

# 人工智慧

## (Artificial Intelligence)

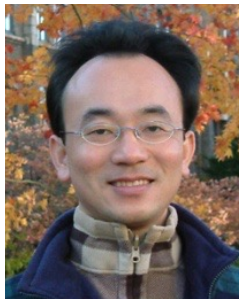
# 深度學習自然語言處理

## (Deep Learning for Natural Language Processing)

1092AI10

MBA, IM, NTPU (M5010) (Spring 2021)

Wed 2, 3, 4 (9:10-12:00) (B8F40)



Min-Yuh Day

戴敏育

Associate Professor

副教授

Institute of Information Management, National Taipei University

國立臺北大學 資訊管理研究所

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

2021-05-26



# 課程大綱 (Syllabus)

- | 週次 (Week) | 日期 (Date)  | 內容 (Subject/Topics)                                              |
|-----------|------------|------------------------------------------------------------------|
| 1         | 2021/02/24 | 人工智慧概論<br>(Introduction to Artificial Intelligence)              |
| 2         | 2021/03/03 | 人工智慧和智慧代理人<br>(Artificial Intelligence and Intelligent Agents)   |
| 3         | 2021/03/10 | 問題解決<br>(Problem Solving)                                        |
| 4         | 2021/03/17 | 知識推理和知識表達<br>(Knowledge, Reasoning and Knowledge Representation) |
| 5         | 2021/03/24 | 不確定知識和推理<br>(Uncertain Knowledge and Reasoning)                  |
| 6         | 2021/03/31 | 人工智慧個案研究 I<br>(Case Study on Artificial Intelligence I)          |

# 課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
7	2021/04/07	放假一天 (Day off)
8	2021/04/14	機器學習與監督式學習 (Machine Learning and Supervised Learning)
9	2021/04/21	期中報告 (Midterm Project Report)
10	2021/04/28	學習理論與綜合學習 (The Theory of Learning and Ensemble Learning)
11	2021/05/05	深度學習 (Deep Learning)
12	2021/05/12	人工智慧個案研究 II (Case Study on Artificial Intelligence II)

# 課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
13	2021/05/19	強化學習 (Reinforcement Learning)
14	2021/05/26	深度學習自然語言處理 (Deep Learning for Natural Language Processing)
15	2021/06/02	機器人技術 (Robotics)
16	2021/06/09	人工智慧哲學與倫理，人工智慧的未來 (Philosophy and Ethics of AI, The Future of AI)
17	2021/06/16	期末報告 I (Final Project Report I)
18	2021/06/23	期末報告 II (Final Project Report II)

