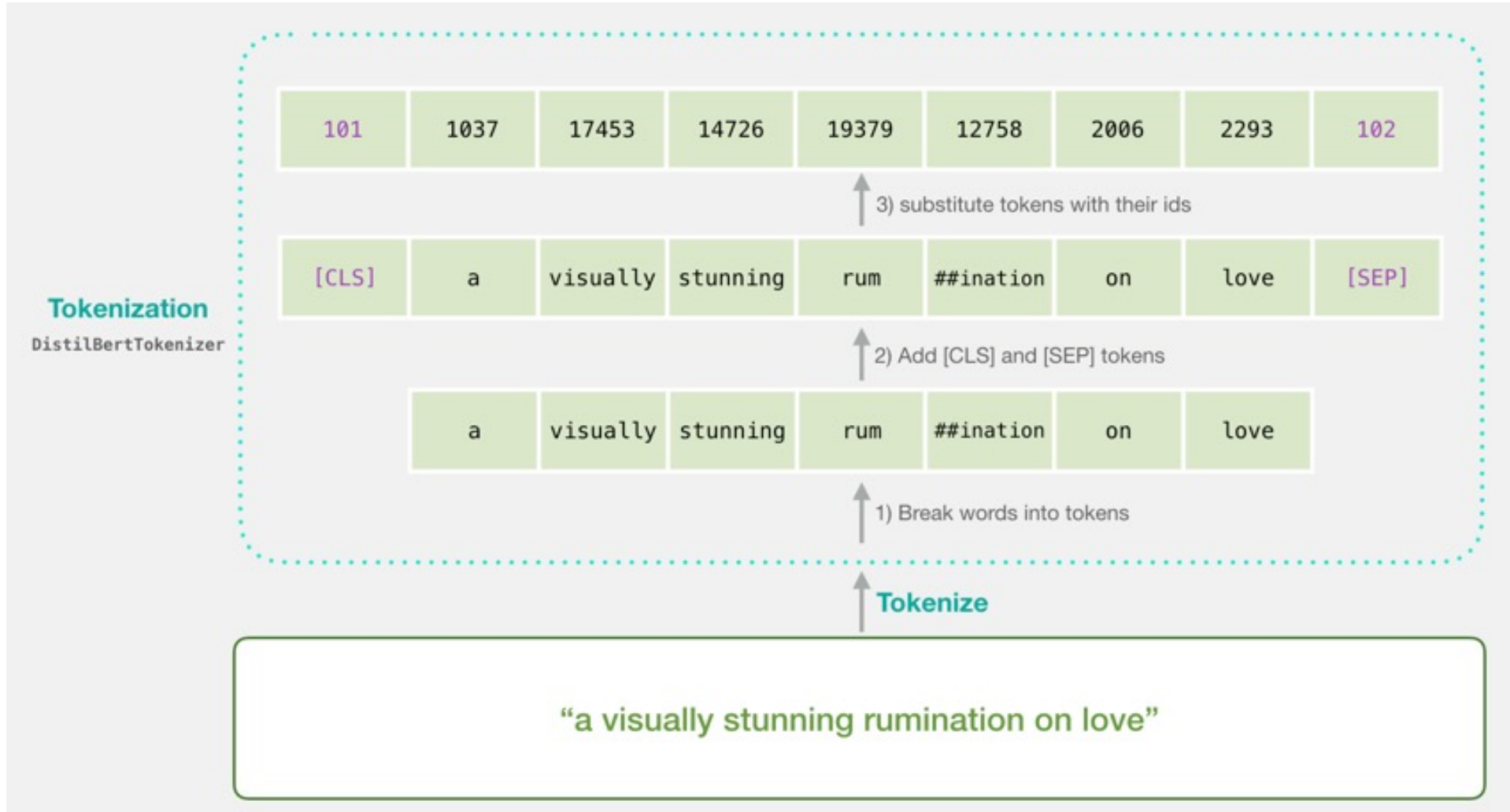
Tokenization

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

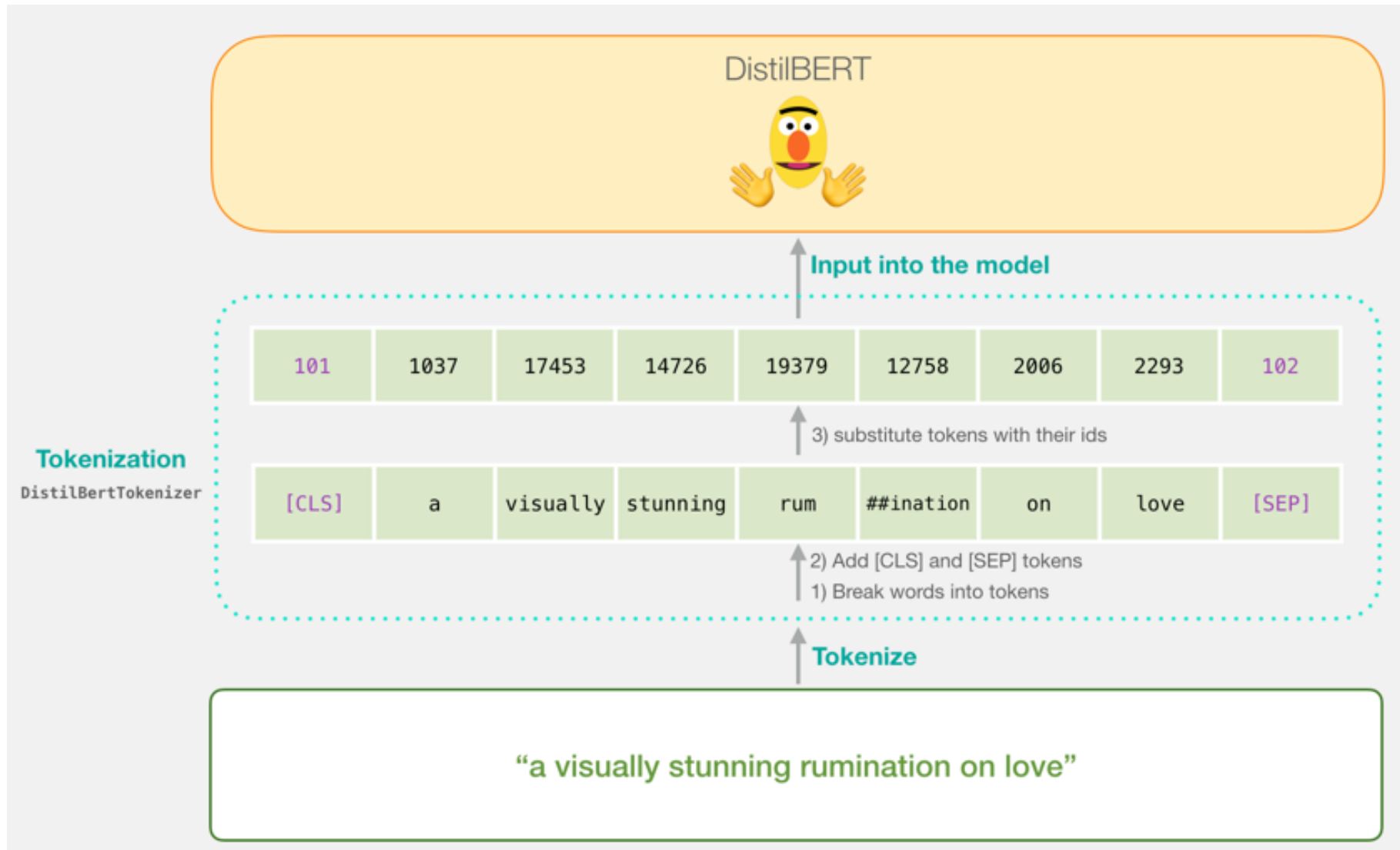


Tokenization

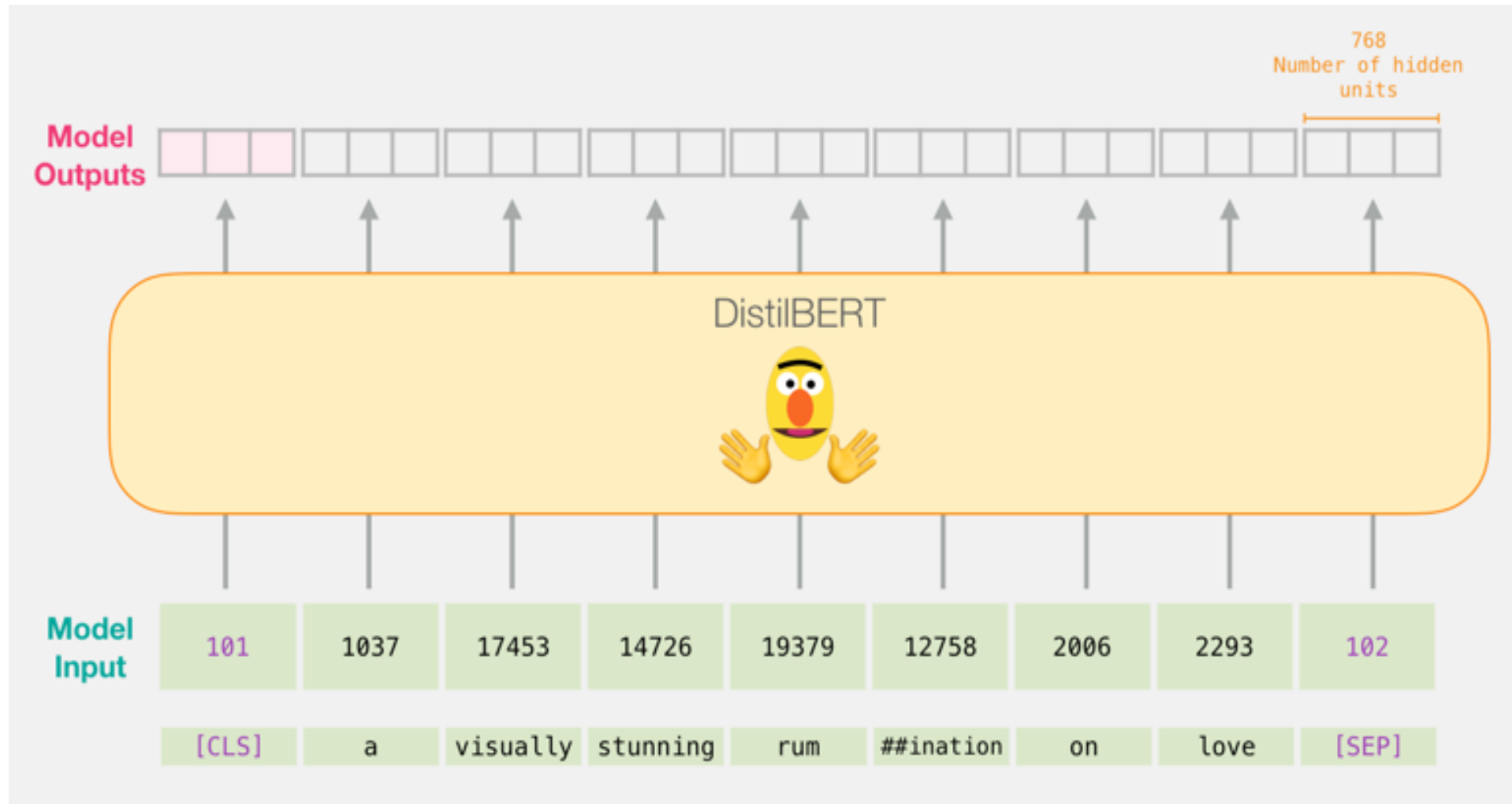
```
tokenizer.encode("a visually stunning rumination on love",  
                add_special_tokens=True)
```



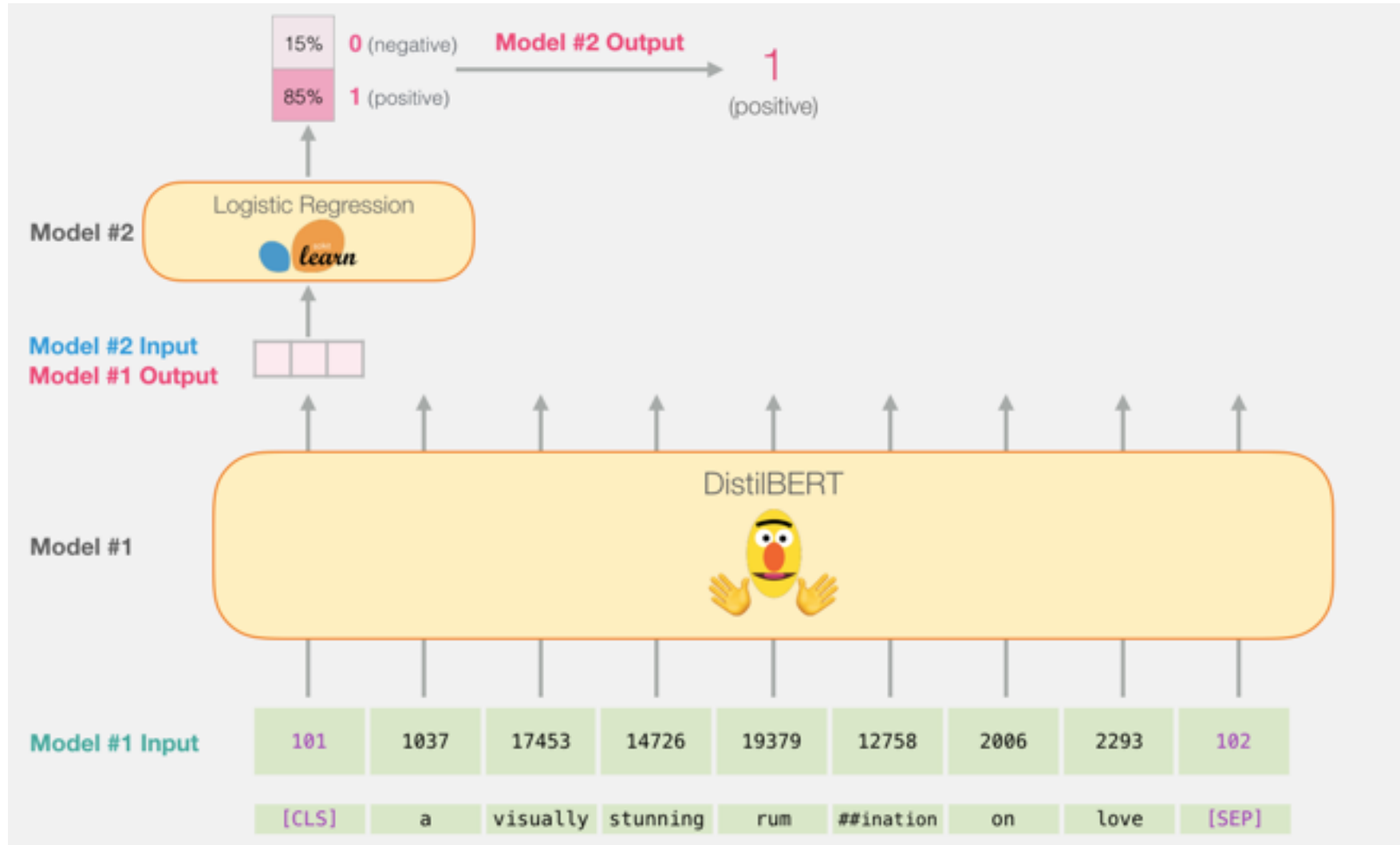
Tokenization for BERT Model



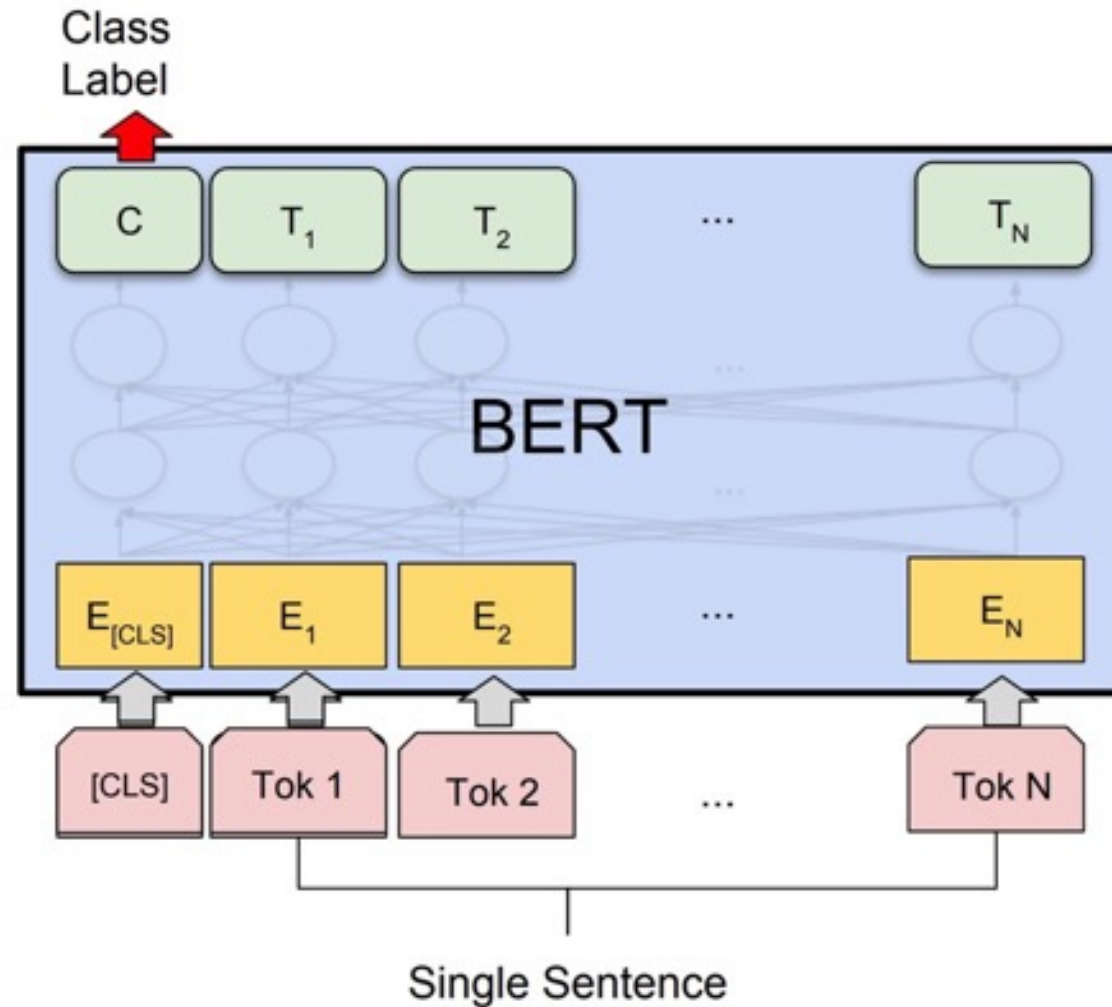
Flowing Through DistilBERT (768 features)