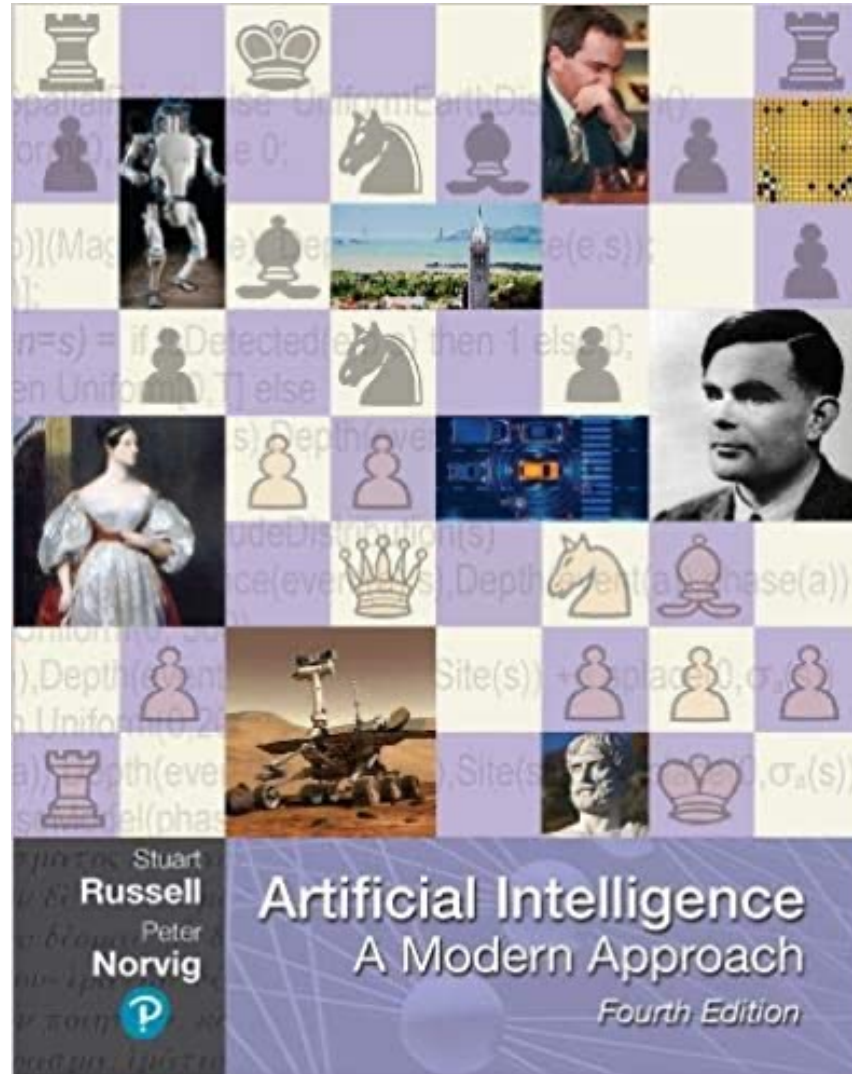


# **Deep Learning for Natural Language Processing**

# 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)



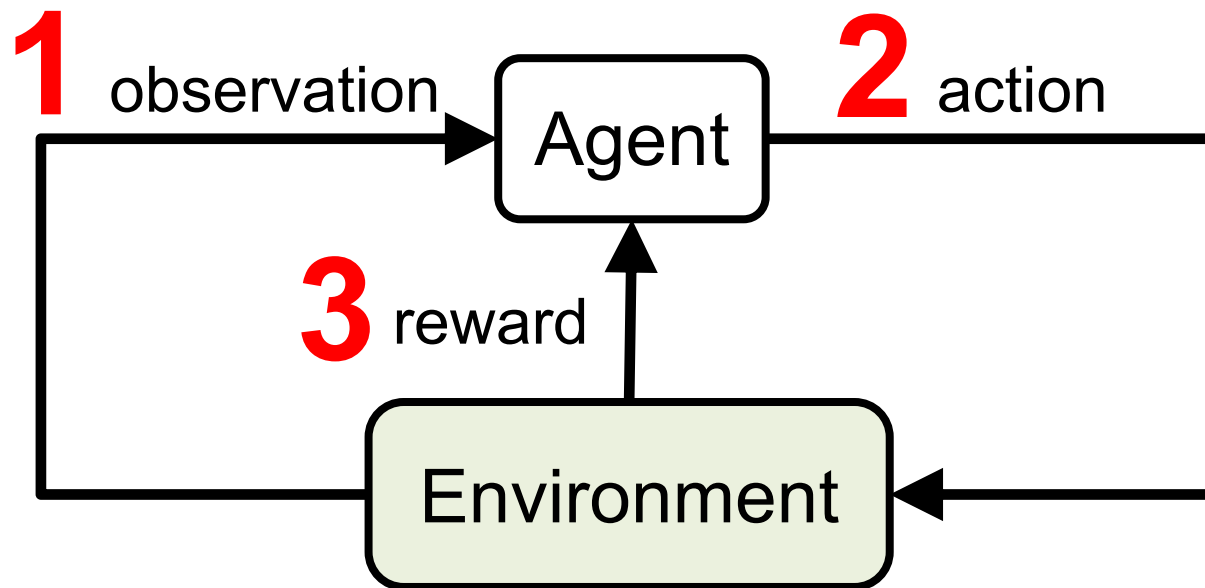
# Reinforcement Learning (DL)

The diagram illustrates the Reinforcement Learning loop. It consists of two main components: an Agent and an Environment. The Agent is represented by a white rounded rectangle with a black border, positioned above the Environment. The Environment is represented by a light green rounded rectangle with a black border, positioned below the Agent. The interaction between the Agent and the Environment is implied by their relative positions in the loop.

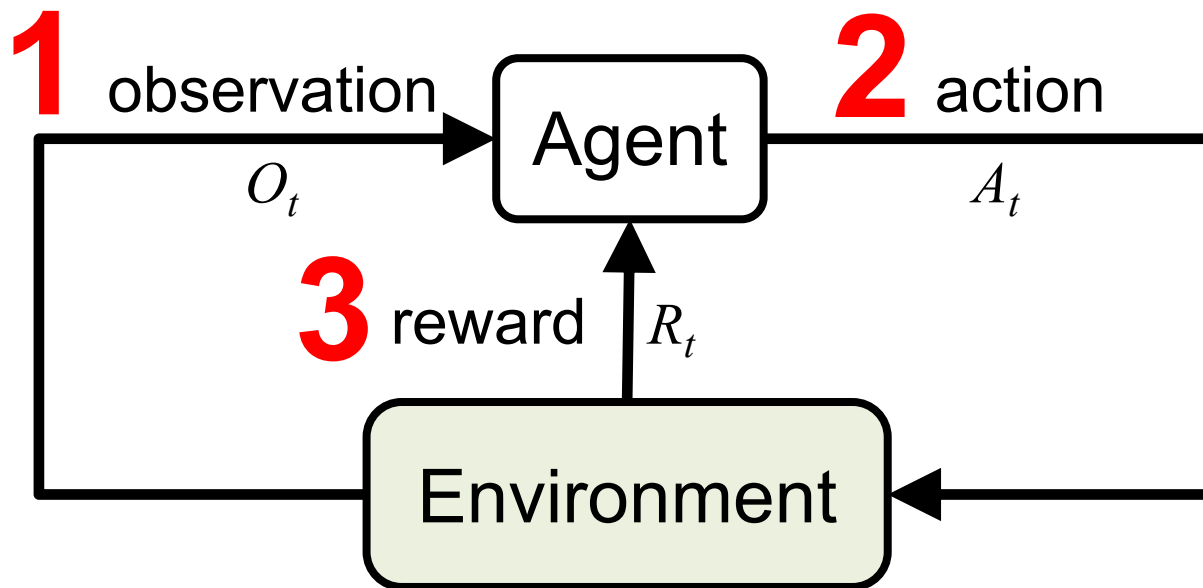
Agent

Environment

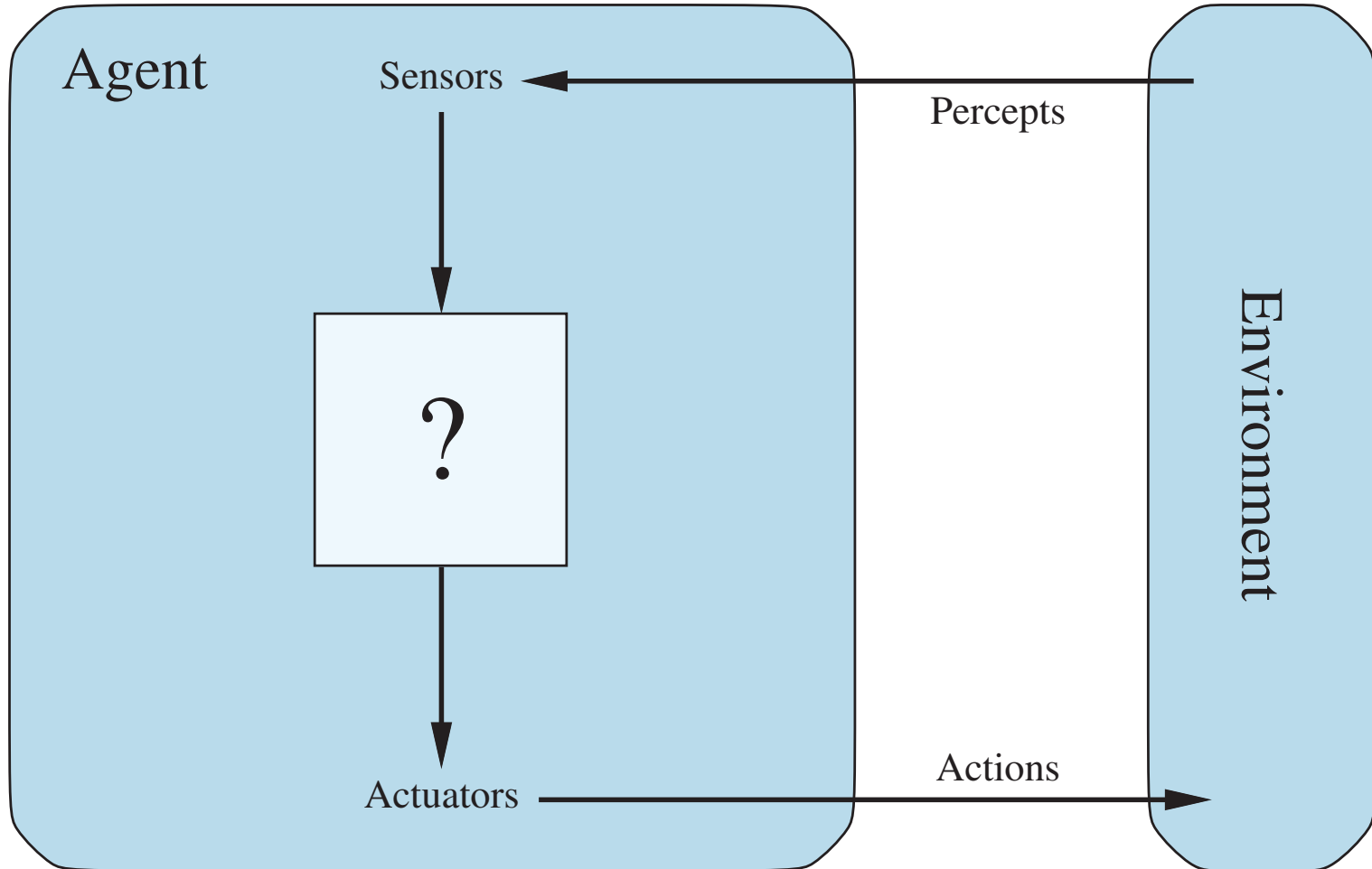
# Reinforcement Learning (DL)