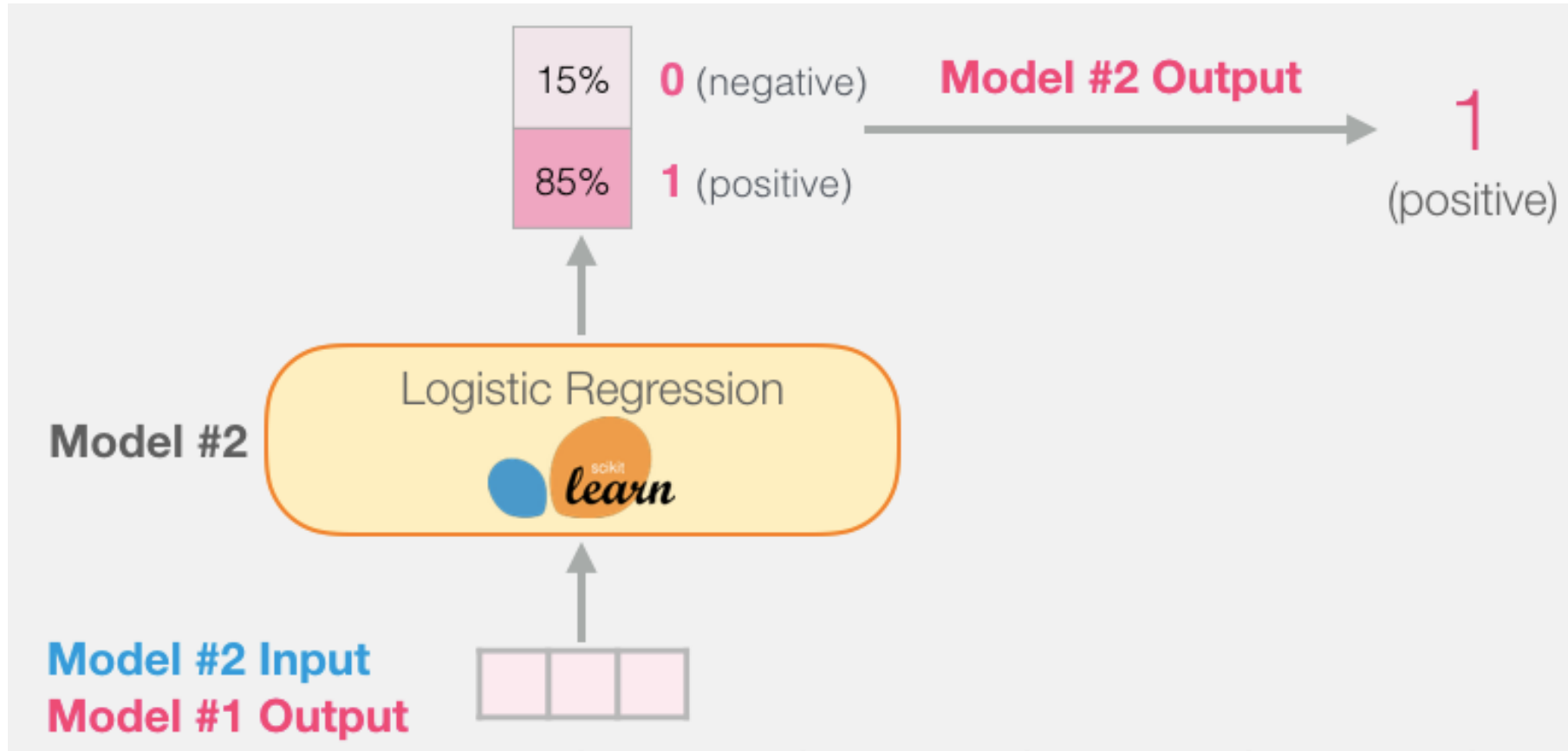
Model #1 Output **Class** vector as Model #2 Input



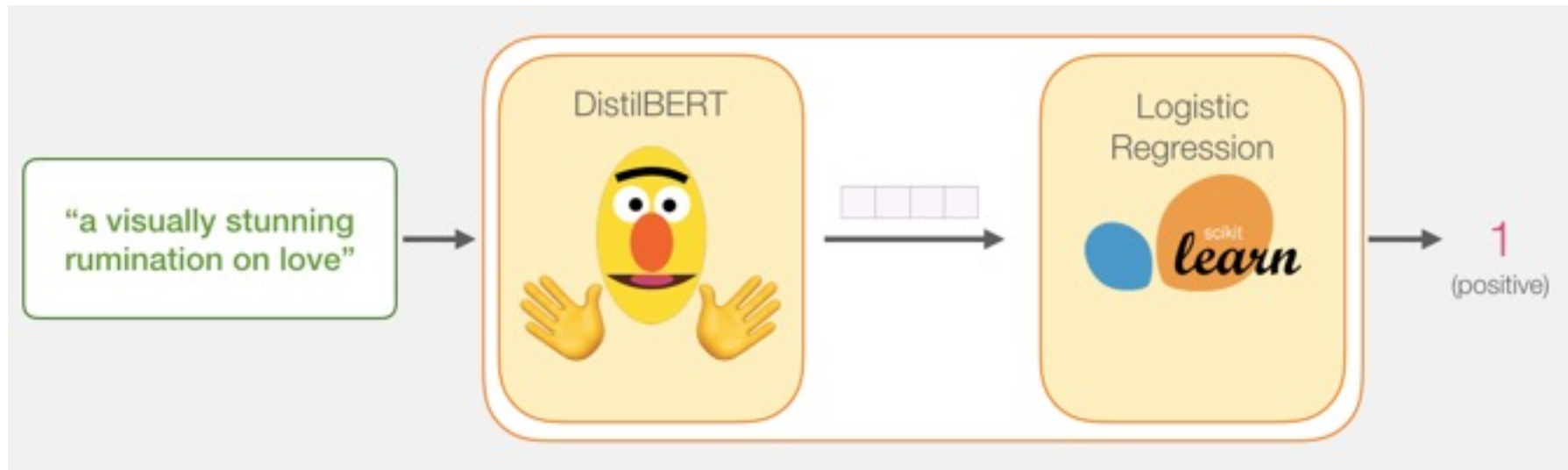
Fine-tuning BERT on Single Sentence Classification Tasks



Model #1 Output Class vector as Model #2 Input



Logistic Regression Model to classify **Class** vector

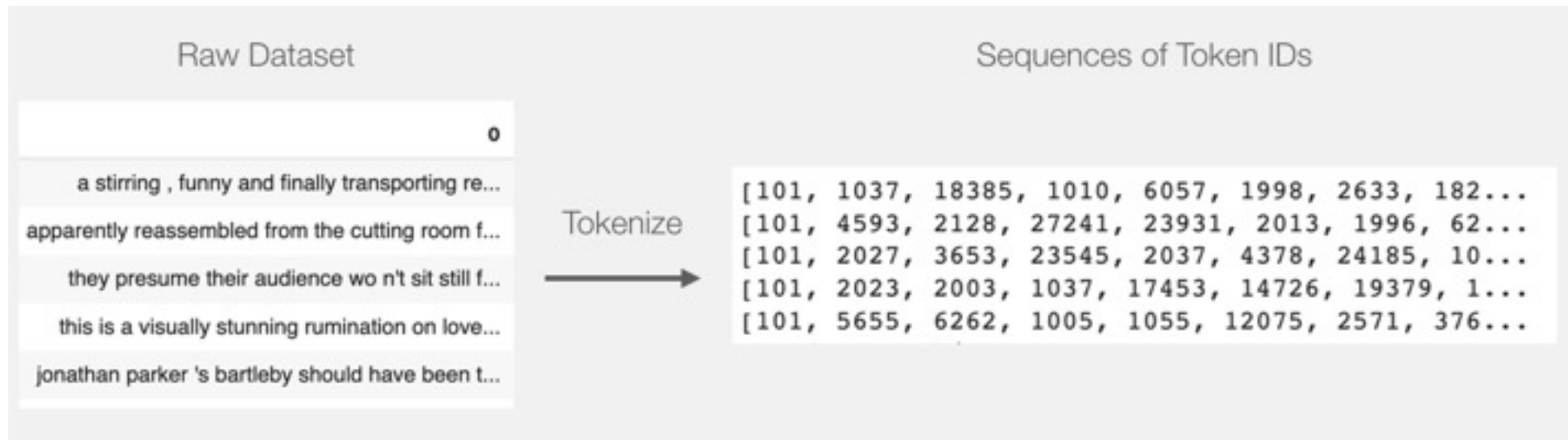



```
df = pd.read_csv('https://github.com/clairett/pytorch-  
sentiment-classification/raw/master/data/SST2/train.tsv',  
delimiter='\t', header=None)  
  
df.head()
```

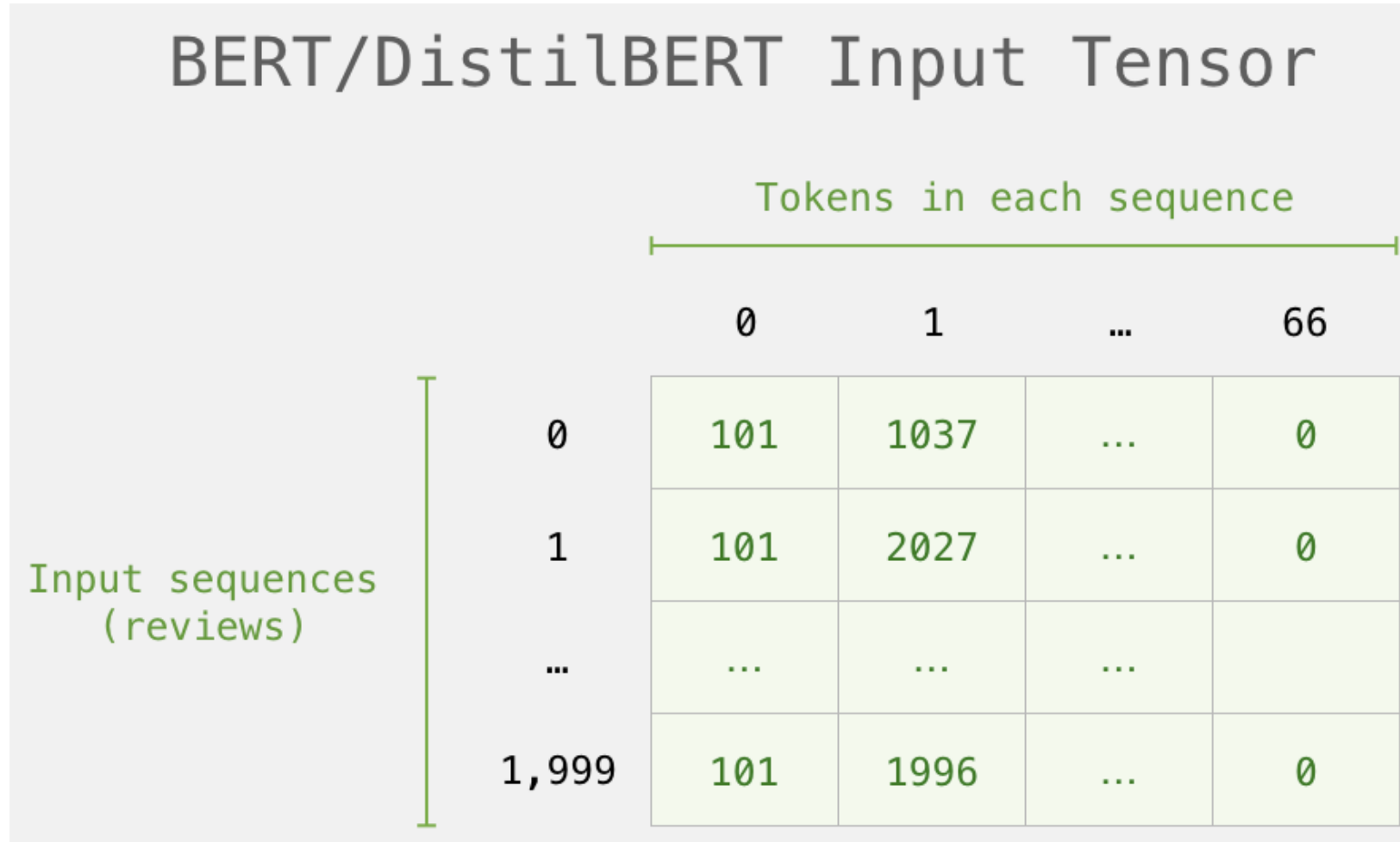
		0	1
0	a stirring , funny and finally transporting re...		1
1	apparently reassembled from the cutting room f...		0
2	they presume their audience wo n't sit still f...		0
3	this is a visually stunning rumination on love...		1
4	jonathan parker 's bartleby should have been t...		1

Tokenization

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

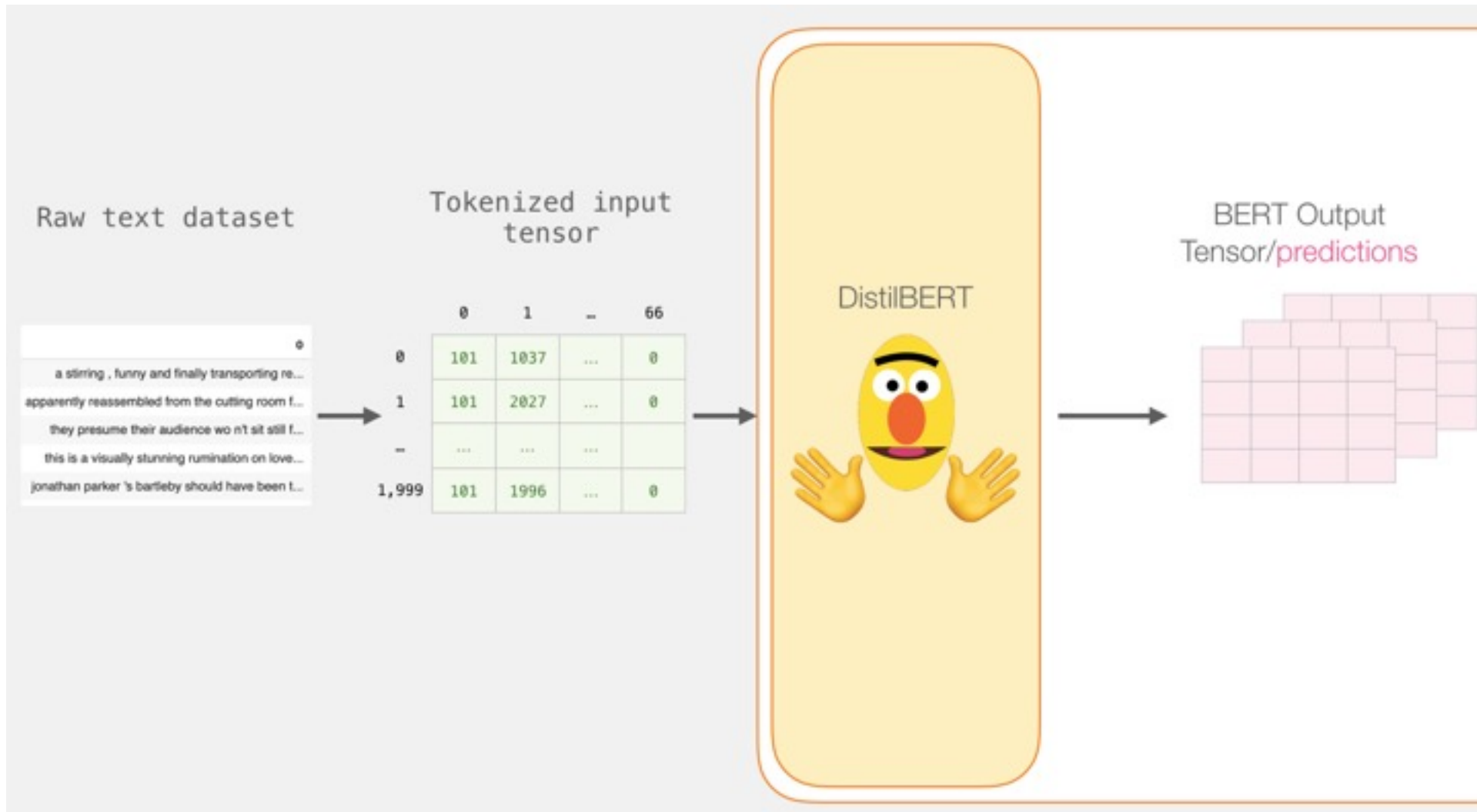


BERT Input Tensor

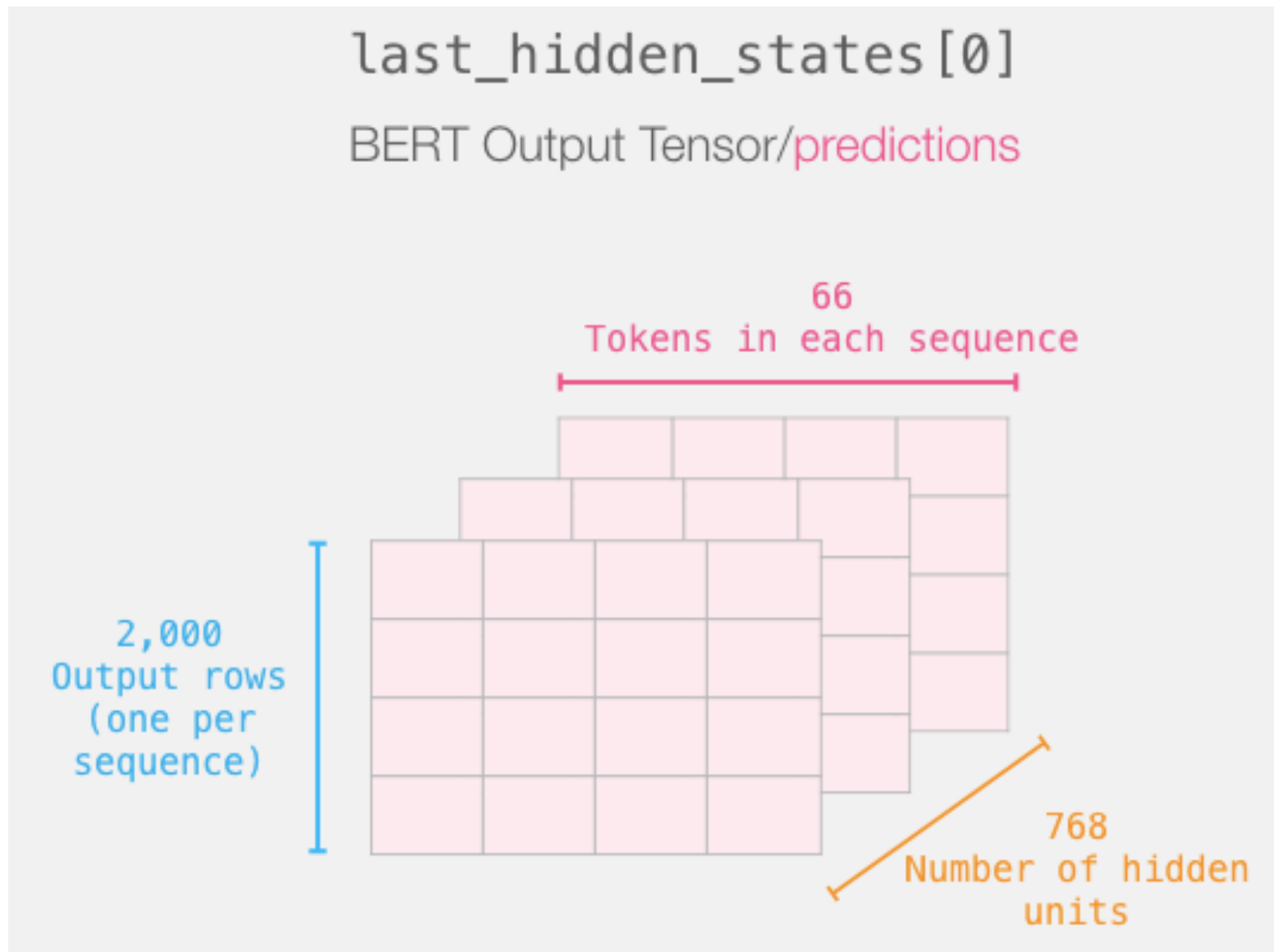


Processing with DistilBERT

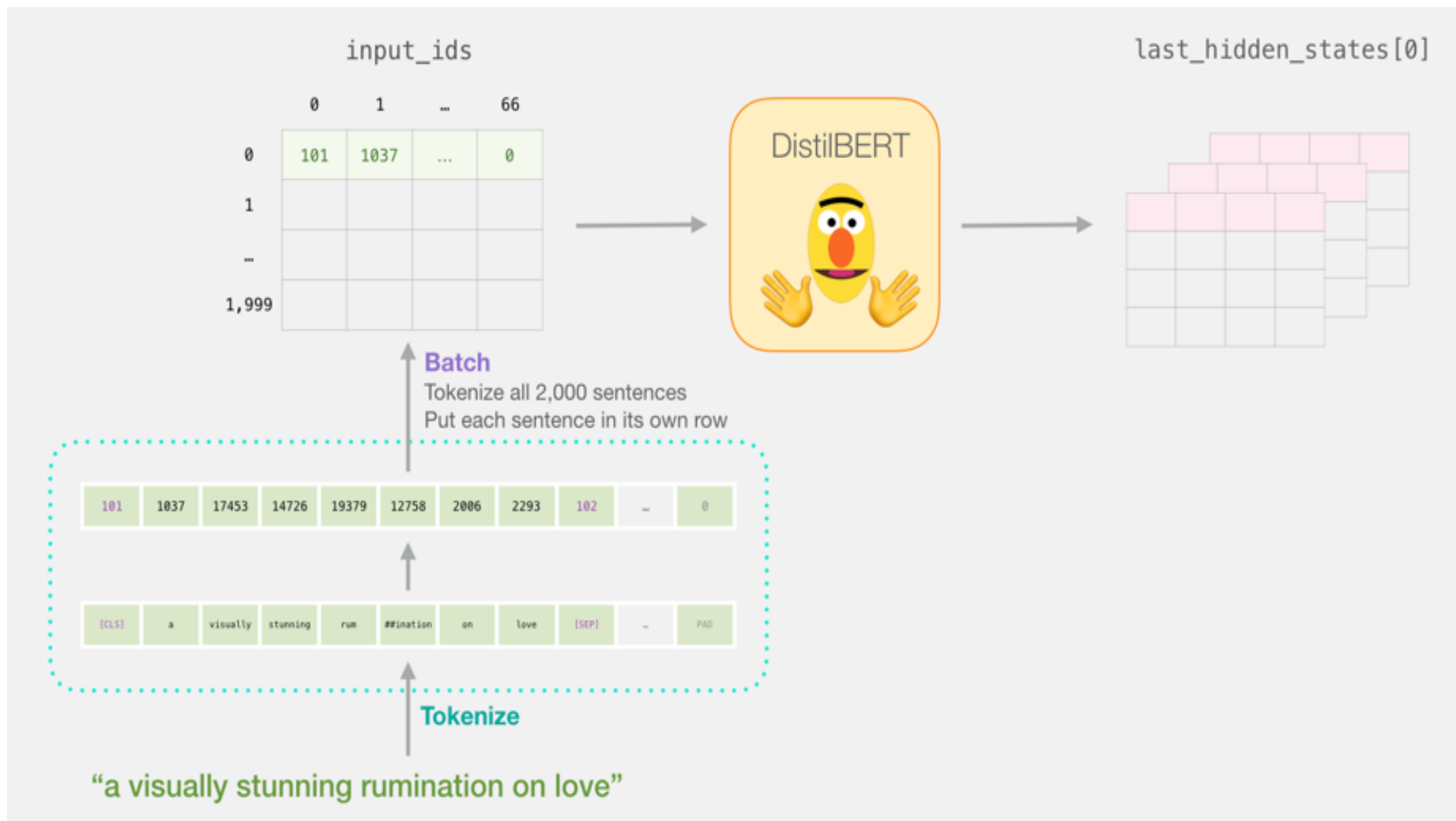
```
input_ids = torch.tensor(np.array(padded))  
last_hidden_states = model(input_ids)
```