# Reinforcement Learning (DL)

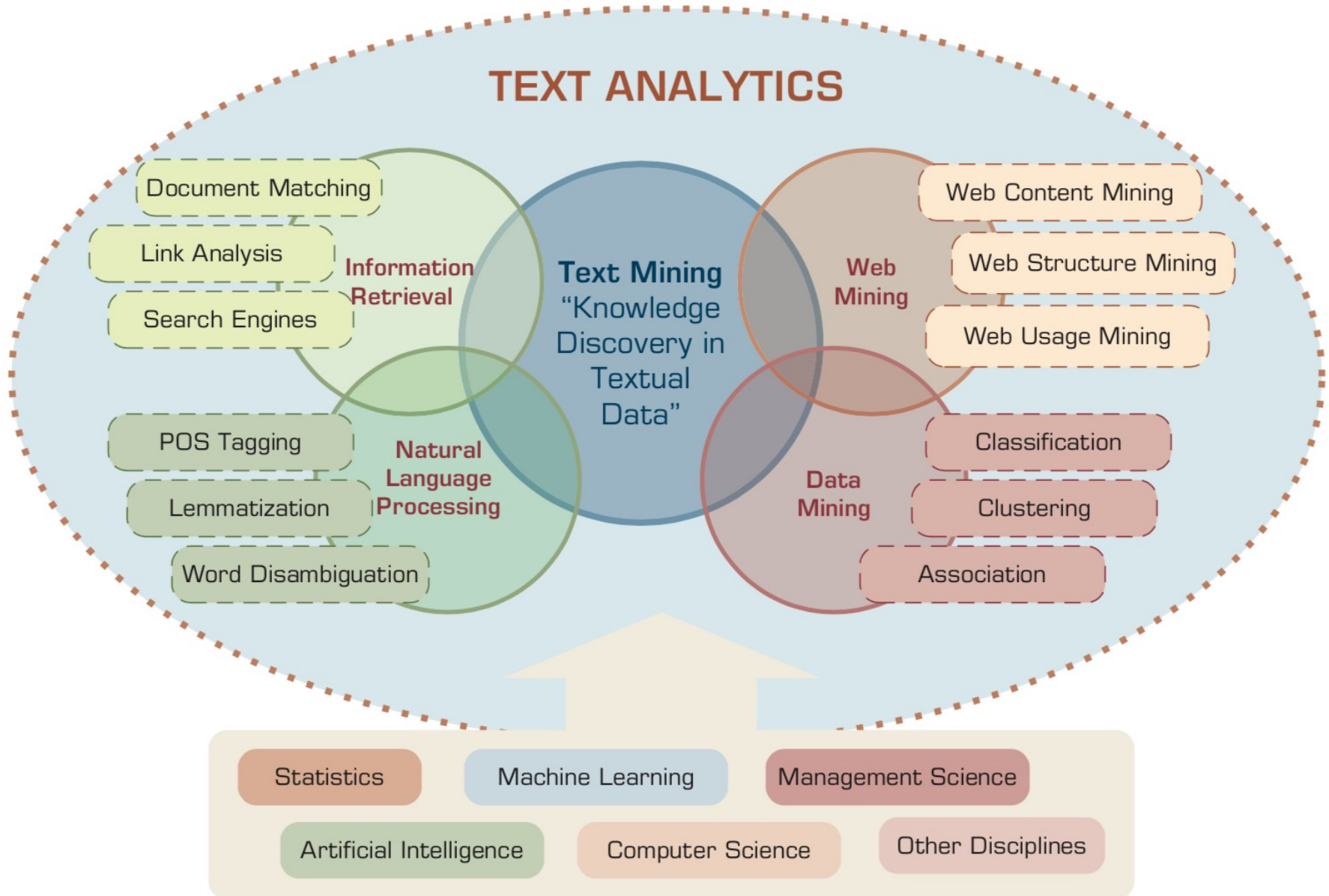


# Agents interact with environments through sensors and actuators



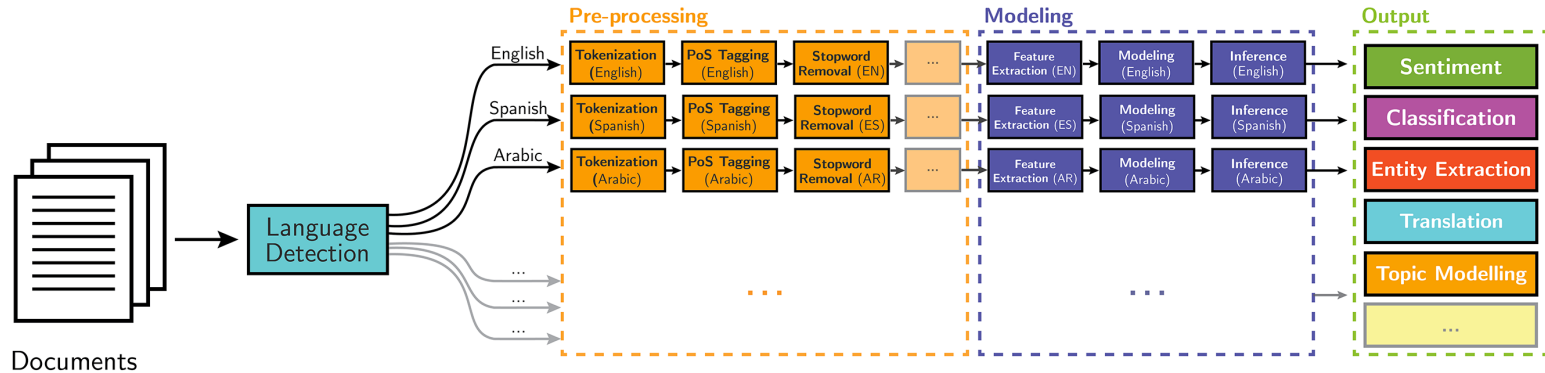
**Deep Learning  
for  
Natural Language  
Processing**

# AI for Text Analytics

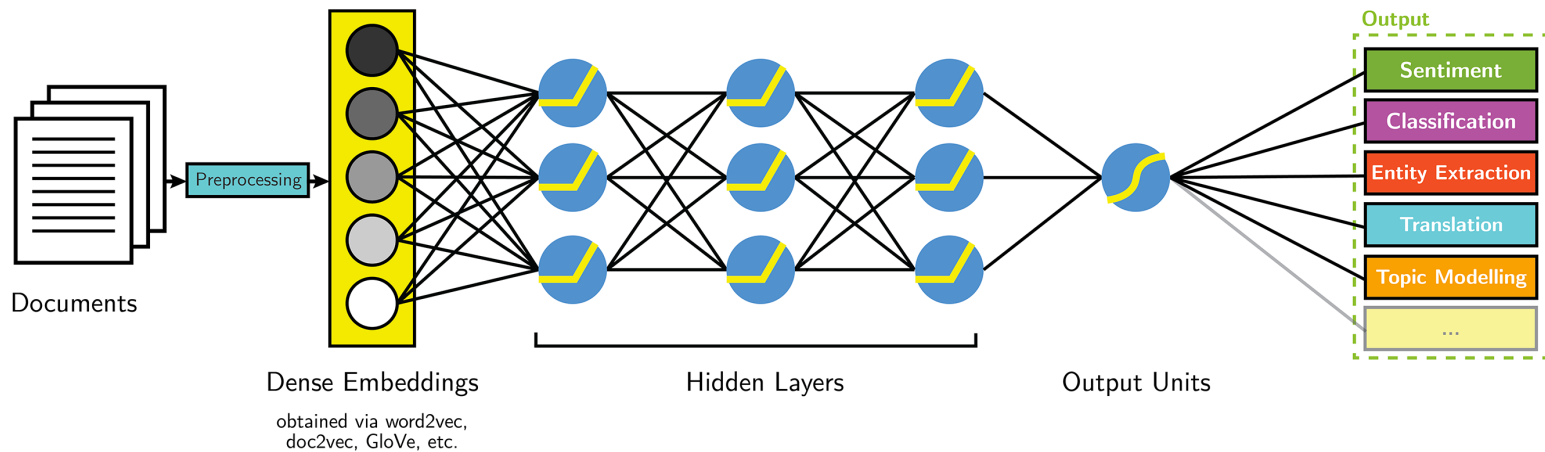


# NLP

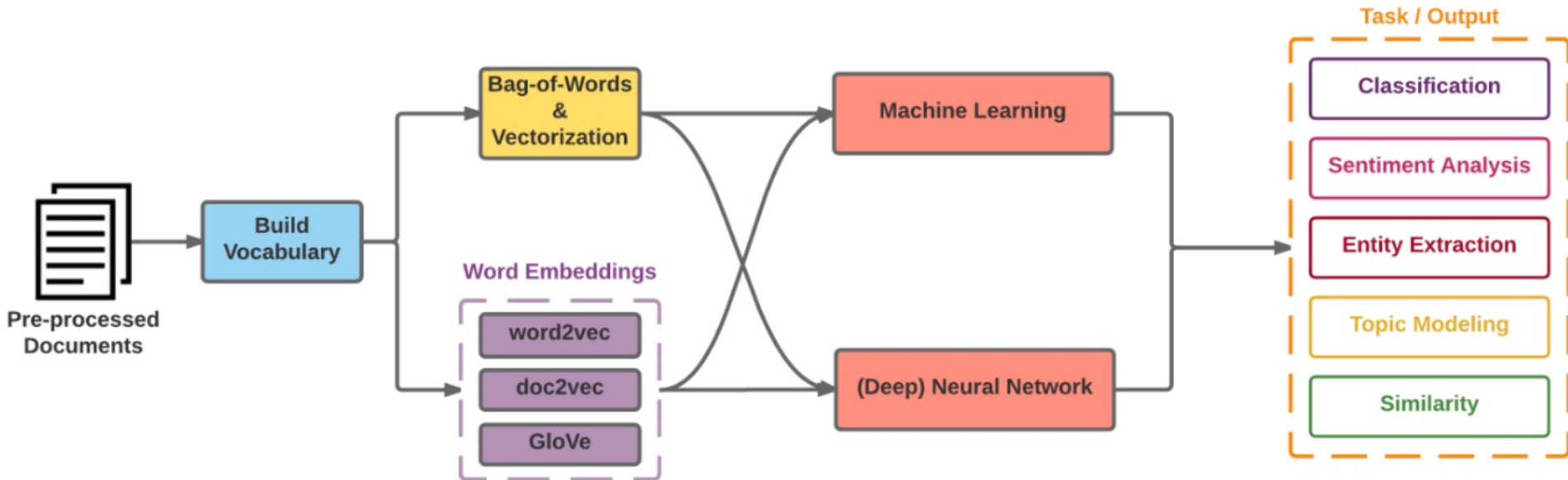
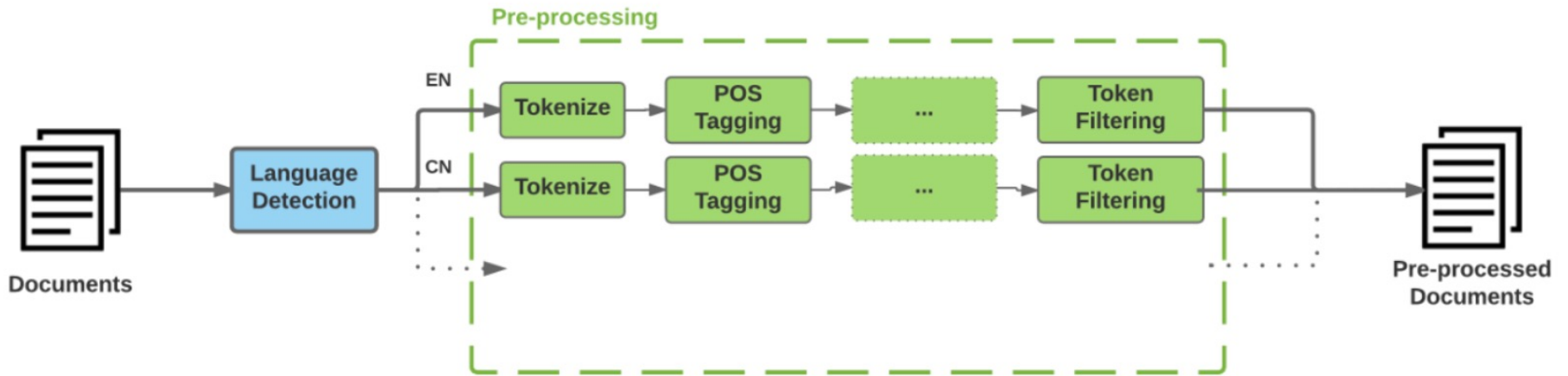
## Classical NLP