Unpacking the BERT output tensor



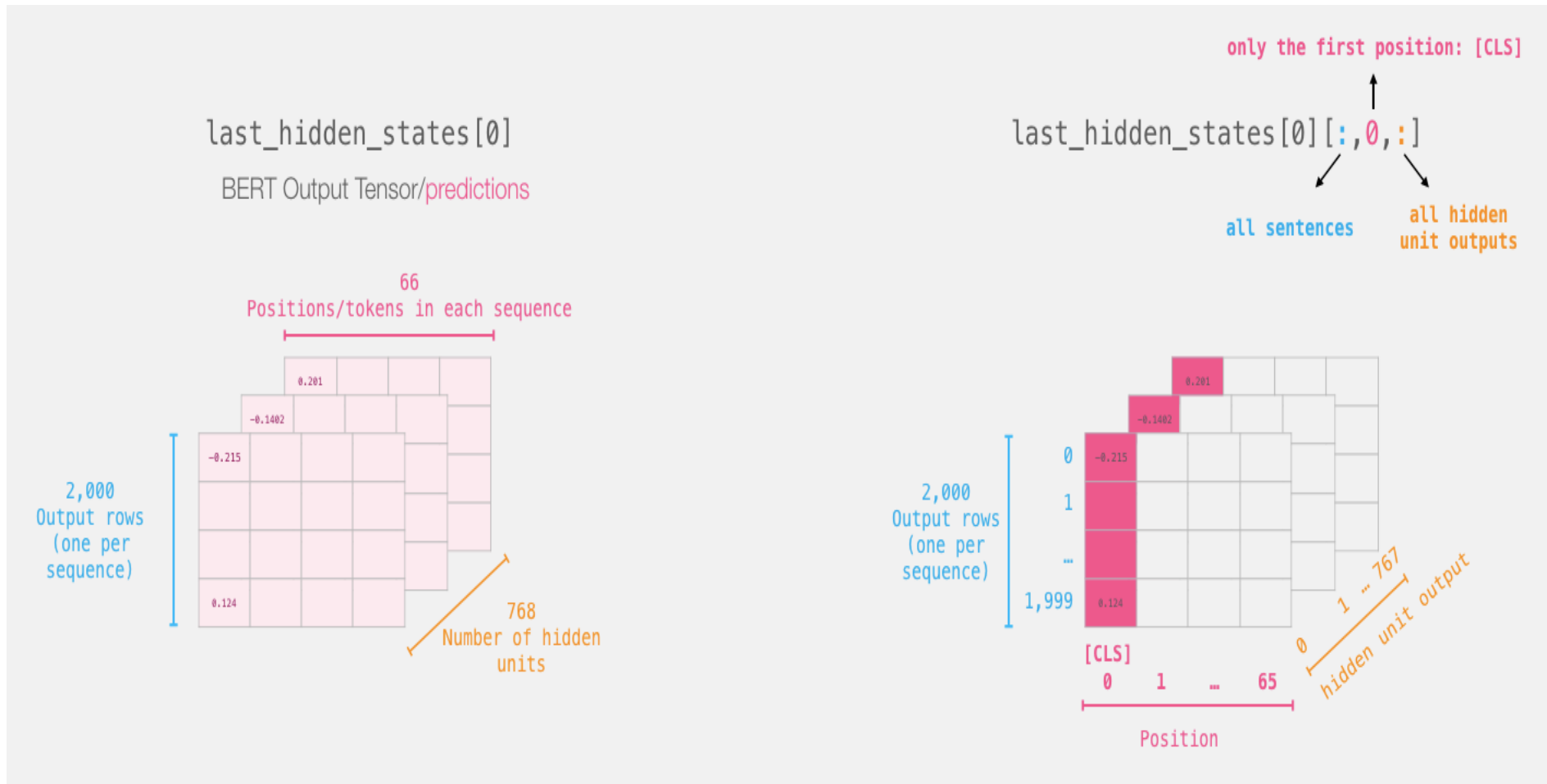
Sentence to last_hidden_state[0]



BERT's output for the [CLS] tokens

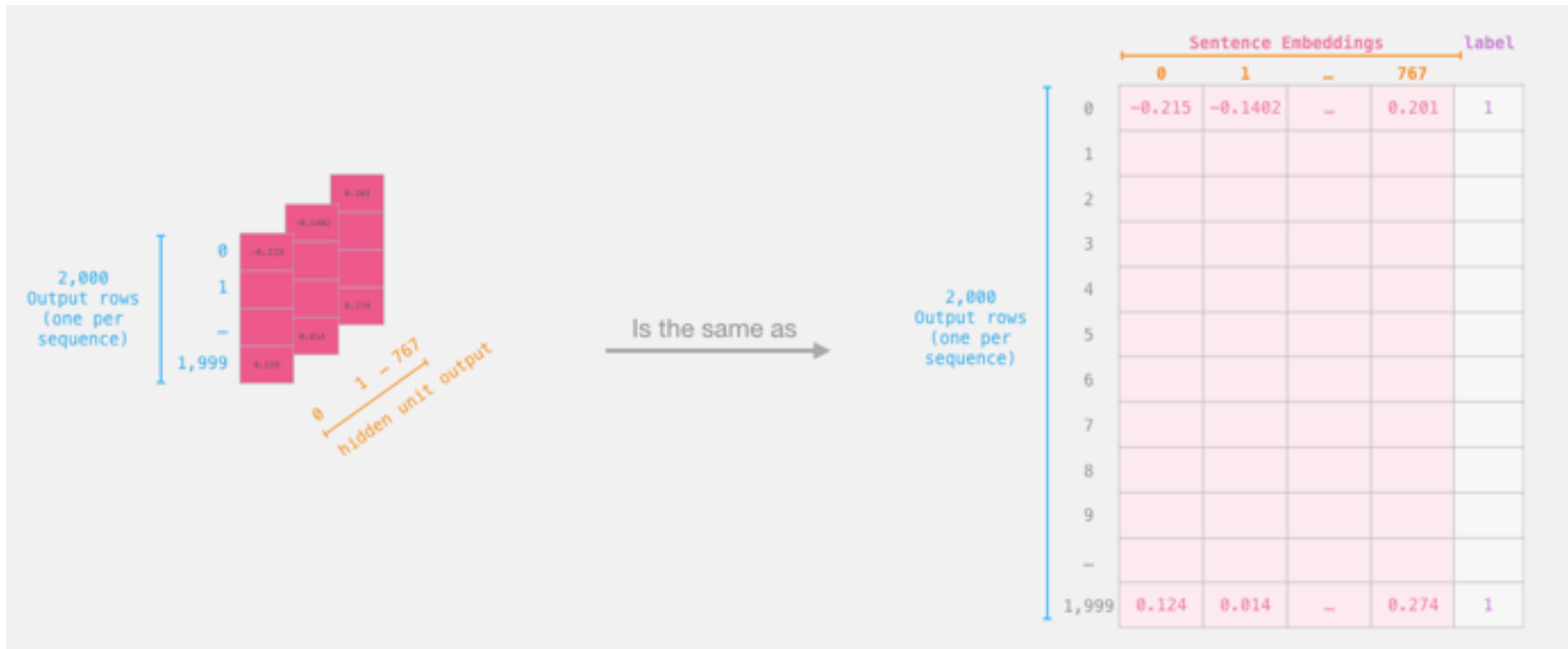
```
# Slice the output for the first position for all the
sequences, take all hidden unit outputs
```

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



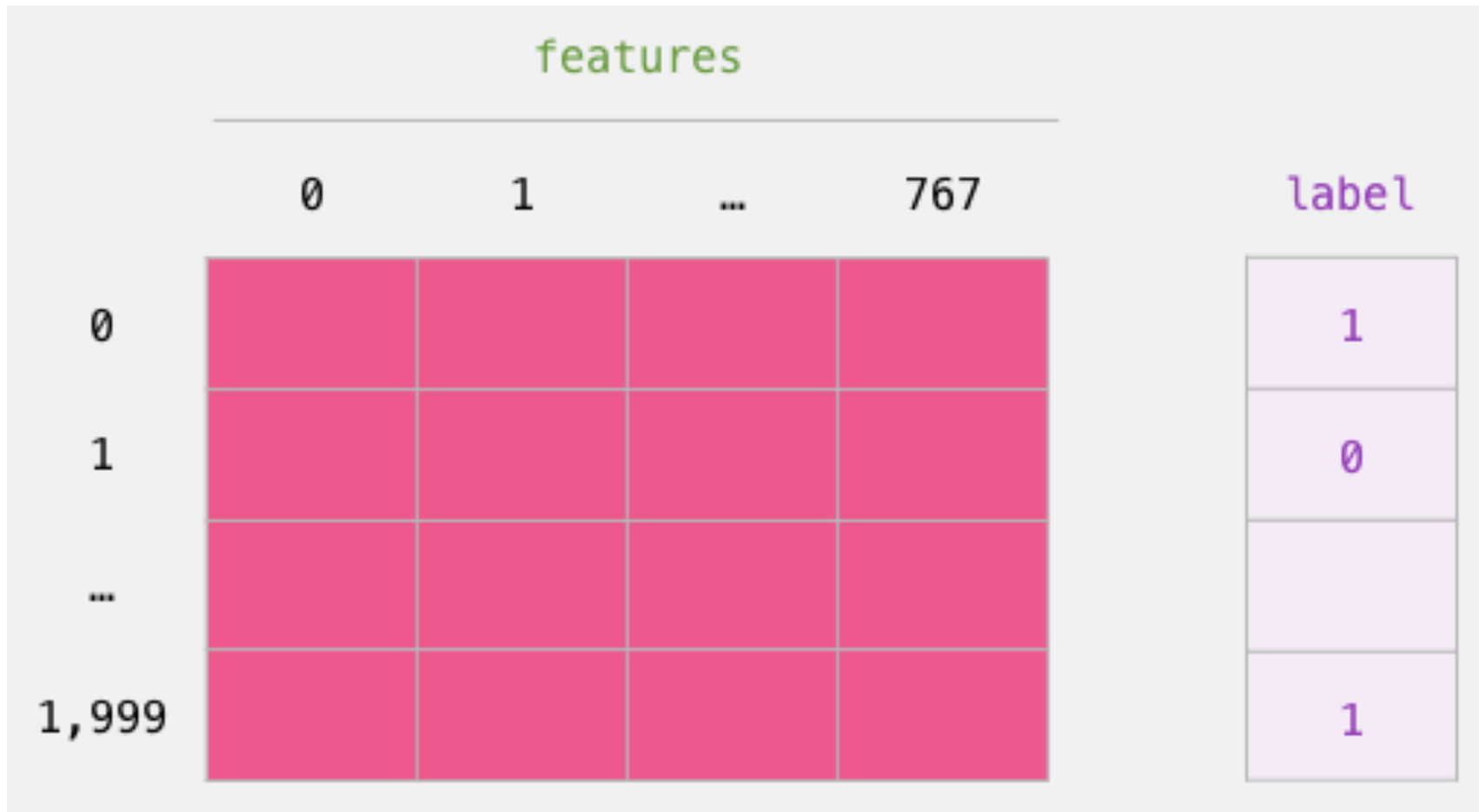
The tensor sliced from BERT's output

Sentence Embeddings



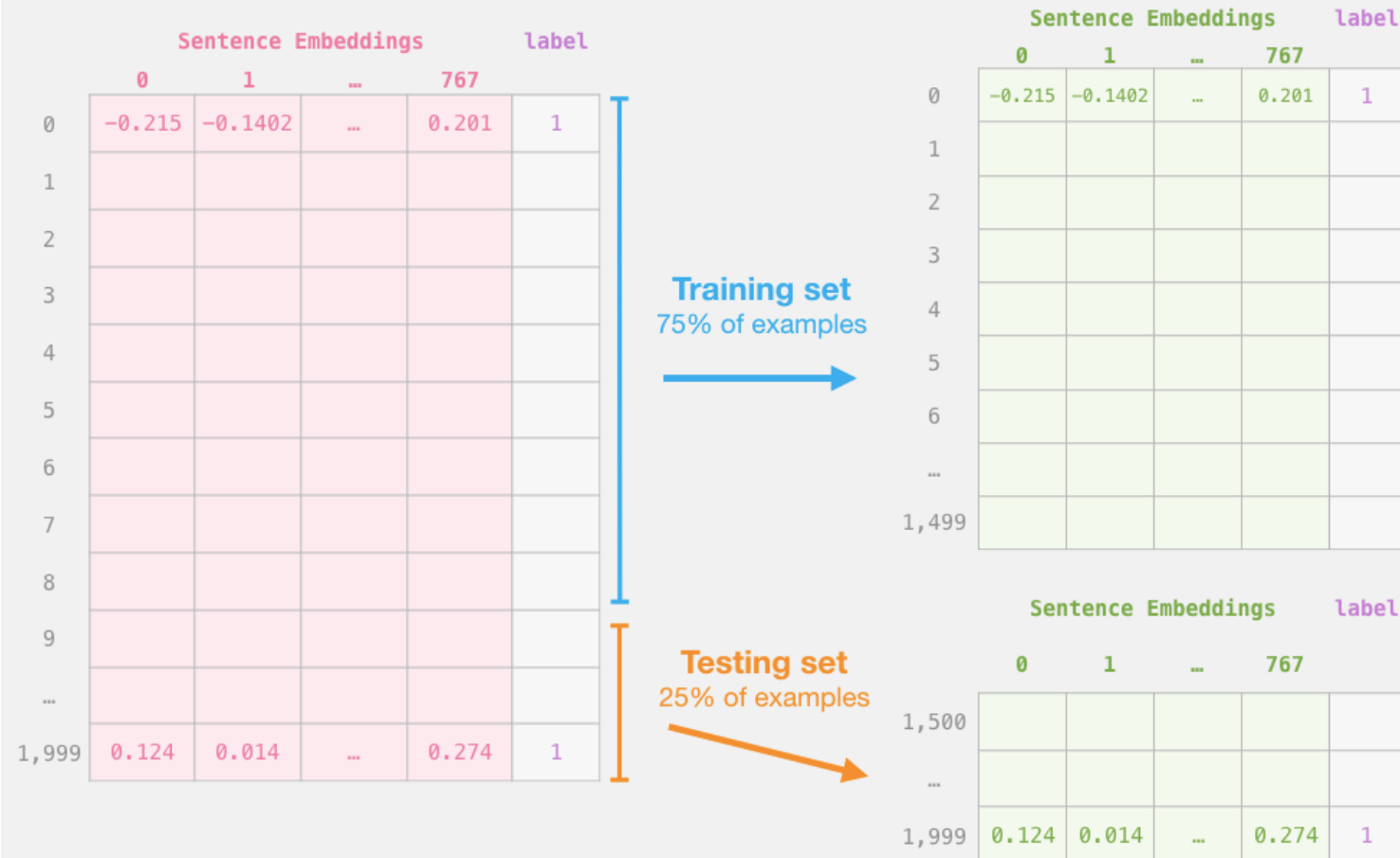
Dataset for Logistic Regression (768 Features)

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




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

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



Score Benchmarks

Logistic Regression Model on SST-2 Dataset

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

#Testing
lr_clf.score(test_features, test_labels)

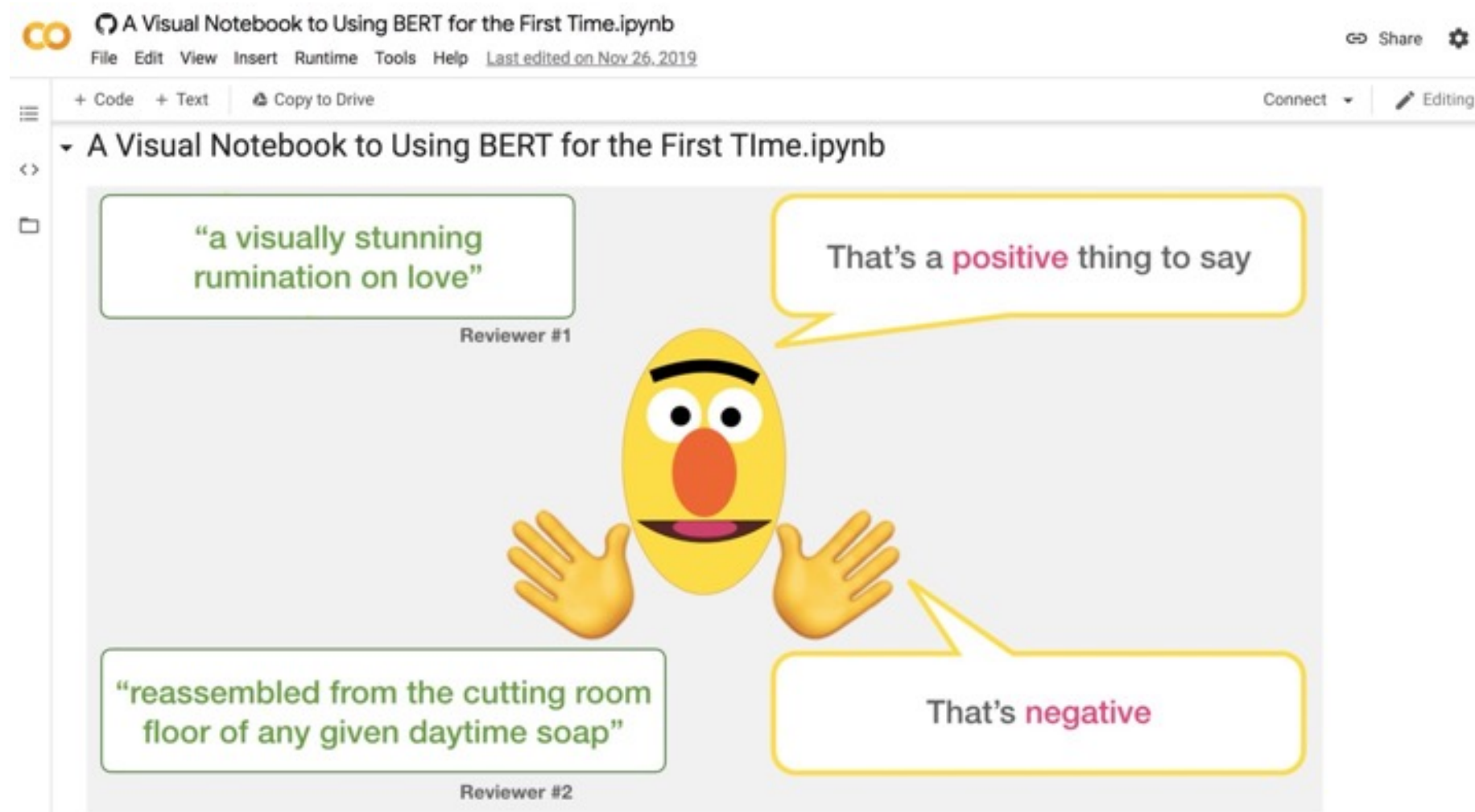
# Accuracy: 81%
# Highest accuracy: 96.8%
# Fine-tuned DistilBERT: 90.7%
# Full size BERT model: 94.9%
```


Sentiment Classification: SST2

Sentences from movie reviews

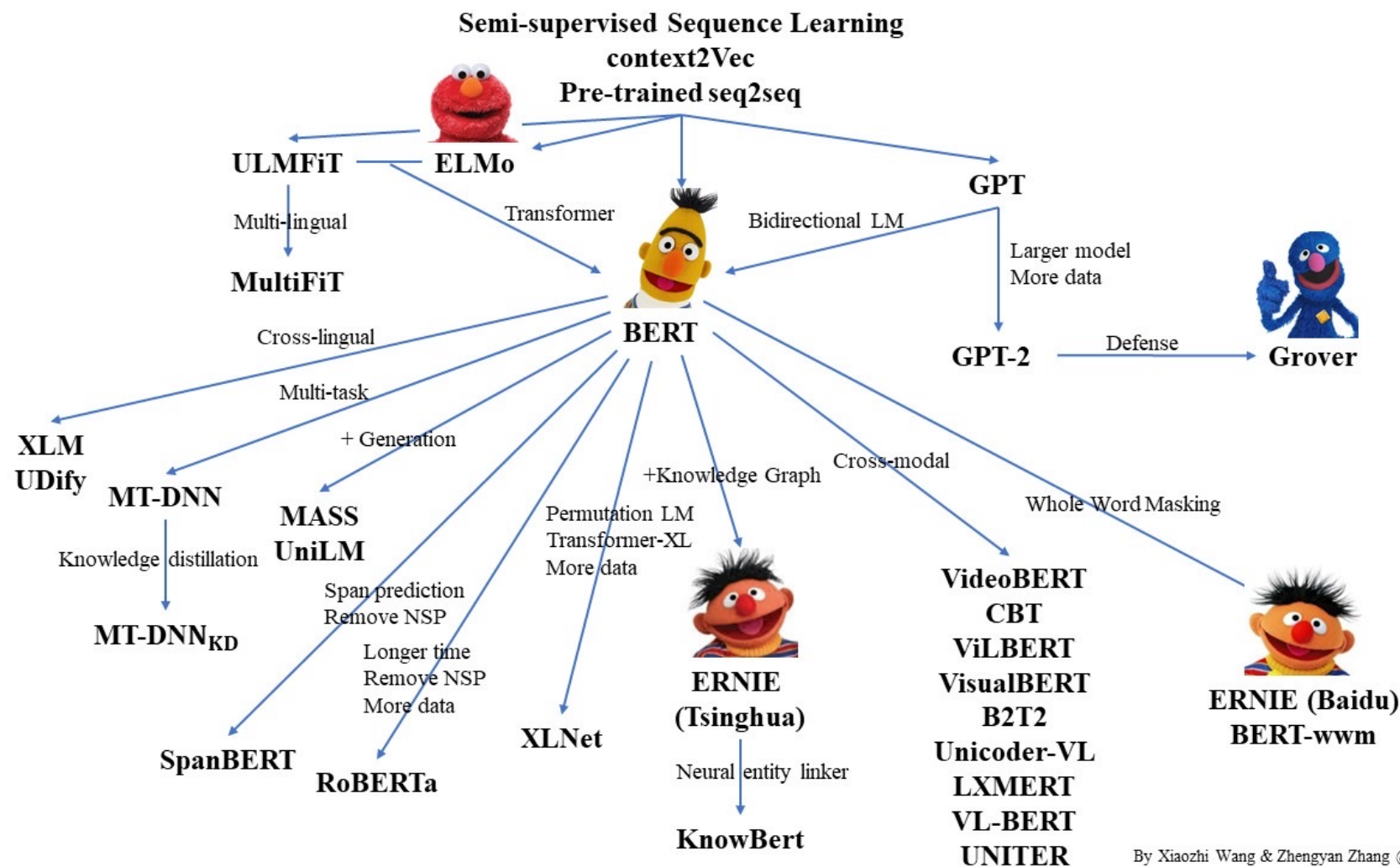
sentence	label
a stirring , funny and finally transporting re imagining of beauty and the beast and 1930s horror films	1
apparently reassembled from the cutting room floor of any given daytime soap	0
they presume their audience won't sit still for a sociology lesson	0
this is a visually stunning rumination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

A Visual Notebook to Using BERT for the First Time



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)

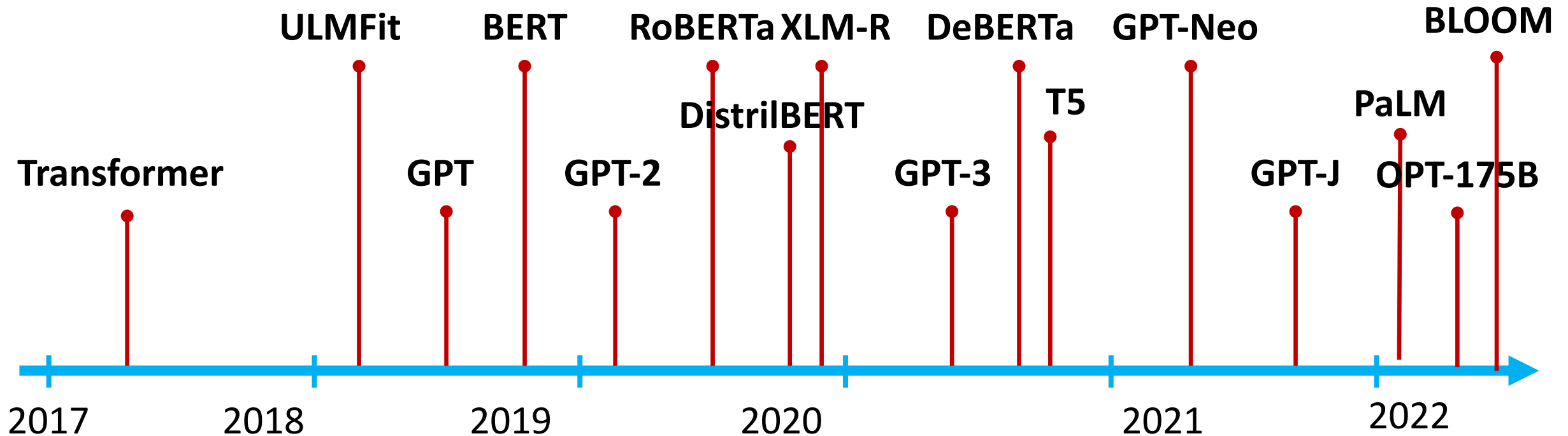


By Xiaozhi Wang & Zhengyan Zhang @THUNLP

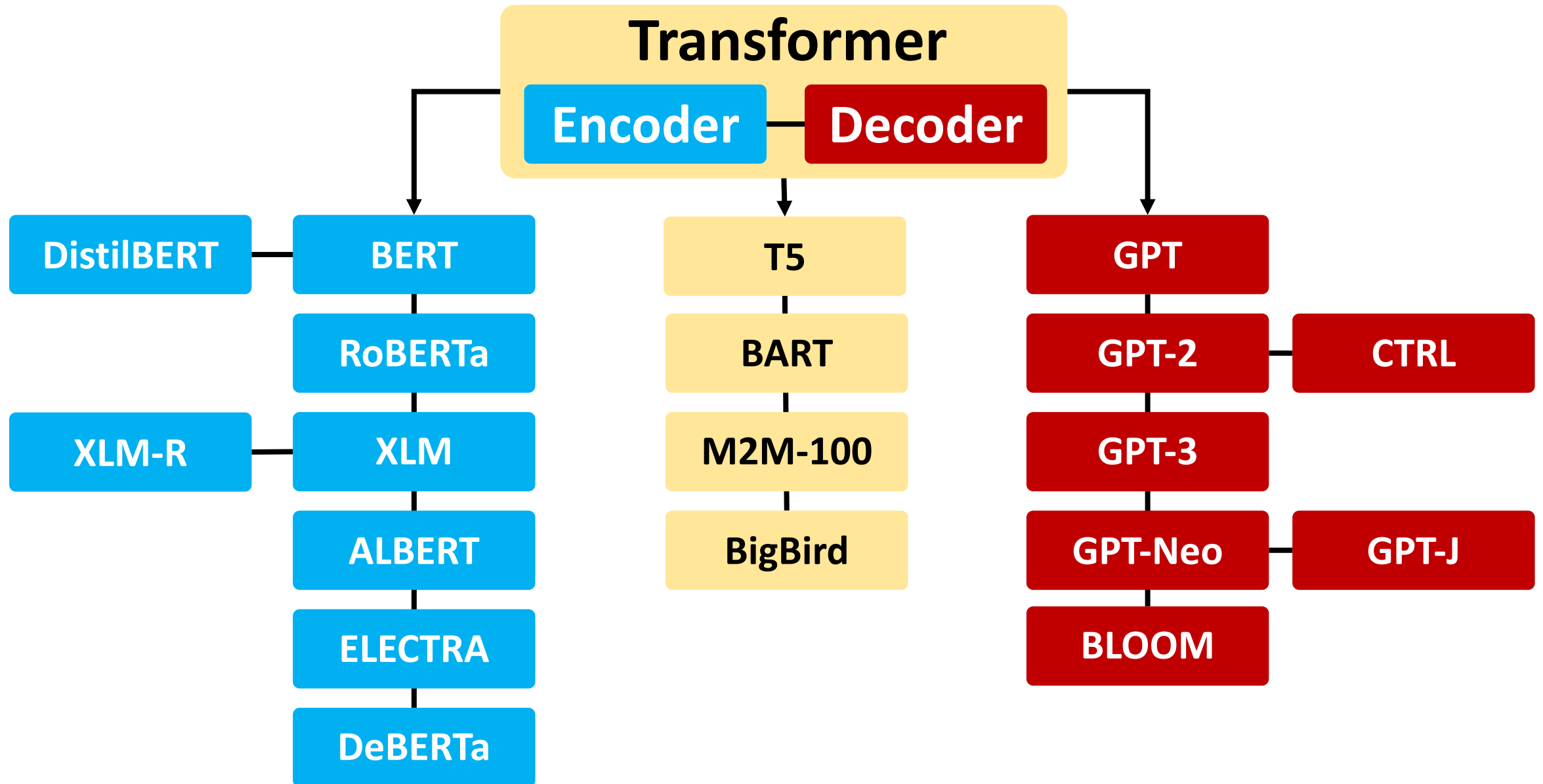
Outline

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

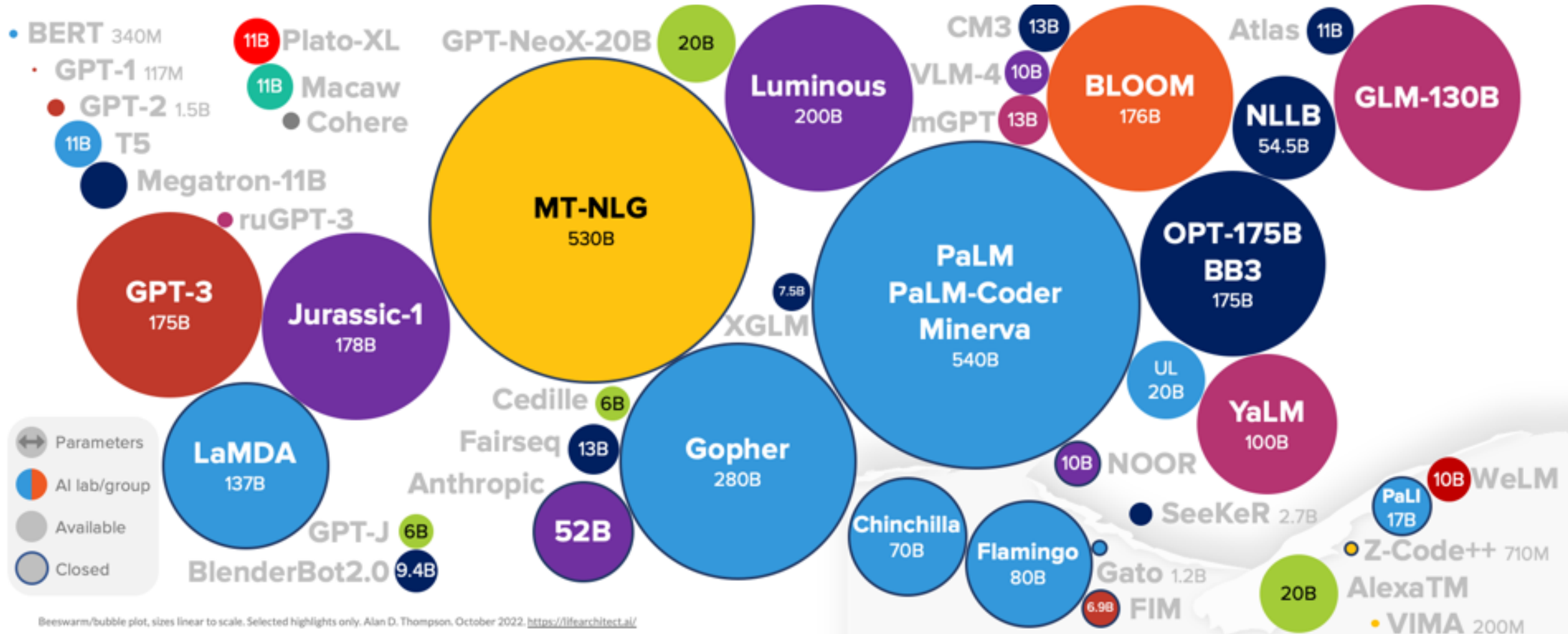
The Transformers Timeline



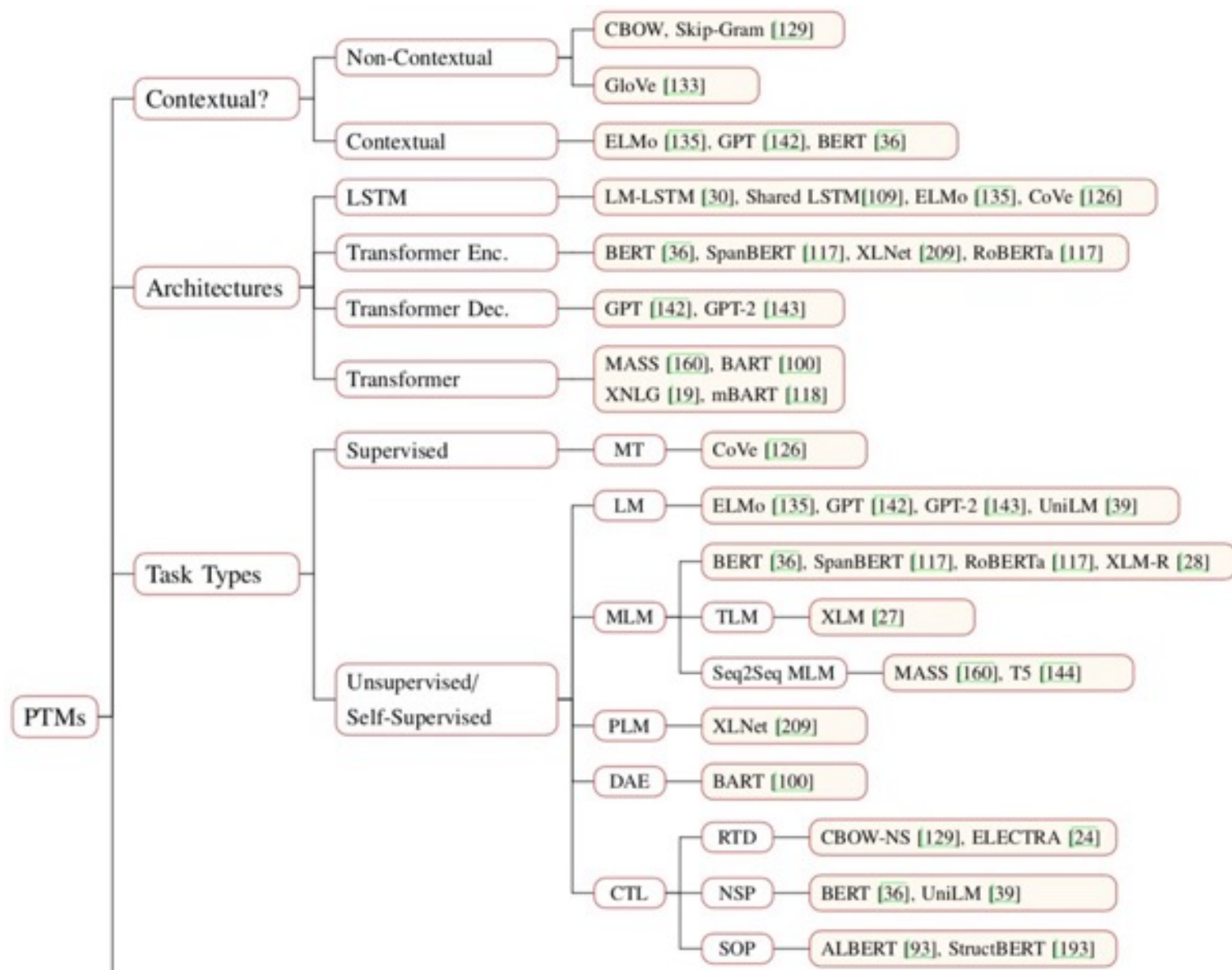
Transformer Models



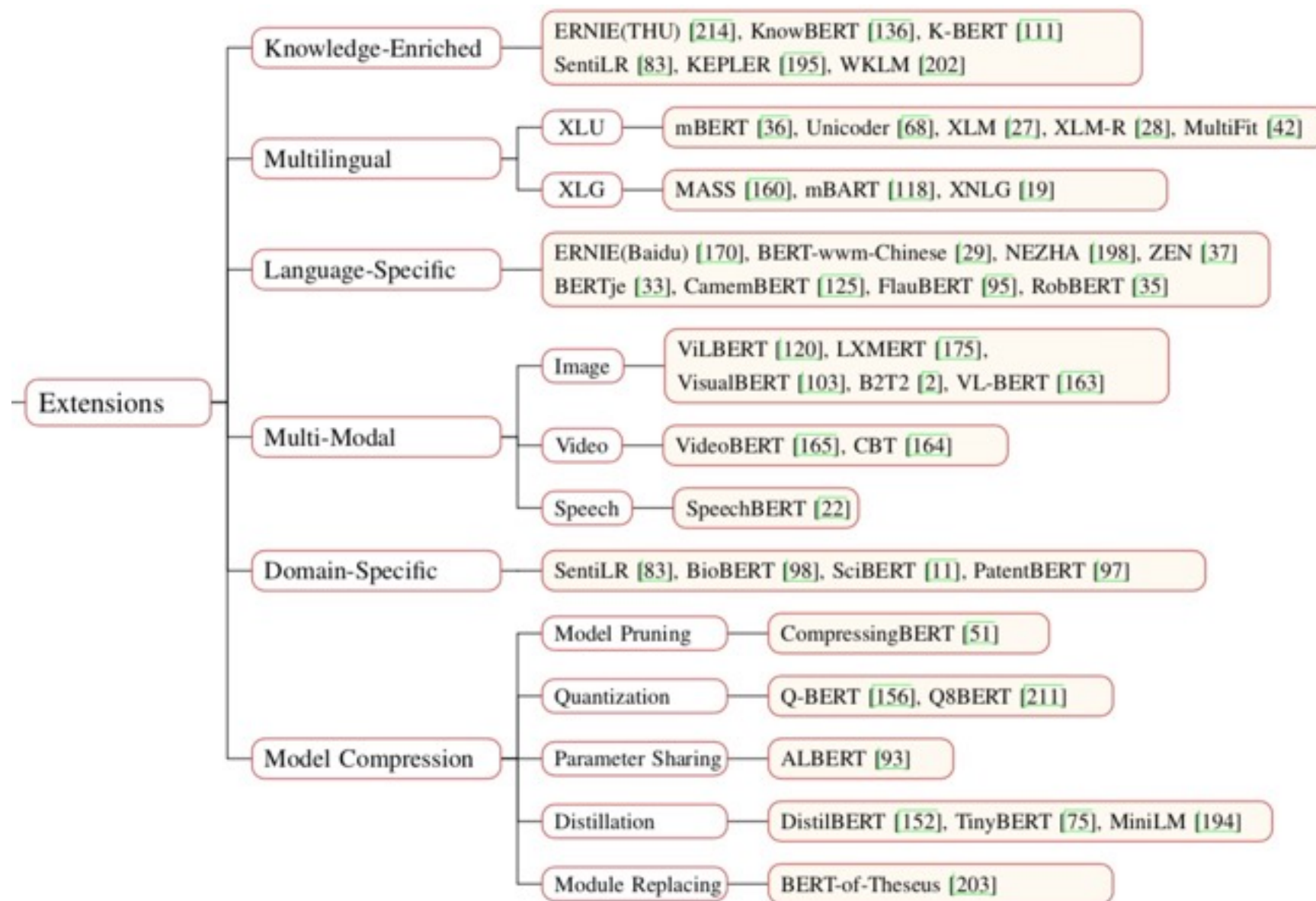
Language Models Sizes (GPT-3, PaLM, BLOOM)



Pre-trained Models (PTM)



Pre-trained Models (PTM)





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.

NLP Benchmark Datasets

Task	Dataset	Link
Machine Translation	WMT 2014 EN-DE WMT 2014 EN-FR	http://www-lium.univ-lemans.fr/~schwenk/cs1m_joint_paper/
Text Summarization	CNN/DM Newsroom DUC Gigaword	https://cs.nyu.edu/~kcho/DMQA/ https://summari.es/ https://www-nlpir.nist.gov/projects/duc/data.html https://catalog.ldc.upenn.edu/LDC2012T21
Reading Comprehension Question Answering Question Generation	ARC CliCR CNN/DM NewsQA RACE SQuAD Story Cloze Test NarrativeQA Quasar SearchQA	http://data.allenai.org/arc/ http://aclweb.org/anthology/N18-1140 https://cs.nyu.edu/~kcho/DMQA/ https://datasets.maluuba.com/NewsQA http://www.qizhexie.com/data/RACE_leaderboard https://rajpurkar.github.io/SQuAD-explorer/ http://aclweb.org/anthology/W17-0906.pdf https://github.com/deepmind/narrativeqa https://github.com/bdhingra/quasar https://github.com/nyu-dl/SearchQA
Semantic Parsing	AMR parsing ATIS (SQL Parsing) WikiSQL (SQL Parsing)	https://amr.isi.edu/index.html https://github.com/jkkummerfeld/text2sql-data/tree/master/data https://github.com/salesforce/WikiSQL
Sentiment Analysis	IMDB Reviews SST Yelp Reviews Subjectivity Dataset	http://ai.stanford.edu/~amaas/data/sentiment/ https://nlp.stanford.edu/sentiment/index.html https://www.yelp.com/dataset/challenge http://www.cs.cornell.edu/people/pabo/movie-review-data/
Text Classification	AG News DBpedia TREC 20 NewsGroup	http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html https://wiki.dbpedia.org/Datasets https://trec.nist.gov/data.html http://qwone.com/~jason/20Newsgroups/
Natural Language Inference	SNLI Corpus MultiNLI SciTail	https://nlp.stanford.edu/projects/snli/ https://www.nyu.edu/projects/bowman/multinli/ http://data.allenai.org/scitail/
Semantic Role Labeling	Proposition Bank OneNotes	http://propbank.github.io/ https://catalog.ldc.upenn.edu/LDC2013T19

Question Answering (QA) SQuAD

Stanford Question Answering Dataset

SQuAD

SQuAD

HomeExplore 2.0Explore 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
?	Retro-Reader (ensemble)	90.578	92.978

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, **grau-pel** 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?

grau-pel

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]

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.)

- 0. [Preface](#)
- 1. [Language Processing and Python](#)
- 2. [Accessing Text Corpora and Lexical Resources](#)
- 3. [Processing Raw Text](#)
- 4. [Writing Structured Programs](#)
- 5. [Categorizing and Tagging Words](#) (minor fixes still required)
- 6. [Learning to Classify Text](#)
- 7. [Extracting Information from Text](#)
- 8. [Analyzing Sentence Structure](#)
- 9. [Building Feature Based Grammars](#)
- 10. [Analyzing the Meaning of Sentences](#) (minor fixes still required)
- 11. [Managing Linguistic Data](#) (minor fixes still required)
- 12. [Afterword: Facing the Language Challenge](#)

[Bibliography](#)

[Term Index](#)

This book is made available under the terms of the [Creative Commons Attribution Noncommercial No-Derivative-Works 3.0 US License](#). 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

The image is a screenshot of the spaCy website's main banner. It has a blue background with a pattern of white line-art icons related to AI, linguistics, and technology. At the top left is the 'spaCy' logo. At the top right is a navigation menu with links: 'HOME', 'USAGE', 'API', 'DEMOS', 'BLOG', and a circular icon. The main heading is 'Industrial-Strength Natural Language Processing in Python'. Below this are three white rectangular boxes, each with a title and a paragraph of text.

spaCy

HOME USAGE API DEMOS BLOG

Industrial-Strength Natural Language Processing in Python

Fastest in the world

spaCy excels at large-scale information extraction tasks. It's written from the ground up in carefully memory-managed Cython. Independent research has confirmed that spaCy is the fastest in the world. If your application needs to process entire web dumps, spaCy is the library you want to be using.

Get things done

spaCy is designed to help you do real work — to build real products, or gather real insights. The library respects your time, and tries to avoid wasting it. It's easy to install, and its API is simple and productive. I like to think of spaCy as the Ruby on Rails of Natural Language Processing.

Deep learning

spaCy is the best way to prepare text for deep learning. It interoperates seamlessly with [TensorFlow](#), [Keras](#), [Scikit-Learn](#), [Gensim](#) and the rest of Python's awesome AI ecosystem. spaCy helps you connect the statistical models trained by these libraries to the rest of your application.

gensim

[Fork me on GitHub](#)



gensim

topic modelling for humans

[Download](#)
latest version from the Python Package Index

[Direct install with:
easy_install -U gensim](#)

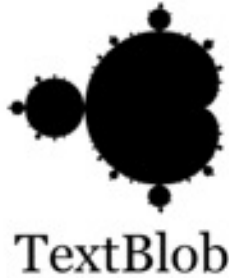
[Home](#) [Tutorials](#) [Install](#) [Support](#) [API](#) [About](#)