## Deep Learning-based NLP

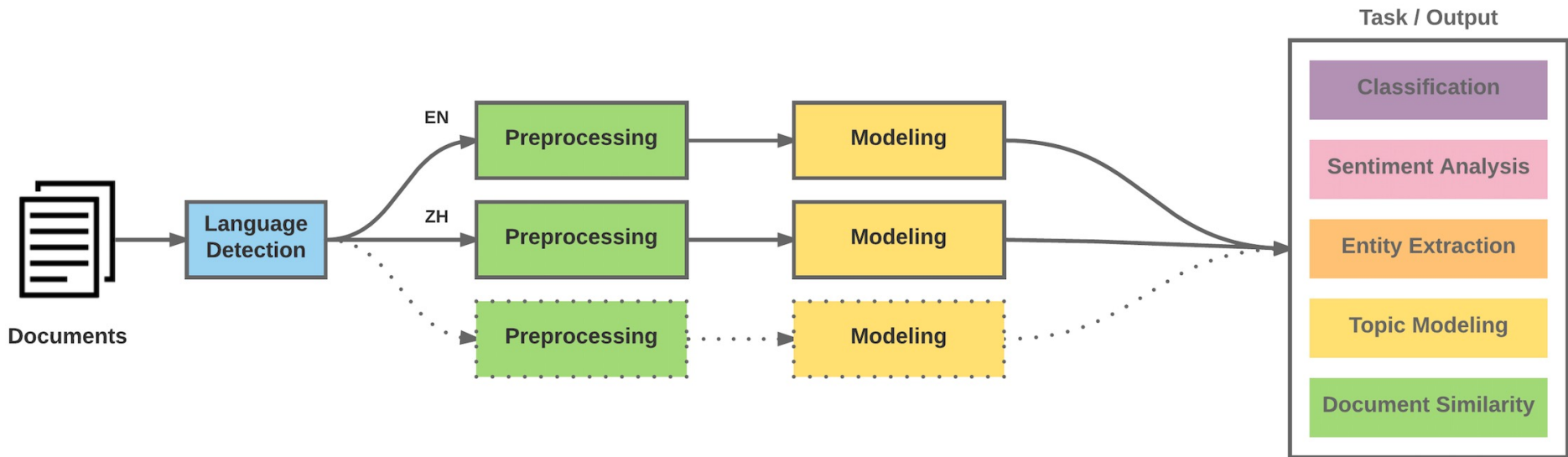


# Modern NLP Pipeline

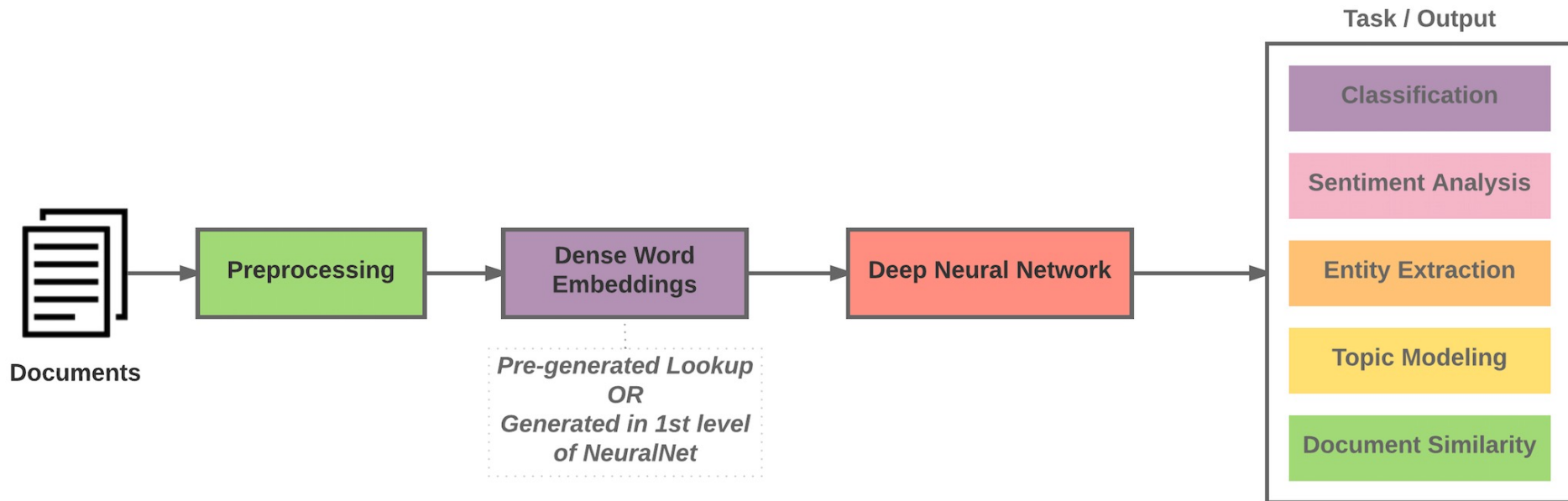




# Modern NLP Pipeline



# 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

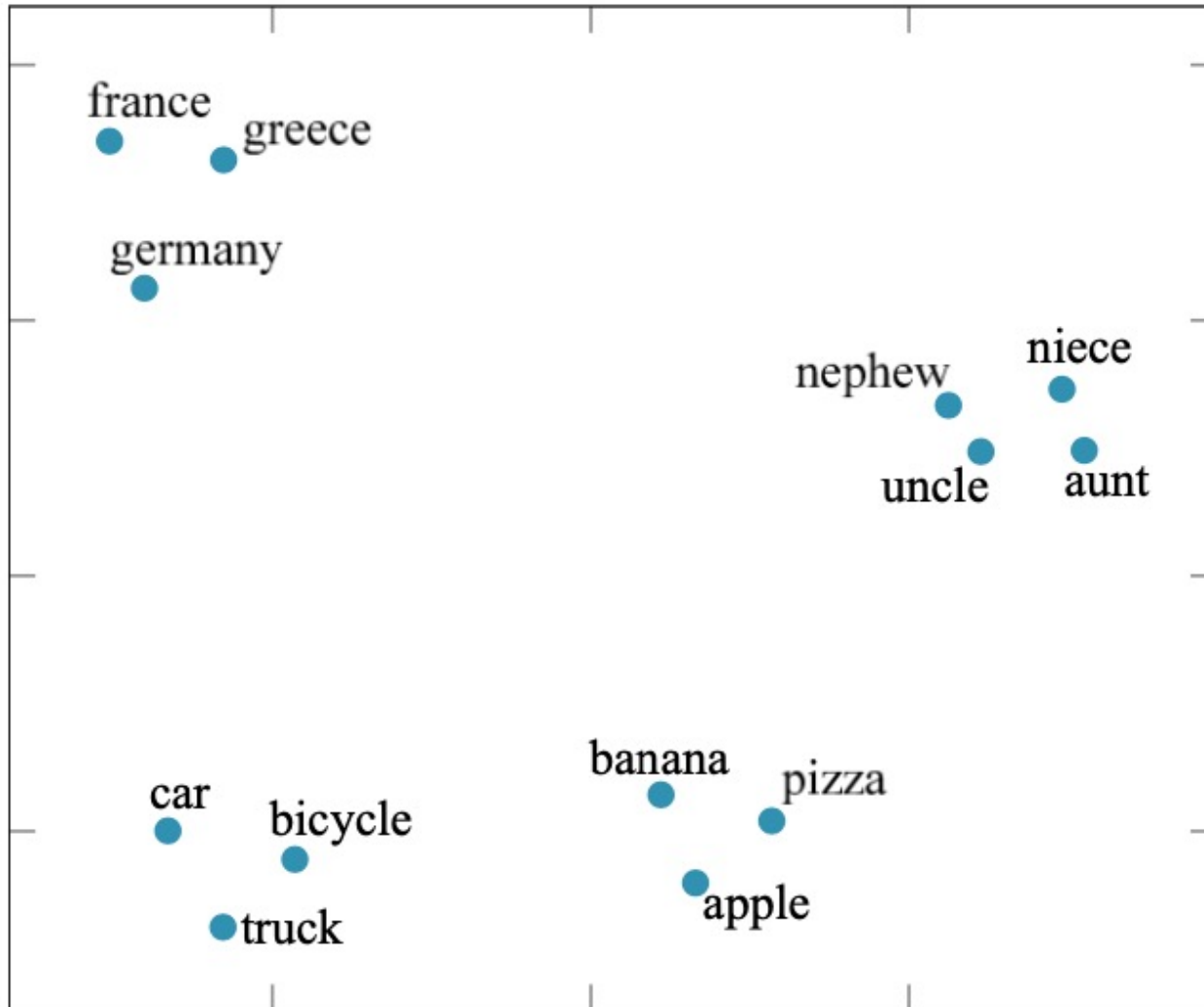
'The mouse ran up the clock' =

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

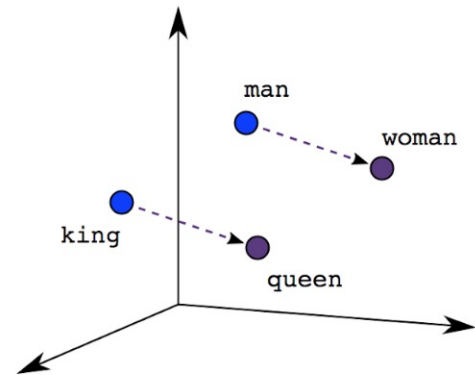


# Word Embedding model

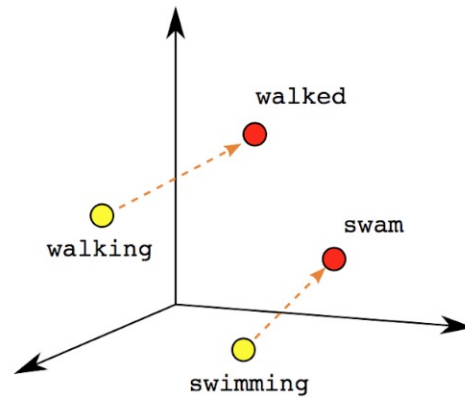
answer the question “A is to B as C is to [what]?”

<b>A</b>	<b>B</b>	<b>C</b>	<b>D = C + (B - A)</b>	<b>Relationship</b>
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>