```
>>> from gensim import corpora, models, similarities
>>>
>>> # Load corpus iterator from a Matrix Market file on disk.
>>> corpus = corpora.MmCorpus('/path/to/corpus.mm')
>>>
>>> # Initialize Latent Semantic Indexing with 200 dimensions.
>>> lsi = models.LsiModel(corpus, num_topics=200)
>>>
>>> # Convert another corpus to the latent space and index it.
>>> index = similarities.MatrixSimilarity(lsi[another_corpus])
>>>
>>> # Compute similarity of a query vs. indexed documents
>>> sims = index[query]
```

Gensim is a FREE Python library

- ✓ Scalable statistical semantics
- ✓ Analyze plain-text documents for semantic structure
- ✓ Retrieve semantically similar documents

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.
'''

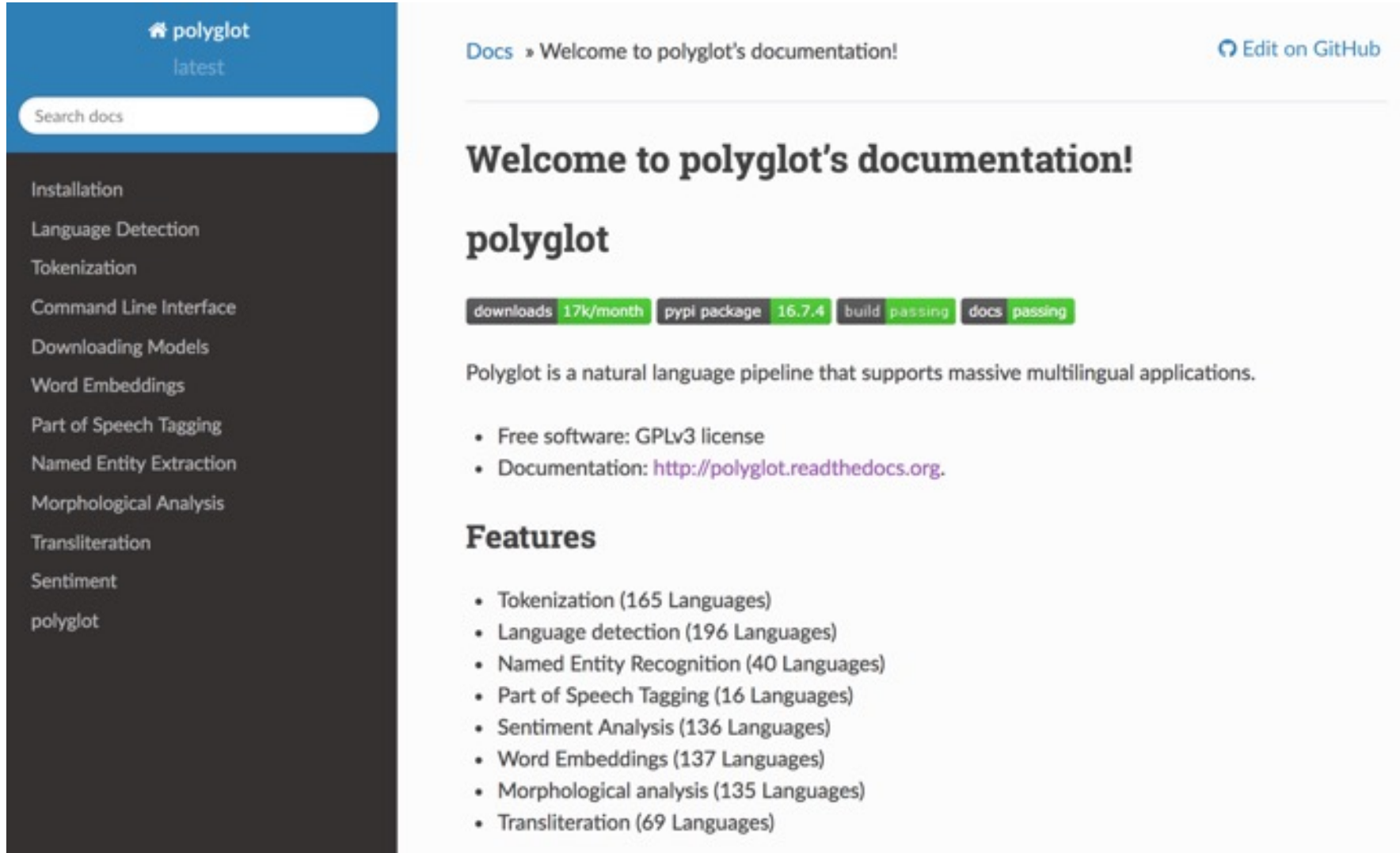
blob = TextBlob(text)
blob.tags          # [('The', 'DT'), ('titular', 'JJ'),
                    #  ('threat', 'NN'), ('of', 'IN'), ...]

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.

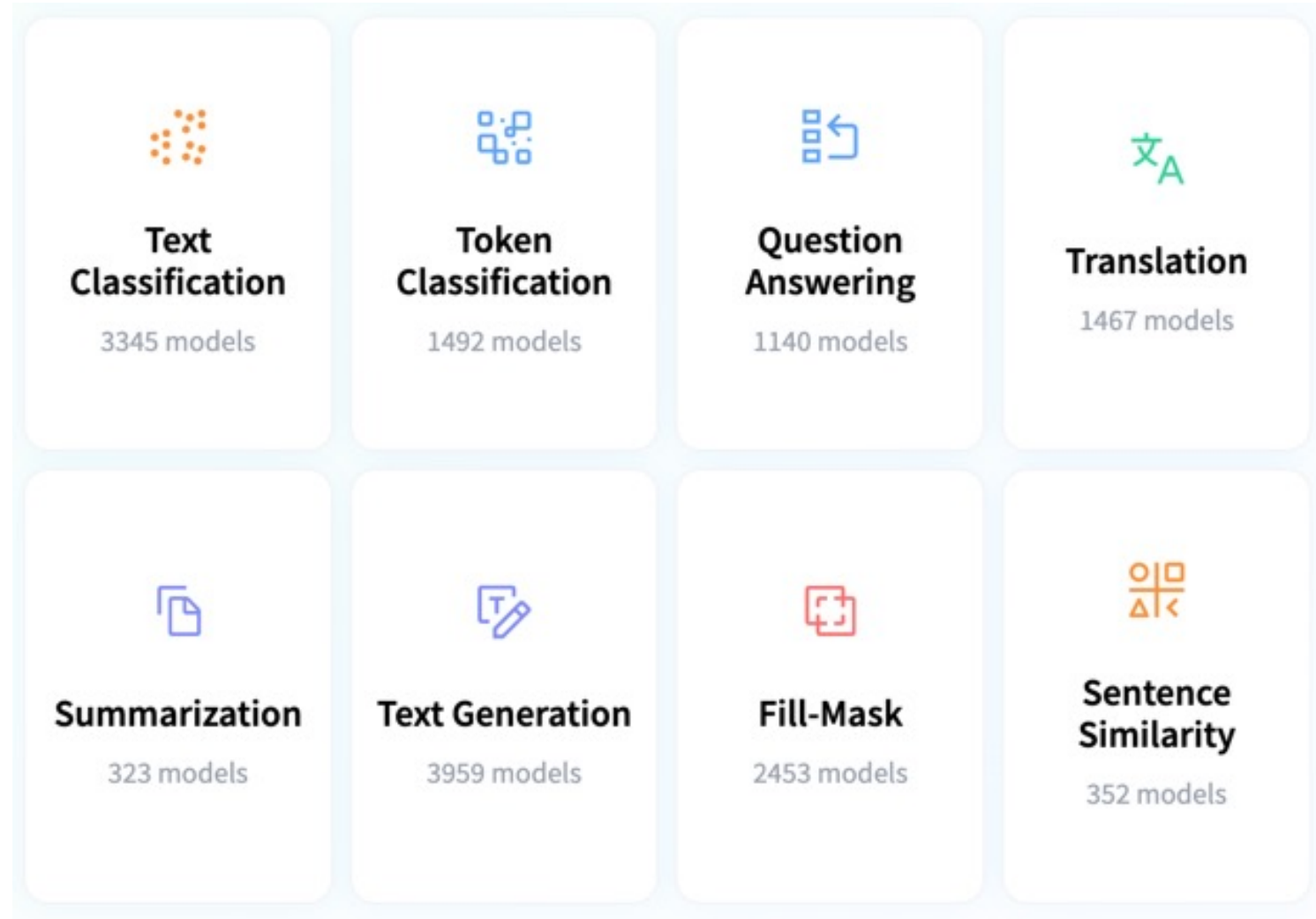
- Free software: GPLv3 license
- Documentation: <http://polyglot.readthedocs.org>.

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)

Hugging Face Tasks

Natural Language Processing



NLP with Transformers Github

The screenshot shows the GitHub repository page for 'nlp-with-transformers/notebooks'. The repository is public and has 170 forks and 1.1k stars. The main branch is 'main'. The repository contains several files and folders, including a README, a .gitignore, and several Jupyter notebooks. The repository is described as 'Jupyter notebooks for the Natural Language Processing with Transformers book'.

Repository Details:

- Repository: `nlp-with-transformers / notebooks` (Public)
- Notifications
- Fork: 170
- Star: 1.1k
- Code
- Issues
- Pull requests
- Actions
- Projects
- Wiki
- Security
- Insights

Branches and Tags:

- main
- 1 branch
- 0 tags

Files and Commits:

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

About:

- Jupyter notebooks for the Natural Language Processing with Transformers book
- transformersbook.com/
- Readme
- Apache-2.0 License
- 1.1k stars
- 33 watching
- 170 forks

Releases:

- No releases published

Packages:

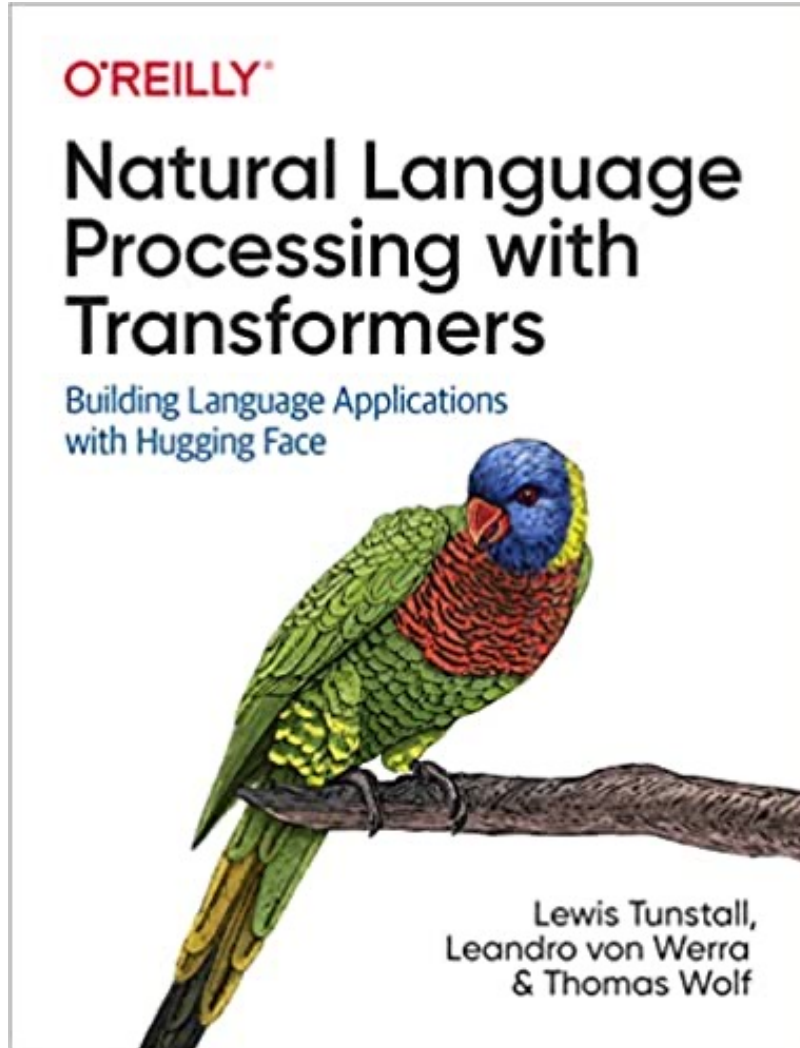
- No packages published

Book Cover:

O'REILLY
Natural Language Processing with Transformers
Building Language Applications with Hugging Face
Lewis Tunstall, Leandro von Werra & Thomas Wolf

<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!

Python in Google Colab (Python101)

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

NLP with Transformers

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

Github: <https://github.com/nlp-with-transformers/notebooks>

```
[1] 1 !git clone https://github.com/nlp-with-transformers/notebooks.git
    2 %cd notebooks
    3 from install import *
    4 install_requirements()
```

```
[3] 1 from utils import *
    2 setup_chapter()
```

```
[12] 1 text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
    2 from your online store in Germany. Unfortunately, when I opened the package, \
    3 I discovered to my horror that I had been sent an action figure of Megatron \
    4 instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
    5 dilemma. To resolve the issue, I demand an exchange of Megatron for the \
    6 Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
    7 this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
```

Text Classification

```
[13] 1 from transformers import pipeline
    2 classifier = pipeline("text-classification")
```

```
[14] 1 import pandas as pd
    2 outputs = classifier(text)
    3 pd.DataFrame(outputs)
```

<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

Table of contents

Text Classification with Transformers

The Dataset

From Datasets to DataFrames

From Text to Tokens

Character Tokenization

Word Tokenization

Subword Tokenization

Tokenizing the Whole Dataset

Training a Text Classifier

Transformers as Feature Extractors

Extracting the last hidden states

Creating a feature matrix

Visualizing the training set

Training a simple classifier

Fine-Tuning Transformers

Loading a pretrained model

Defining the performance metrics

Training the model

Sidebar: Fine-Tuning with Keras

Error analysis

Saving and sharing the model

Text Classification

Text Classification with Transformers

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

[10]

1 `!nvidia-smi`

[11]

1 `# Uncomment and run this cell if you're on Colab or Kaggle`

2 `!git clone https://github.com/nlp-with-transformers/notebooks.git`

3 `%cd notebooks`

4 `from install import *`

5 `install_requirements()`

[12]

1 `# hide`

2 `from utils import *`

3 `setup_chapter()`

The Dataset

[13]

1 `from datasets import list_datasets`

2 `all_datasets = list_datasets()`

3 `print(f"There are {len(all_datasets)} datasets currently available on the Hub")`

4 `print(f"The first 10 are: {all_datasets[:10]}")`

There are 3783 datasets currently available on the Hub

The first 10 are: ['acronym_identification', 'ade_corpus_v2', 'adversarial_qa',

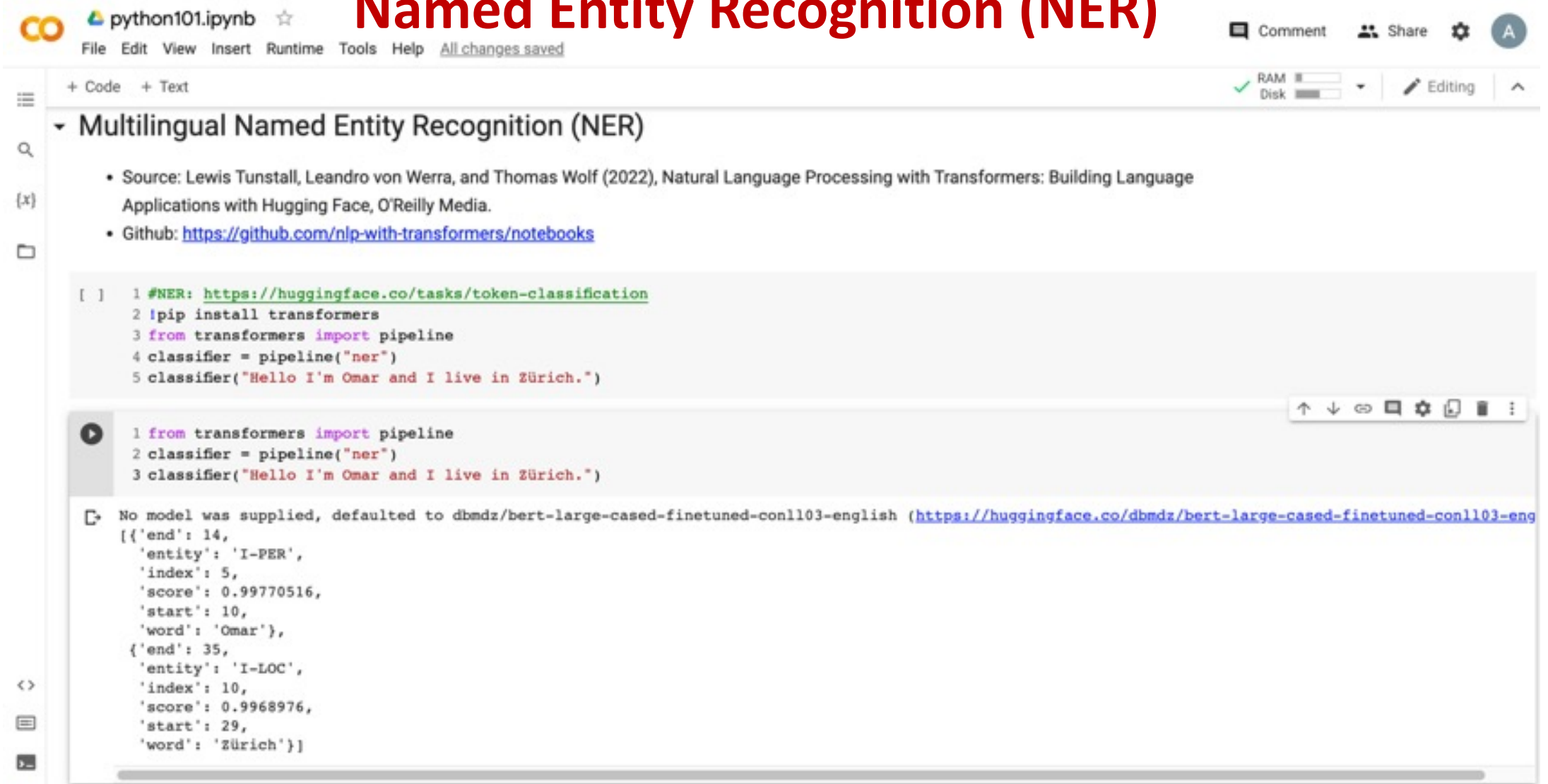
<https://tinyurl.com/aintpupython101>

175

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

+ 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>

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

```
1 from transformers import pipeline
2 classifier = pipeline("ner")
3 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...

Comment Share Settings User

+ 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>

```
1 #Source: https://huggingface.co/tasks/summarization
2 !pip install transformers
3 from transformers import pipeline
4 classifier = pipeline("summarization")
5 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 than 105 km²."
6 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'}]]

```
1 #!pip install transformers
2 text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
3 from your online store in Germany. Unfortunately, when I opened the package, \
4 I discovered to my horror that I had been sent an action figure of Megatron \
5 instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
6 dilemma. To resolve the issue, I demand an exchange of Megatron for the \
7 Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
8 this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
9 from transformers import pipeline
10 summarizer = pipeline("summarization")
11 outputs = summarizer(text, max_length=45, clean_up_tokenization_spaces=True)
12 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

- 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] 1 #Source: https://huggingface.co/tasks/text-generation
    2 #!pip install transformers
    3 from transformers import pipeline
    4 generator = pipeline('text-generation', model = 'gpt2')
    5 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"}]
```

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

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.

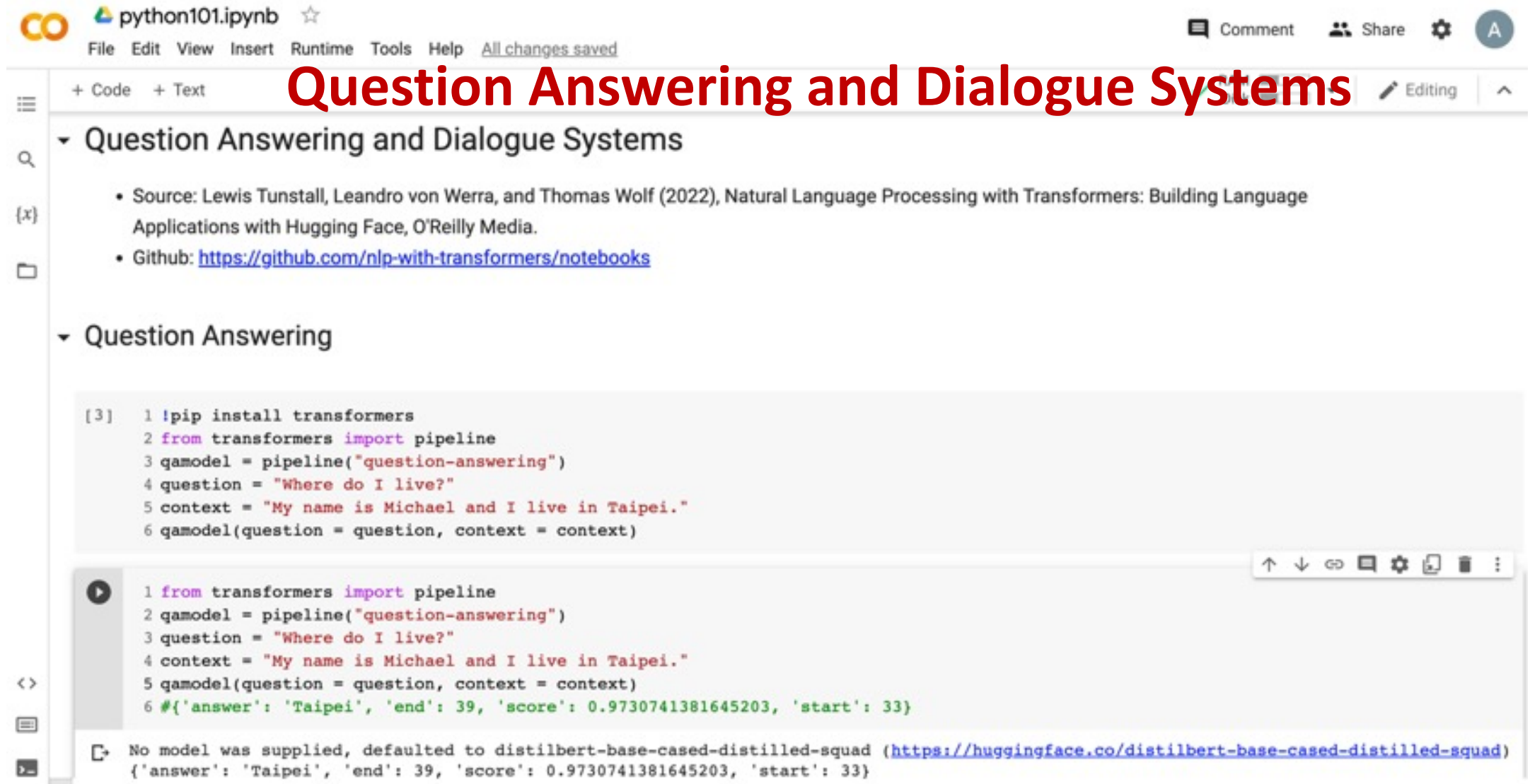
Once upon a time, every person who ever saw Jesus, knew that He was Christ. And even though he might not have known Him, He was

```
[1] 1 from transformers import pipeline
```

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

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



The screenshot shows a Google Colab notebook interface. At the top, the title bar says 'python101.ipynb' with a star icon. Below it is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help', followed by the text 'All changes saved'. On the right side of the title bar are icons for 'Comment', 'Share', 'Settings', and a user profile icon labeled 'A'. The notebook content is divided into two sections. The first section is titled 'Question Answering and Dialogue Systems' and contains a bulleted list: 'Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.' and 'Github: <https://github.com/nlp-with-transformers/notebooks>'. The second section is titled 'Question Answering' and contains two code blocks. The first code block is a cell with the following code:

```
[3] 1 !pip install transformers
2 from transformers import pipeline
3 qamodel = pipeline("question-answering")
4 question = "Where do I live?"
5 context = "My name is Michael and I live in Taipei."
6 qamodel(question = question, context = context)
```

 The second code block is a cell with the following code:

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

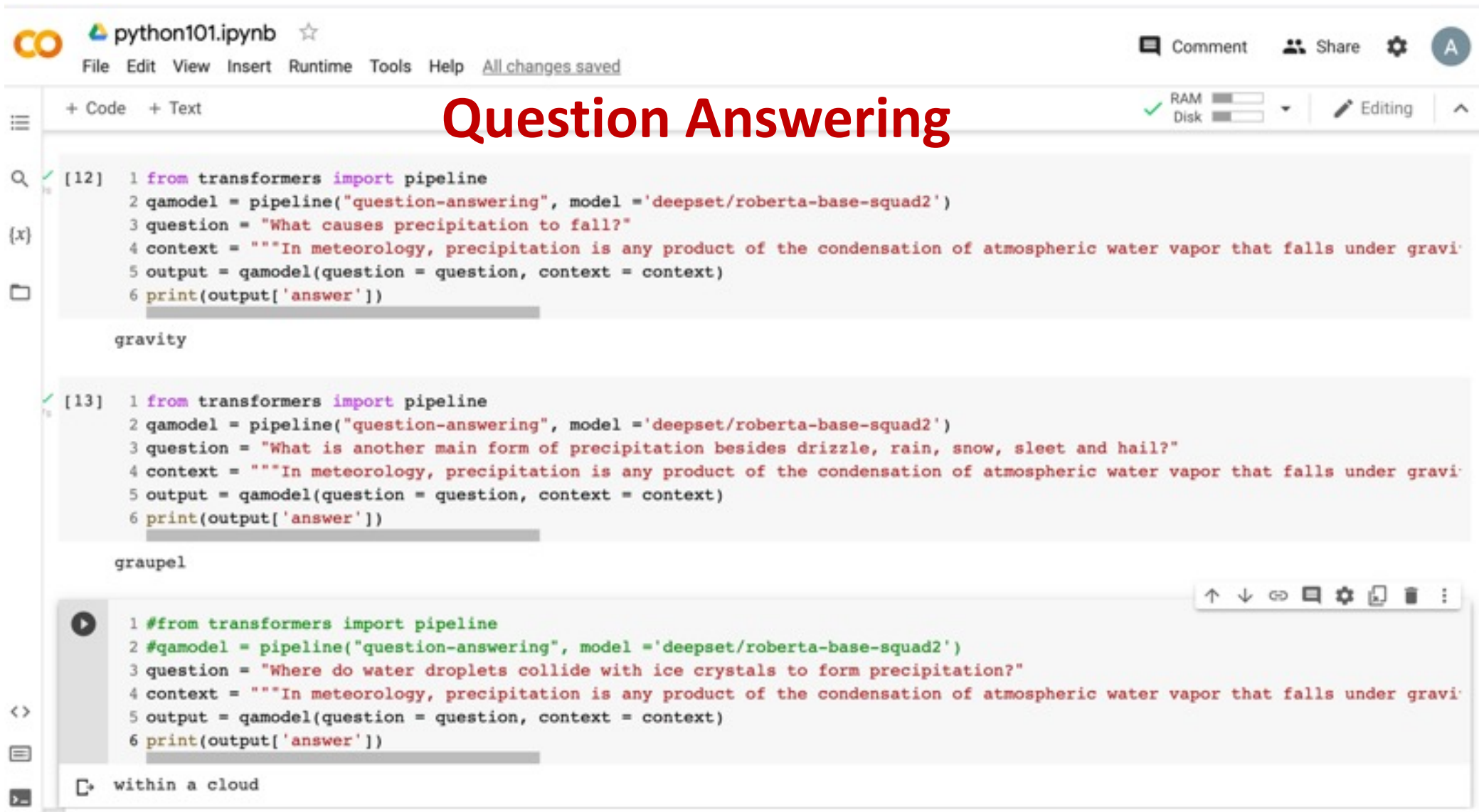
 Below the code blocks, there is a message: 'No model was supplied, defaulted to distilbert-base-cased-distilled-squad (<https://huggingface.co/distilbert-base-cased-distilled-squad>)' followed by the output:

```
{'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>



The screenshot shows a Google Colab notebook interface. At the top, the title bar says 'python101.ipynb' with a star icon. Below it is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. To the right of the menu bar are icons for 'Comment', 'Share', and a settings gear. The main area of the notebook contains three code cells. The first two cells are executed, showing their outputs. The third cell is not executed, showing a play button icon. The code in each cell uses the 'transformers' library to create a question-answering pipeline with a 'roberta-base-squad2' model. The context for all questions is a paragraph about meteorology and precipitation. The first cell asks 'What causes precipitation to fall?' and the output is 'gravity'. The second cell asks 'What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?' and the output is 'graupel'. The third cell asks 'Where do water droplets collide with ice crystals to form precipitation?'.

Question Answering

```
[12] 1 from transformers import pipeline
      2 qamodel = pipeline("question-answering", model='deepset/roberta-base-squad2')
      3 question = "What causes precipitation to fall?"
      4 context = """In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravi
      5 output = qamodel(question = question, context = context)
      6 print(output['answer'])

gravity
```

```
[13] 1 from transformers import pipeline
      2 qamodel = pipeline("question-answering", model='deepset/roberta-base-squad2')
      3 question = "What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?"
      4 context = """In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravi
      5 output = qamodel(question = question, context = context)
      6 print(output['answer'])

graupel
```

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

within a cloud

<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

RAM Disk

Editing

Table of contents

- Semantic Analysis
 - Named Entity Recognition (NER)
 - NER with CRF
 - NER with CRF RandomizedSearchCV
 - Sentiment Analysis
 - Sentiment Analysis - Unsupervised Lexical
 - Sentiment Analysis - Supervised Machine Learning
 - Sentiment Analysis - Supervised Deep Learning Models
 - Sentiment Analysis - Advanced Deep Learning
 - Deep Learning and Universal Sentence-Embedding Models
 - Universal Sentence Encoder (USE)
 - Universal Sentence Encoder Multilingual (USEM)
 - Question Answering and Dialogue Systems**
 - Question Answering (QA)
 - BERT for Question Answering**
 - Dialogue Systems
 - Joint Intent Classification and Slot Filling with Transformers
 - Data Visualization
 - Section

+ Code + Text

Question Answering and Dialogue Systems

Question Answering (QA)

BERT for Question Answering

Source: Apoorv Nandan (2020), BERT (from HuggingFace Transformers) for Text Extraction, https://keras.io/examples/nlp/text_extraction_with_bert/

Description: Fine tune pretrained BERT from HuggingFace Transformers on SQuAD.

Introduction

This demonstration uses SQuAD (Stanford Question-Answering Dataset). In SQuAD, an input consists of a question, and a paragraph for context. The goal is to find the span of text in the paragraph that answers the question. We evaluate our performance on this data with the "Exact Match" metric, which measures the percentage of predictions that exactly match any one of the ground-truth answers.

We fine-tune a BERT model to perform this task as follows:

1. Feed the context and the question as inputs to BERT.
2. Take two vectors S and T with dimensions equal to that of hidden states in BERT.
3. Compute the probability of each token being the start and end of the answer span. The probability of a token being the start of the answer is given by a dot product between S and the representation of the token in the last layer of BERT, followed by a softmax over all tokens. The probability of a token being the end of the answer is computed similarly with the vector T.
4. Fine-tune BERT and learn S and T along the way.

References:

- [BERT](#)
- [SQuAD](#)

<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 Settings A

RAM Disk Editing

Table of contents

- RandomizedSearchCV
- Sentiment Analysis
 - Sentiment Analysis - Unsupervised Lexical
 - Sentiment Analysis - Supervised Machine Learning
 - Sentiment Analysis - Supervised Deep Learning Models
 - Sentiment Analysis - Advanced Deep Learning
- Deep Learning and Universal Sentence-Embedding Models
 - Universal Sentence Encoder (USE)
 - Universal Sentence Encoder Multilingual (USEM)
- Question Answering and Dialogue Systems
 - Question Answering (QA)
 - BERT for Question Answering**
 - Dialogue Systems
 - Joint Intent Classification and Slot Filling with Transformers
- Data Visualization
- Section

+ Code + Text

Downloading: 100% 433/433 [00:29<00:00, 14.5B/s]

Downloading: 100% 536M/536M [00:29<00:00, 18.3MB/s]

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 384)]	0	
input_3 (InputLayer)	[(None, 384)]	0	
input_2 (InputLayer)	[(None, 384)]	0	
tf_bert_model (TFBertModel)	[(None, 384, 768), (109482240		input_1[0][0]
start_logits (Dense)	(None, 384, 1)	768	tf_bert_model[0][0]
end_logits (Dense)	(None, 384, 1)	768	tf_bert_model[0][0]
flatten (Flatten)	(None, 384)	0	start_logits[0][0]
flatten_1 (Flatten)	(None, 384)	0	end_logits[0][0]
activation_7 (Activation)	(None, 384)	0	flatten[0][0]
activation_8 (Activation)	(None, 384)	0	flatten_1[0][0]

Total params: 109,483,776
Trainable params: 109,483,776
Non-trainable params: 0

CPU times: user 20.8 s, sys: 7.75 s, total: 28.5 s
Wall time: 1min 42s

<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

RAM Disk

Editing

Dialogue Systems

```
[ ] 1 #Source: Olivier Grisel (2020), Transformers (BERT fine-tuning): Joint Intent Classification and S
    2 #https://github.com/m2dsupsdicclass/lectures-labs/blob/master/labs/06_deep_nlp/Transformers_Joint_I
```

Joint Intent Classification and Slot Filling with Transformers

The goal of this notebook is to fine-tune a pretrained transformer-based neural network model to convert a user query expressed in English into a representation that is structured enough to be processed by an automated service.

Here is an example of interpretation computed by such a Natural Language Understanding system:

```
>>> nlu("Book a table for two at Le Ritz for Friday night",
        tokenizer, joint_model, intent_names, slot_names)
```

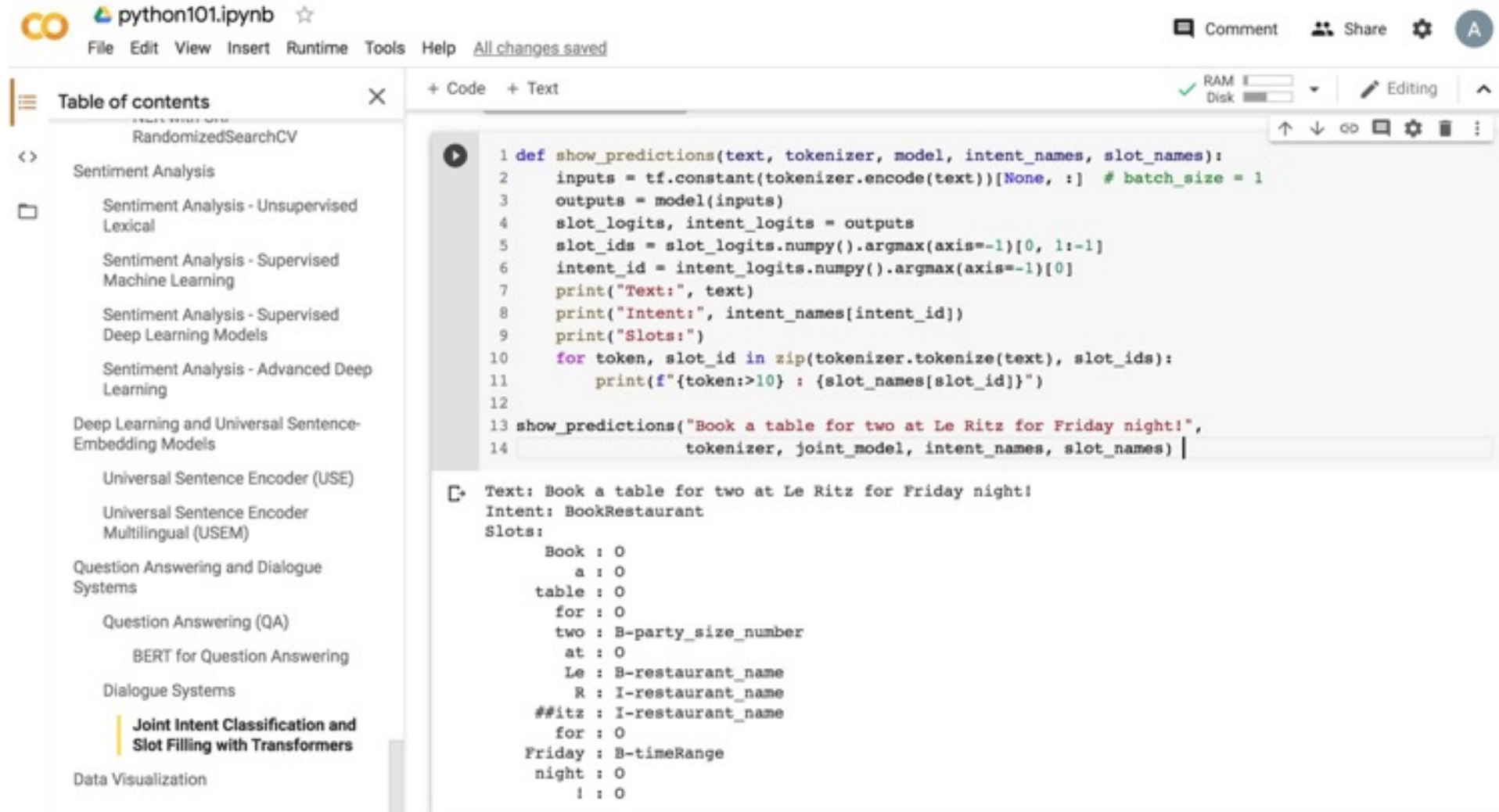
```
{
  'intent': 'BookRestaurant',
  'slots': {
    'party_size_number': 'two',
    'restaurant_name': 'Le Ritz',
    'timeRange': 'Friday night'
  }
}
```

Intent classification is a simple sequence classification problem. The trick is to treat the structured knowledge extraction part ("Slot Filling") as token-level classification problem using BIO-annotations:

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

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



The screenshot displays a Google Colab notebook interface. The top bar shows the notebook title 'python101.ipynb' and standard Colab controls like 'Comment', 'Share', and 'RAM/Disk' usage. The left sidebar contains a 'Table of contents' with a tree view of the notebook's sections. The main editor area shows a Python function `show_predictions` that takes text, a tokenizer, a model, intent names, and slot names as input. The function encodes the text, runs it through the model, and prints the predicted intent and slots. Below the code, the output of the function is displayed for the input text 'Book a table for two at Le Ritz for Friday night!'. The output shows the intent as 'BookRestaurant' and a list of slots with their corresponding values or labels.

```
1 def show_predictions(text, tokenizer, model, intent_names, slot_names):
2     inputs = tf.constant(tokenizer.encode(text))[None, :] # batch_size = 1
3     outputs = model(inputs)
4     slot_logits, intent_logits = outputs
5     slot_ids = slot_logits.numpy().argmax(axis=-1)[0, 1:-1]
6     intent_id = intent_logits.numpy().argmax(axis=-1)[0]
7     print("Text:", text)
8     print("Intent:", intent_names[intent_id])
9     print("Slots:")
10    for token, slot_id in zip(tokenizer.tokenize(text), slot_ids):
11        print(f"{token:>10} : {slot_names[slot_id]}")
12
13 show_predictions("Book a table for two at Le Ritz for Friday night!",
14                 tokenizer, joint_model, intent_names, slot_names)
```

Text: Book a table for two at Le Ritz for Friday night!
Intent: BookRestaurant
Slots:
Book : 0
a : 0
table : 0
for : 0
two : B-party_size_number
at : 0
Le : B-restaurant_name
R : I-restaurant_name
##itz : I-restaurant_name
for : 0
Friday : B-timeRange
night : 0
! : 0

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

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

The screenshot displays a Google Colab notebook environment. The top bar includes the Colab logo, the notebook title 'python101.ipynb', and a star icon. Below this is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help', followed by a status indicator 'All changes saved'. On the right side of the top bar are icons for 'Comment', 'Share', 'Settings', and a user profile icon.

The left sidebar contains a 'Table of contents' panel with a search icon and a list of topics: 'NER with CRF', 'RandomizedSearchCV', 'Sentiment Analysis' (with sub-items like 'Unsupervised Lexical', 'Supervised Machine Learning', 'Supervised Deep Learning Models', and 'Advanced Deep Learning'), 'Deep Learning and Universal Sentence-Embedding Models' (with 'USE' and 'USEM'), 'Question Answering and Dialogue Systems' (with 'QA' and 'BERT for QA'), 'Dialogue Systems', 'Joint Intent Classification and Slot Filling with Transformers' (highlighted with a yellow bar), and 'Data Visualization'.

The main area is a code editor showing Python code. The code is divided into two sections. The first section (lines 19-34) implements a naive slot handling logic for Named Entity Recognition (NER). It iterates through words, identifying new slots and appending words to the active slot. The second section (lines 35-46) defines an NLU function that takes text, a tokenizer, a model, intent names, and slot names as input. It uses TensorFlow to encode the text and a model to generate logits for slots and intents. The function returns a dictionary with the intent and a dictionary of slot names and values.

```
19 # Naive slot handling: treat B- and I- the same...
20 new_slot_name = current_word_slot_name[2:]
21 if active_slot_name is None:
22     active_slot_words.append(word)
23     active_slot_name = new_slot_name
24 elif new_slot_name == active_slot_name:
25     active_slot_words.append(word)
26 else:
27     collected_slots[active_slot_name] = " ".join(active_slot_words)
28     active_slot_words = [word]
29     active_slot_name = new_slot_name
30 if active_slot_name:
31     collected_slots[active_slot_name] = " ".join(active_slot_words)
32 info["slots"] = collected_slots
33 return info
34
35 def nlu(text, tokenizer, model, intent_names, slot_names):
36     inputs = tf.constant(tokenizer.encode(text))[None, :] # batch_size = 1
37     outputs = model(inputs)
38     slot_logits, intent_logits = outputs
39     slot_ids = slot_logits.numpy().argmax(axis=-1)[0, 1:-1]
40     intent_id = intent_logits.numpy().argmax(axis=-1)[0]
41
42     return decode_predictions(text, tokenizer, intent_names, slot_names,
43                               intent_id, slot_ids)
44
45 nlu("Book a table for two at Le Ritz for Friday night",
46     tokenizer, joint_model, intent_names, slot_names)
```

Below the code editor, a JSON object is displayed, representing the output of the NLU function:

```
{ 'intent': 'BookRestaurant',
  'slots': { 'party_size_number': 'two',
             'restaurant_name': 'Le Ritz',
             'timeRange': 'Friday night' } }
```

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

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

The screenshot shows a Google Colab notebook interface. The top bar includes the Colab logo, the notebook title 'python101.ipynb', and a star icon. Below this is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. On the right, there are icons for 'Comment', 'Share', and a user profile. A status bar shows 'RAM' and 'Disk' usage, and a 'Editing' mode indicator.

The left sidebar contains a 'Table of contents' panel with a search icon. It lists various topics: 'Machine Learning with scikit-learn', 'Classification and Prediction', 'Support Vector Machine (SVM)', 'Random Forest', 'K-Means Clustering', 'Deep Learning', 'Image Classification', 'Text Classification: IMDB Movie Review' (highlighted with a yellow bar), 'Deep Learning for Financial Time Series Forecasting', 'Portfolio Optimization and Algorithmic Trading', 'Investment Portfolio Optimisation with Python', 'Efficient Frontier Portfolio Optimisation in Python', 'Investment Portfolio Optimization', 'Text Analytics and Natural Language Processing (NLP)', 'Python for Natural Language Processing', and 'spaCy Chinese Model'.

The main content area is titled 'Text Classification: IMDB Movie Review'. It includes a source link: https://www.tensorflow.org/tutorials/keras/text_classification_with_hub. Below this, there are three code blocks:

```
[1] 1 !pip install -q tensorflow-hub
     2 !pip install -q tensorflow-datasets
```

```
[2] 1 import os
     2 import numpy as np
     3
     4 import tensorflow as tf
     5 import tensorflow_hub as hub
     6 import tensorflow_datasets as tfds
     7
     8 print("Version: ", tf.__version__)
     9 print("Eager mode: ", tf.executing_eagerly())
    10 print("Hub version: ", hub.__version__)
    11 print("GPU is", "available" if tf.config.list_physical_devices("GPU") else "NOT AVAILABLE")
```

Version: 2.4.1
Eager mode: True
Hub version: 0.12.0
GPU is available

```
[3] 1 # Split the training set into 60% and 40% to end up with 15,000 examples
     2 # for training, 10,000 examples for validation and 25,000 examples for testing.
     3 train_data, validation_data, test_data = tfds.load(
     4     name="imdb_reviews",
```

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

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

The screenshot shows a Google Colab notebook interface. The top bar includes the Colab logo, the notebook title 'python101.ipynb', and a star icon. Below this is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status 'All changes saved'. On the right, there are icons for 'Comment', 'Share', a settings gear, and a user profile 'A'.

On the left, a 'Table of contents' sidebar is open, listing various topics: Machine Learning with scikit-learn, Classification and Prediction, Support Vector Machine (SVM), Random Forest, K-Means Clustering, Deep Learning, Image Classification, **Text Classification: IMDB Movie Review** (highlighted), Deep Learning for Financial Time Series Forecasting, Portfolio Optimization and Algorithmic Trading, Investment Portfolio Optimisation with Python, Efficient Frontier Portfolio Optimisation in Python, Investment Portfolio Optimization, Text Analytics and Natural Language Processing (NLP), Python for Natural Language Processing, and spaCy Chinese Model.

The main area shows code execution. A tooltip for 'Huggingface Transformers' points to <https://github.com/huggingface/transformers>. The code cells are as follows:

```
[18] 1 !pip install transformers
```

```
1 from transformers import pipeline
2 classifier = pipeline('sentiment-analysis')
3 classifier('We are very happy to introduce pipeline to the transformers repository.')
```

Below the code, four download progress bars are shown, all at 100%:

- Downloading: 100% 629/629 [00:00<00:00, 1.31kB/s]
- Downloading: 100% 268M/268M [00:05<00:00, 46.9MB/s]
- Downloading: 100% 232k/232k [00:01<00:00, 159kB/s]
- Downloading: 100% 48.0/48.0 [00:00<00:00, 522B/s]

```
[11] 1 classifier('This movie is very good.')
```

```
[12] 1 classifier('This movie is very boring.')
```

The output for the first classification is:

```
{'label': 'POSITIVE', 'score': 0.9996980428695679}}
```

The output for the second classification is:

```
{'label': 'POSITIVE', 'score': 0.9998621940612793}}
```

The output for the third classification is:

```
{'label': 'NEGATIVE', 'score': 0.999795138835907}}
```

<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 Settings

RAM Disk

Editing

Table of contents

- Mail Customer Segmentation
- Machine Learning with scikit-learn
 - Classification and Prediction
 - Support Vector Machine (SVM)
 - Random Forest
 - K-Means Clustering
 - Deep Learning
 - Image Classification
 - Text Classification: IMDB Movie Review**
 - Deep Learning for Financial Time Series Forecasting
 - Portfolio Optimization and Algorithmic Trading
 - Investment Portfolio Optimisation with Python
 - Efficient Frontier Portfolio Optimisation in Python
 - Investment Portfolio Optimization
 - Text Analytics and Natural Language Processing (NLP)

```
1 # from transformers import pipeline
2 question_answerer = pipeline('question-answering')
3 question_answerer({'question': 'What is the name of the repository ?',
4                     'context': 'Pipeline has been included in the huggingface/transformers repository'})
5
```

```
{'answer': 'huggingface/transformers',
 'end': 58,
 'score': 0.309702068567276,
 'start': 34}
```

```
[24] 1 ontext = '''In meteorology, precipitation is any product of the condensation of atmospheric water vapor
2 ontext
```

'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".'

```
[25] 1 question_answerer({'question': 'Where do water droplets collide with ice crystals to form precipitation?'
2                       'context': context})
```

```
{'answer': 'within a cloud',
 'end': 321,
 'score': 0.5175967812538147,
 'start': 307}
```

```
[28] 1 question_answerer({'question': 'What causes precipitation to fall?',
2                       'context': context})
```

<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 Settings A

RAM Disk Editing

Table of contents

- Text Clustering
- Semantic Analysis and Named Entity Recognition (NER)
 - Semantic Analysis
 - Named Entity Recognition (NER)
- Sentiment Analysis
 - Sentiment Analysis - Unsupervised Lexical
 - Sentiment Analysis - Supervised Machine Learning
 - Sentiment Analysis - Supervised Deep Learning Models
 - Sentiment Analysis - Advanced Deep Learning
- Deep Learning and Universal Sentence-Embedding Models
 - Universal Sentence Encoder (USE)**
 - Universal Sentence Encoder Multilingual (USEM)
- Data Visualization
- Section

Deep Learning and Universal Sentence-Embedding Models

Universal Sentence Encoder (USE)

- Source: Universal Sentence Encoder: <https://tfhub.dev/google/universal-sentence-encoder/4>

```
[ ] 1 import tensorflow as tf
    2 import tensorflow_hub as hub
    3 import numpy as np
    4 import pandas as pd
    5 import os
    6 import re
    7 import matplotlib.pyplot as plt
    8 import seaborn as sns
    9
   10 module_url = "https://tfhub.dev/google/universal-sentence-encoder/4"
   11 # "https://tfhub.dev/google/universal-sentence-encoder-large/5"
   12 model = hub.load(module_url)
   13 print ("module %s loaded" % module_url)
   14 def embed(input):
   15     return model(input)
```

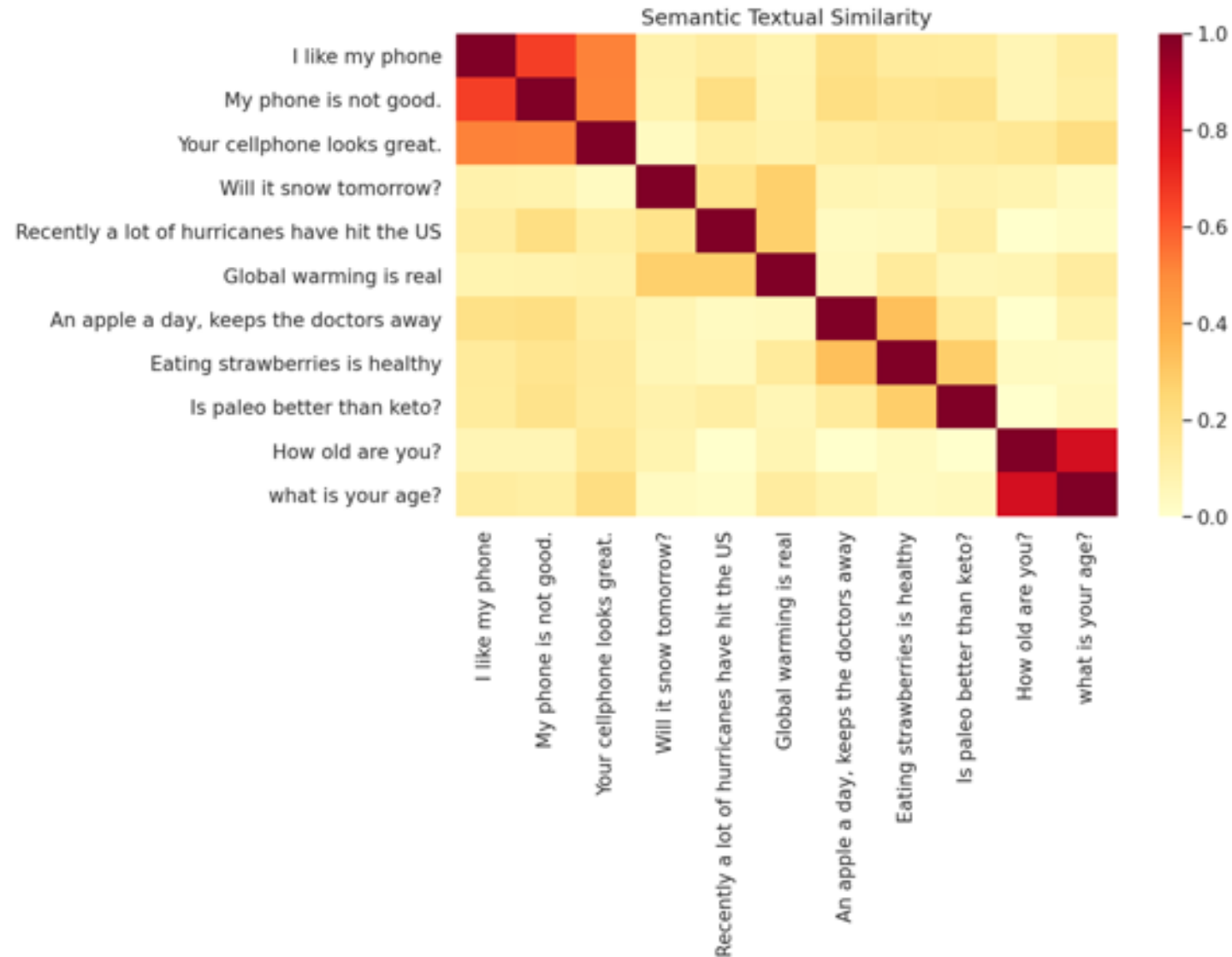
module <https://tfhub.dev/google/universal-sentence-encoder/4> loaded

```
[ ] 1 word = "Elephant"
    2 sentence = "I am a sentence for which I would like to get its embedding."
```

<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>

The screenshot displays a Google Colab notebook titled "python101.ipynb". The left sidebar contains a "Table of contents" with various NLP topics. The main area shows a code cell with the following Python code:

```
1 text = "Steve Jobs and Steve Wozniak incorporated Apple Computer on January 3, 1977, in Cupertino, California."
2 doc = nlp(text)
3 displacy.render(doc, style="ent", jupyter=True)
```

Below the code, the visual NLP analysis is shown, highlighting entities in the sentence: "Steve Jobs PERSON and Steve Wozniak PERSON incorporated Apple Computer ORG on January 3, 1977 DATE, in Cupertino GPE, California GPE."

The second code cell contains the following Python code:

```
[ ] 1 import spacy
2 nlp = spacy.load("en_core_web_sm")
3 doc = nlp("Stanford University is located in California. It is a great university.")
4 import pandas as pd
5 cols = ("text", "lemma", "pos", "tag", "pos_explain", "stopword")
6 rows = []
7 for t in doc:
8     row = [t.text, t.lemma_, t.pos_, t.tag_, spacy.explain(t.pos_), t.is_stop]
9     rows.append(row)
10 df = pd.DataFrame(rows, columns=cols)
11 df
```

The output of the second code cell is a pandas DataFrame table:

	text	lemma	pos	tag	pos_explain	stopword
0	Stanford	Stanford	PROPN	NNP	proper noun	False
1	University	University	PROPN	NNP	proper noun	False
2	is	be	VERB	VBZ	verb	True
3	located	locate	VERB	VRN	verb	False
4	in	in	ADP	IN	adposition	True
5	California	California	PROPN	NNP	proper noun	False
6	.	.	PUNCT	.	punctuation	False
7	It	-PRON-	PRON	PRP	pronoun	True

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

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

The screenshot shows a Google Colab notebook interface. The top bar includes the Colab logo, the notebook title 'python101.ipynb', and a star icon. Below this is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. A status bar indicates 'All changes saved'. On the right side of the top bar are icons for 'Comment', 'Share', 'Settings', and a user profile icon 'A'.

The left sidebar contains a 'Table of contents' panel. It lists the following items:

- Text Analytics and Natural Language Processing (NLP)
 - Python for Natural Language Processing
 - spaCy Chinese Model
 - Open Chinese Convert (OpenCC, 開放中文轉換)
 - Jieba 結巴中文分詞
 - Natural Language Toolkit (NLTK)
 - Stanza: A Python NLP Library for Many Human Languages
- Text Processing and Understanding
 - NLTK (Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit)
 - NLP Zero to Hero
 - Natural Language Processing - Tokenization (NLP Zero to Hero, part 1)
 - Natural Language Processing - Sequencing - Turning sentence into data (NLP Zero to Hero, part 2)
 - Natural Language Processing - Training a model to recognize sentiment in text (NLP Zero to Hero, part 3)

The main content area shows a code cell with the following text:

```
+ Code + Text
```

Text Analytics and Natural Language Processing (NLP)

Python for Natural Language Processing

spaCy

- spaCy: Industrial-Strength Natural Language Processing in Python
- Source: <https://spacy.io/usage/spacy-101>

```
[1] 1 !python -m spacy download en_core_web_sm
```

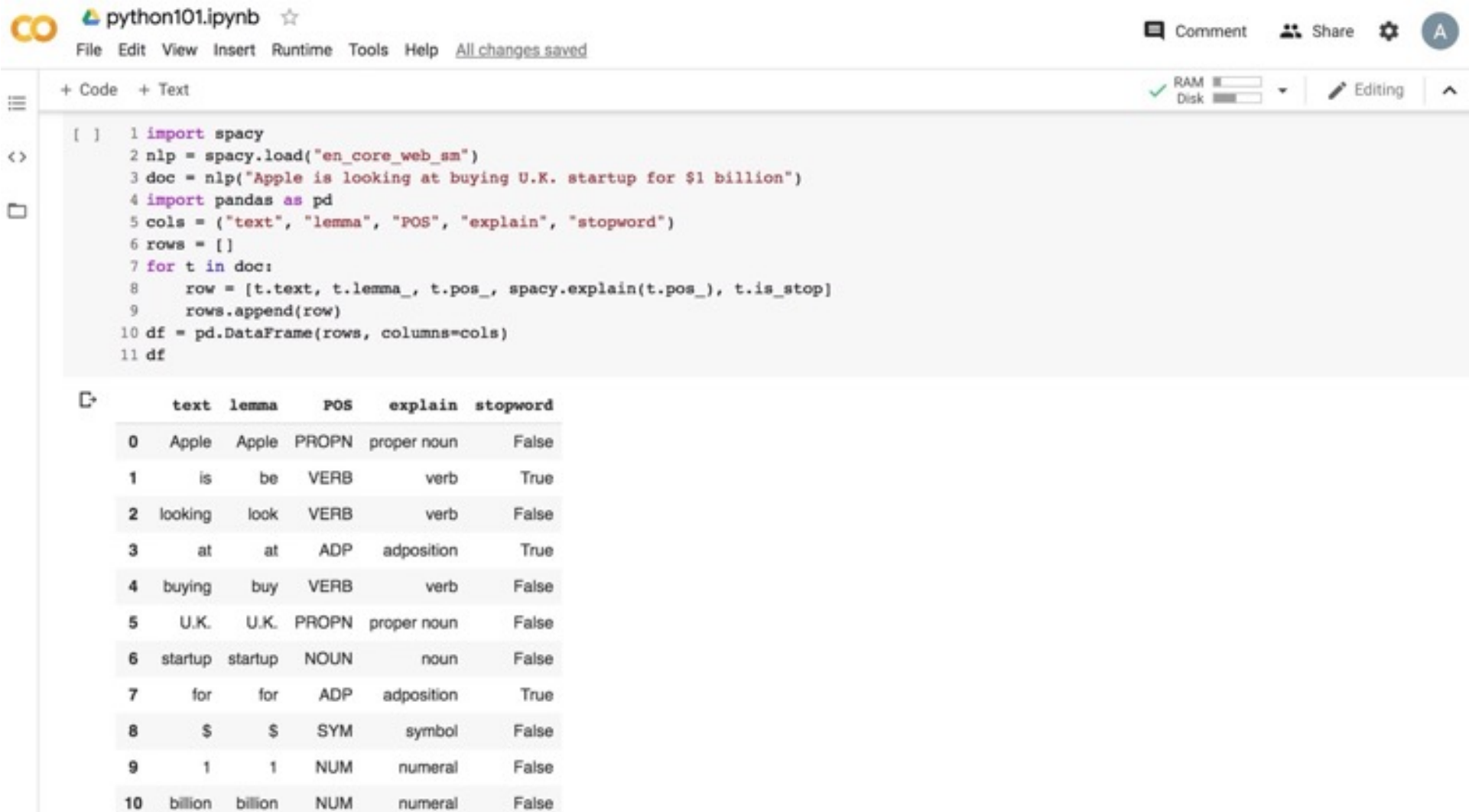
```
[3] 1 import spacy
2 nlp = spacy.load("en_core_web_sm")
3 doc = nlp("Apple is looking at buying U.K. startup for $1 billion")
4 for token in doc:
5     print(token.text, token.pos_, token.dep_)
```

Apple PROPn nsubj
is AUX aux
looking VERB ROOT
at ADP prep
buying VERB pcomp
U.K. PROPn compound
startup NOUN dobj
for ADP prep
\$ SYM quantmod
1 NUM compound
billion NUM pobj

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

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



The screenshot shows a Google Colab notebook interface. At the top, the title bar says "python101.ipynb" with a star icon. Below it is a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". To the right of the menu bar are icons for "Comment", "Share", and a settings gear. Below the menu bar is a toolbar with "+ Code" and "+ Text" buttons. To the right of the toolbar are indicators for "RAM" and "Disk" usage, and an "Editing" button. The main area of the notebook contains a Python script. Below the script, a table displays the results of the script's execution.

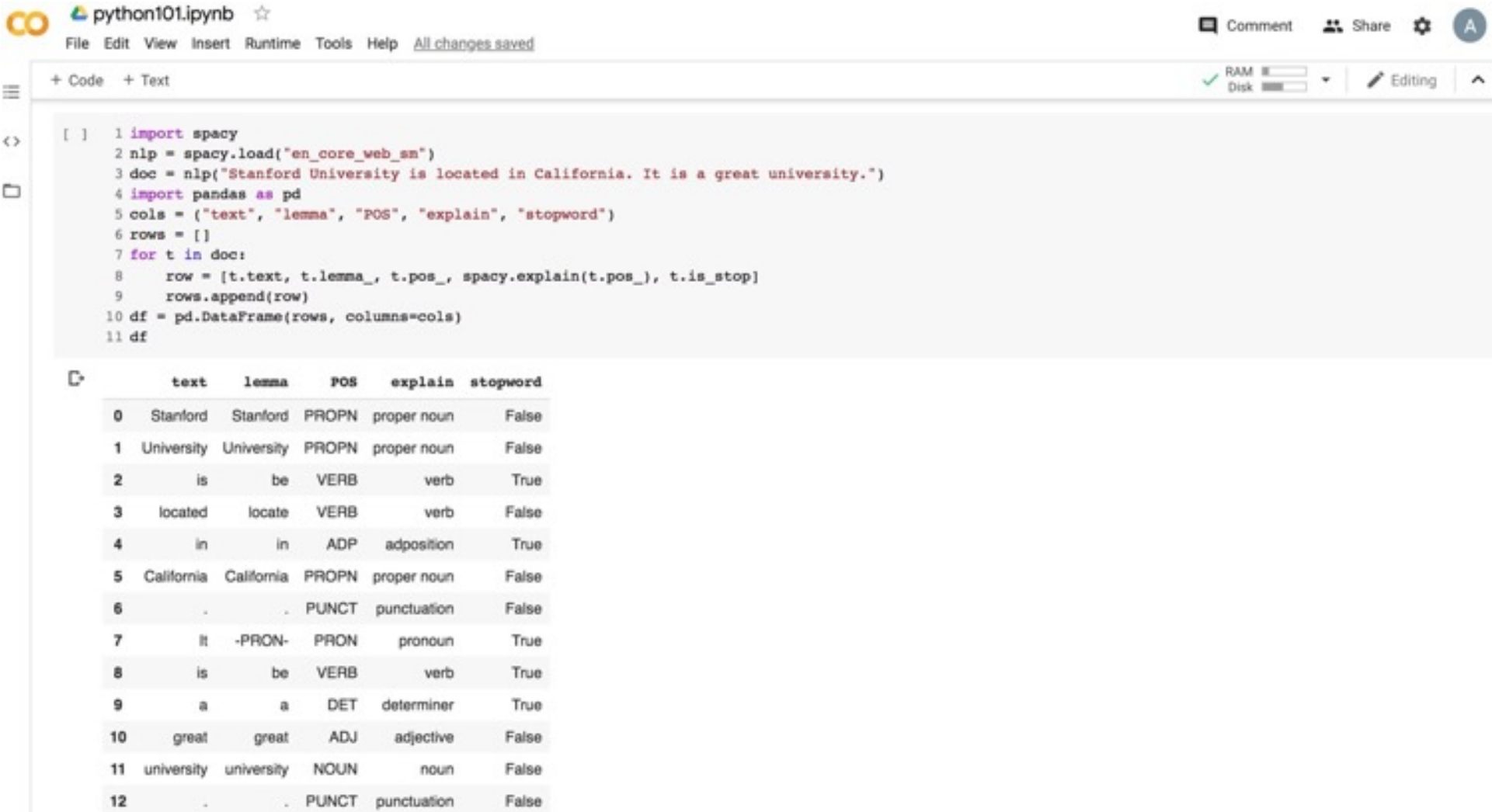
```
[ ] 1 import spacy
    2 nlp = spacy.load("en_core_web_sm")
    3 doc = nlp("Apple is looking at buying U.K. startup for $1 billion")
    4 import pandas as pd
    5 cols = ("text", "lemma", "POS", "explain", "stopword")
    6 rows = []
    7 for t in doc:
    8     row = [t.text, t.lemma_, t.pos_, spacy.explain(t.pos_), t.is_stop]
    9     rows.append(row)
   10 df = pd.DataFrame(rows, columns=cols)
   11 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>



The screenshot shows a Google Colab notebook interface. At the top, the title bar says 'python101.ipynb' with a star icon. Below it is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help', followed by a status 'All changes saved'. On the right side of the top bar are icons for 'Comment', 'Share', a settings gear, and a user profile 'A'. Below the menu bar is a toolbar with '+ Code' and '+ Text' buttons, and on the right, a green checkmark, 'RAM' and 'Disk' usage indicators, and an 'Editing' button. The main area contains a code cell with the following Python code:

```
[ ] 1 import spacy
2 nlp = spacy.load("en_core_web_sm")
3 doc = nlp("Stanford University is located in California. It is a great university.")
4 import pandas as pd
5 cols = ("text", "lemma", "POS", "explain", "stopword")
6 rows = []
7 for t in doc:
8     row = [t.text, t.lemma_, t.pos_, spacy.explain(t.pos_), t.is_stop]
9     rows.append(row)
10 df = pd.DataFrame(rows, columns=cols)
11 df
```

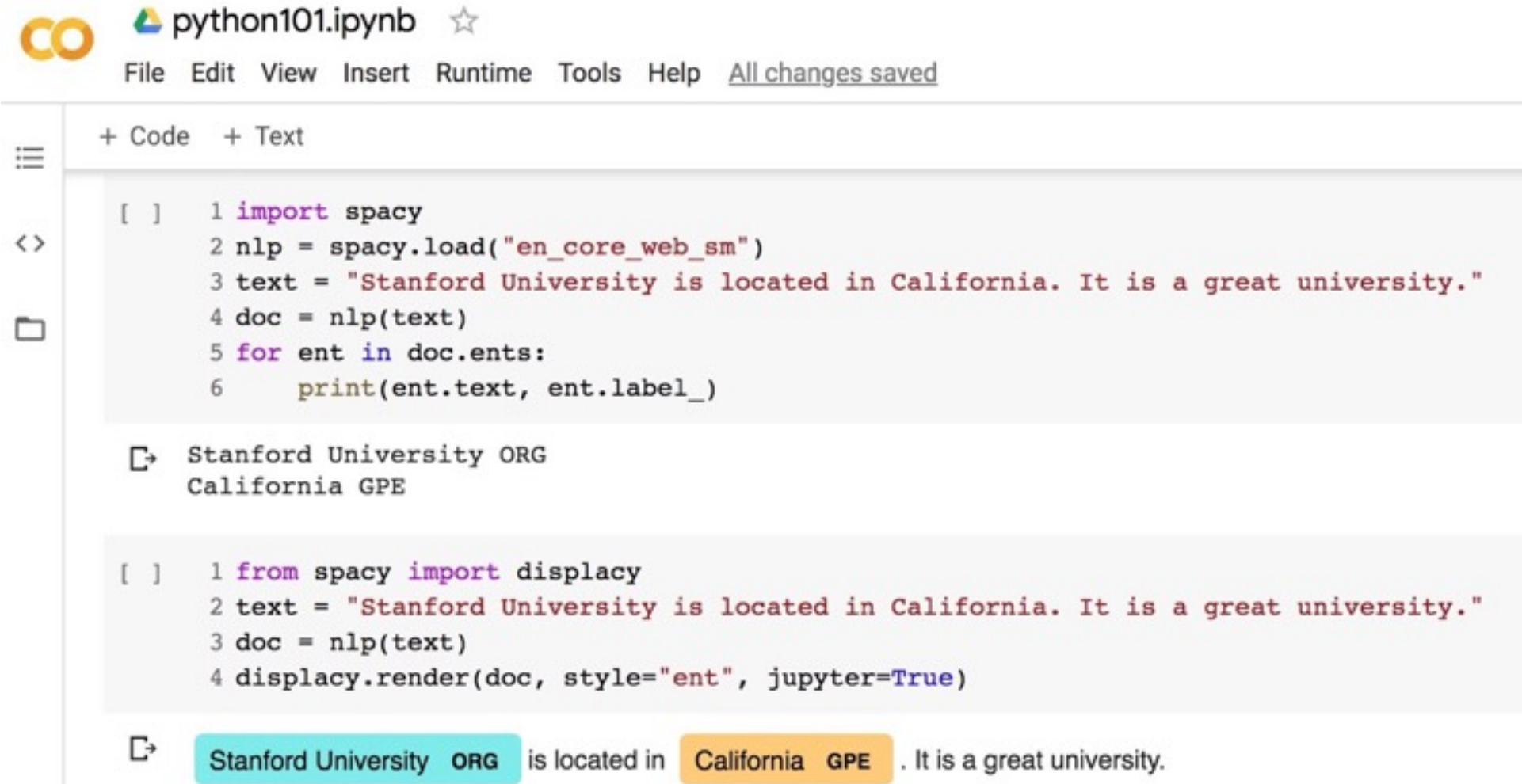
Below the code cell, the output is displayed as a table with 6 columns: 'text', 'lemma', 'POS', 'explain', and 'stopword'. The table contains 13 rows of data, indexed from 0 to 12.

	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>



The image shows a Google Colab notebook titled "python101.ipynb". The interface includes a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help", along with a status "All changes saved". On the left, there are icons for a menu, code execution, and a file explorer. The notebook contains two code cells. The first cell imports spaCy, loads the "en_core_web_sm" model, processes a text string about Stanford University, and prints the named entities. The output shows "Stanford University" as an ORG and "California" as a GPE. The second cell uses displacy to render the same text with the entities highlighted in colored boxes (cyan for ORG, orange for GPE).

```
[ ] 1 import spacy
    2 nlp = spacy.load("en_core_web_sm")
    3 text = "Stanford University is located in California. It is a great university."
    4 doc = nlp(text)
    5 for ent in doc.ents:
    6     print(ent.text, ent.label_)
```

☞ Stanford University ORG
California GPE

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

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

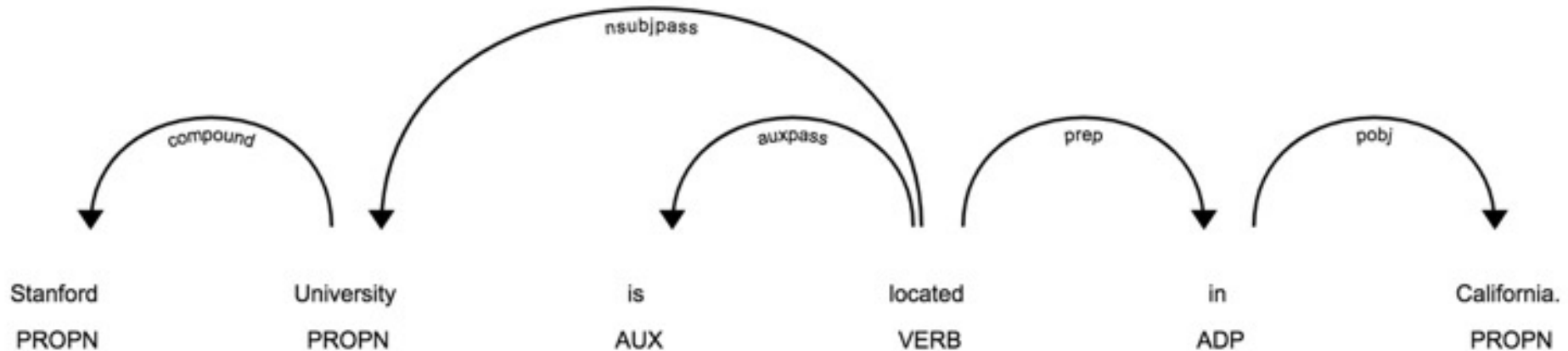
<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

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

```
1 from spacy import displacy
2 text = "Stanford University is located in California. It is a great university."
3 doc = nlp(text)
4 displacy.render(doc, style="ent", jupyter=True)
5 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>

The screenshot shows a Google Colab notebook titled "python101.ipynb". The left sidebar contains a "Table of contents" with links to various NLP topics. The main area displays a code cell with the following Python code:

```
1 text = "Steve Jobs and Steve Wozniak incorporated Apple Computer on January 3, 1977, in Cupertino, California."
2 doc = nlp(text)
3 displacy.render(doc, style="ent", jupyter=True)
```

Below the code, the output shows the sentence with entities highlighted: "Steve Jobs" (PERSON), "Steve Wozniak" (PERSON), "Apple Computer" (ORG), "January 3, 1977" (DATE), "Cupertino" (GPE), and "California" (GPE).

Below the output, there is a code cell with the following Python code:

```
[ ] 1 import spacy
2 nlp = spacy.load("en_core_web_sm")
3 doc = nlp("Stanford University is located in California. It is a great university.")
4 import pandas as pd
5 cols = ("text", "lemma", "pos", "tag", "pos_explain", "stopword")
6 rows = []
7 for t in doc:
8     row = [t.text, t.lemma_, t.pos_, t.tag_, spacy.explain(t.pos_), t.is_stop]
9     rows.append(row)
10 df = pd.DataFrame(rows, columns=cols)
11 df
```

The output of this code is a DataFrame with the following columns: text, lemma, pos, tag, pos_explain, and stopwords. The DataFrame contains 8 rows of token information:

	text	lemma	pos	tag	pos_explain	stopword
0	Stanford	Stanford	PROPN	NNP	proper noun	False
1	University	University	PROPN	NNP	proper noun	False
2	is	be	VERB	VBZ	verb	True
3	located	locate	VERB	VRN	verb	False
4	in	in	ADP	IN	adposition	True
5	California	California	PROPN	NNP	proper noun	False
6	.	.	PUNCT	.	punctuation	False
7	It	-PRON-	PRON	PRP	pronoun	True

<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>