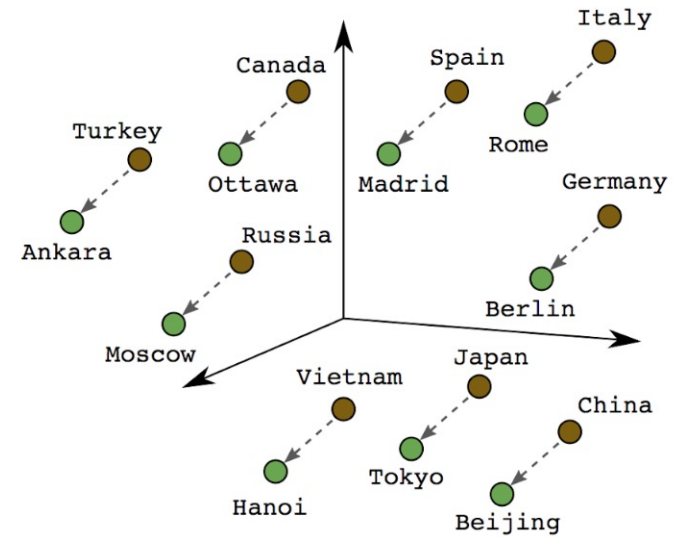
# Word embeddings



Male-Female



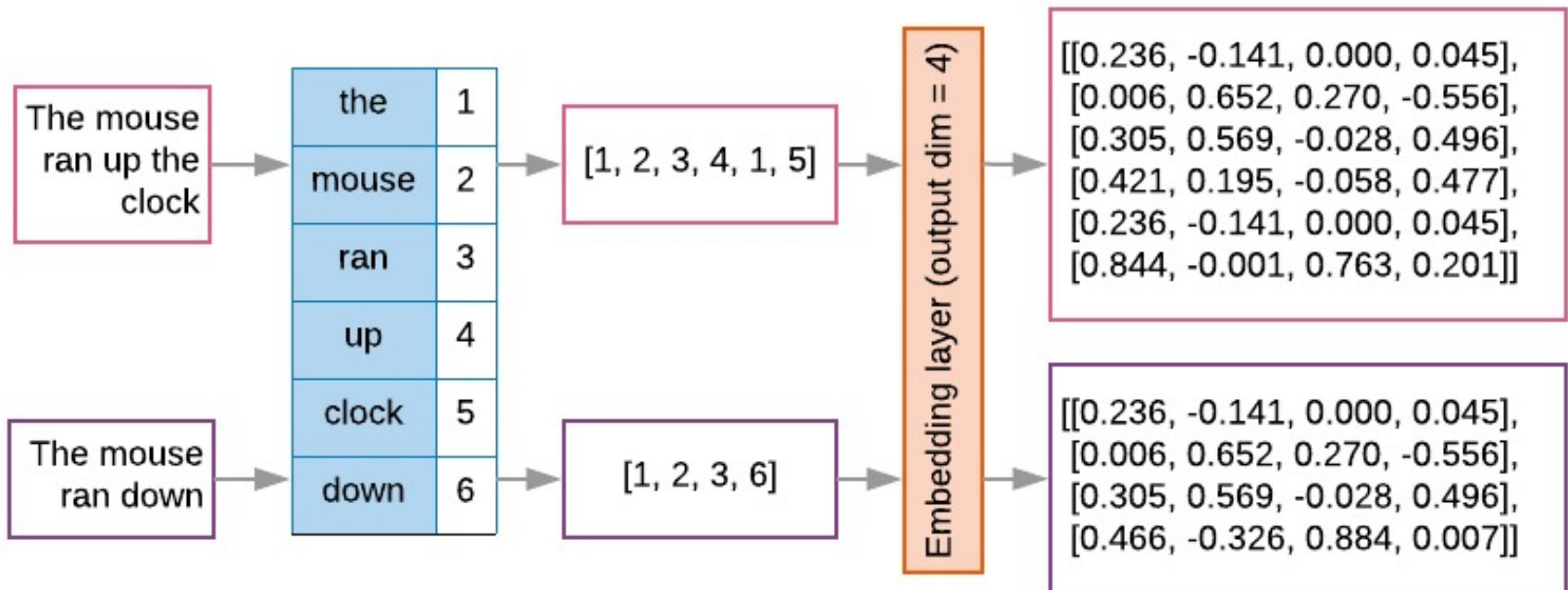
Verb Tense



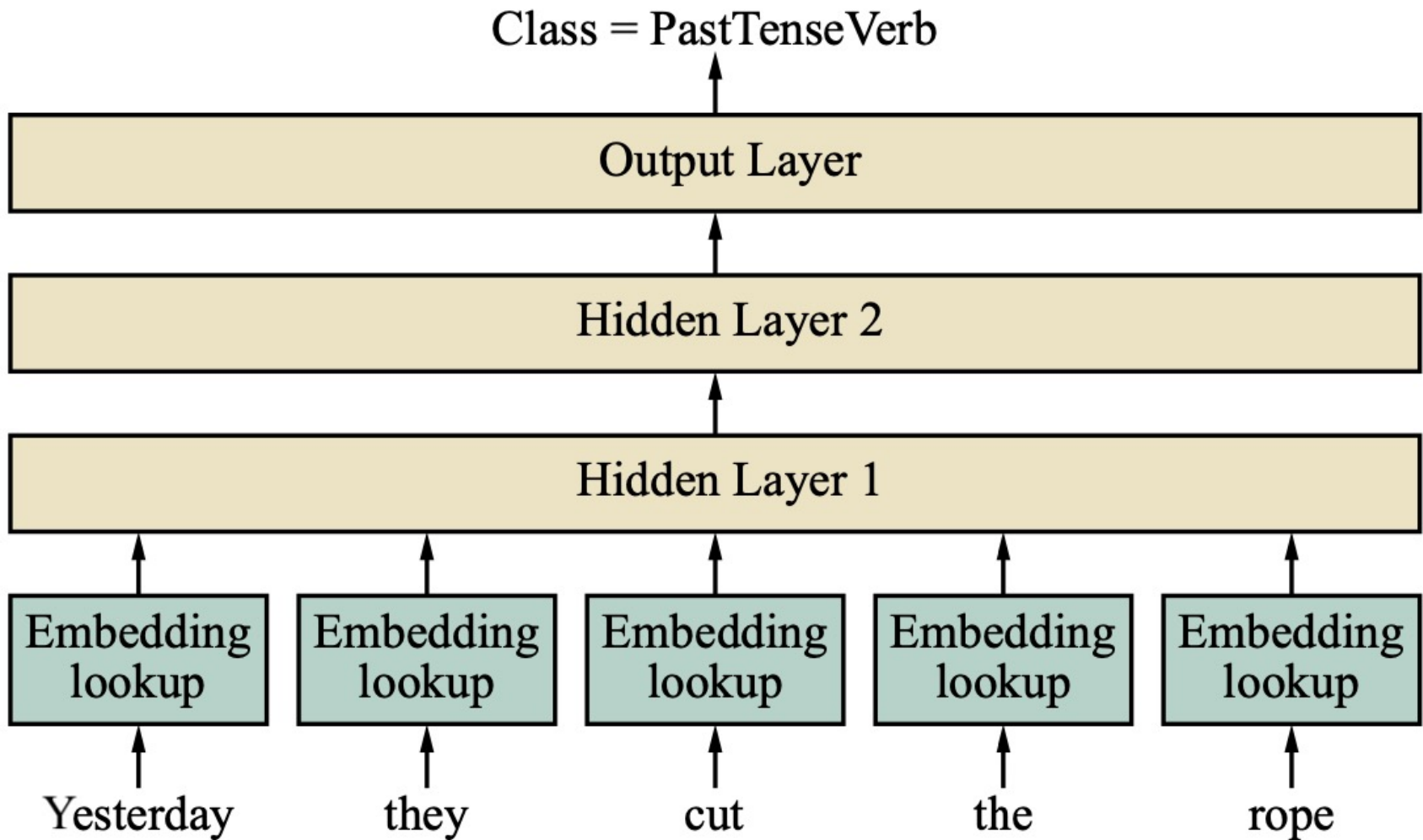
Country-Capital



# Word embeddings



# Feedforward part-of-speech (POS) tagging model

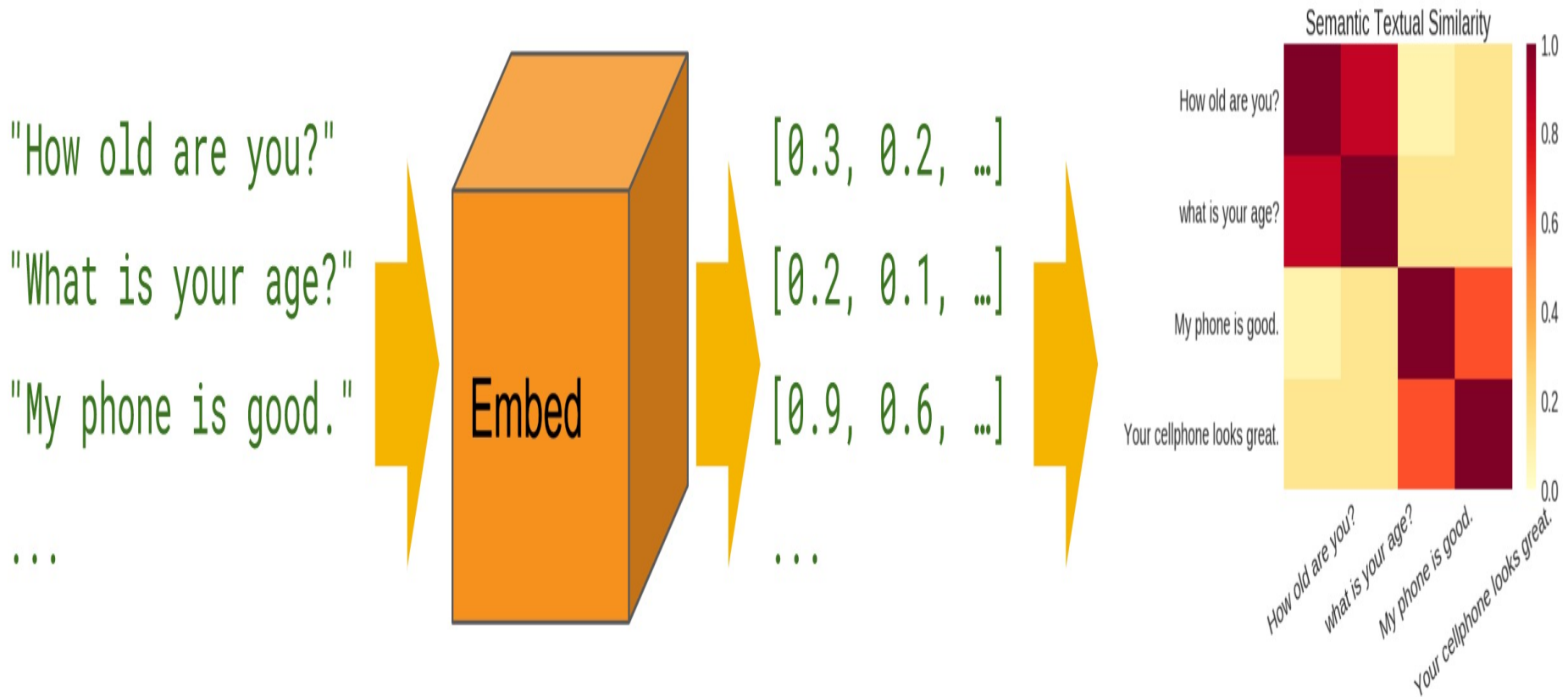


# Universal Sentence Encoder (USE)

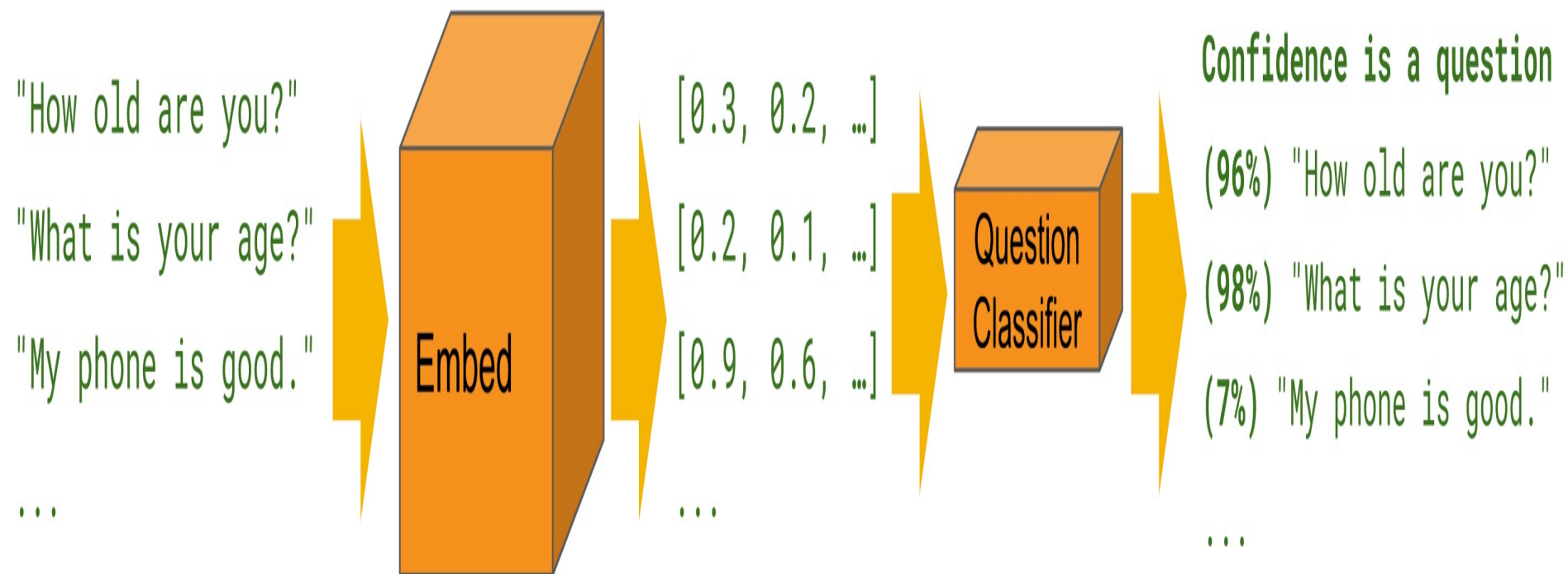
- The **Universal Sentence Encoder** encodes **text** into high-dimensional **vectors** that can be used for text classification, semantic similarity, clustering and other natural language tasks.
- The universal-sentence-encoder model is trained with a **deep averaging network (DAN)** encoder.

# 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>

The screenshot shows a Google Colab notebook titled "python101.ipynb". The interface includes a top navigation bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help" menus. A "Table of contents" sidebar on the left lists various topics, with "Universal Sentence Encoder (USE)" highlighted. The main workspace contains a code cell with the following Python code:

```
[ ] 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)
```

Below the code cell, the output shows: `module https://tfhub.dev/google/universal-sentence-encoder/4 loaded`. A second code cell is partially visible at the bottom:

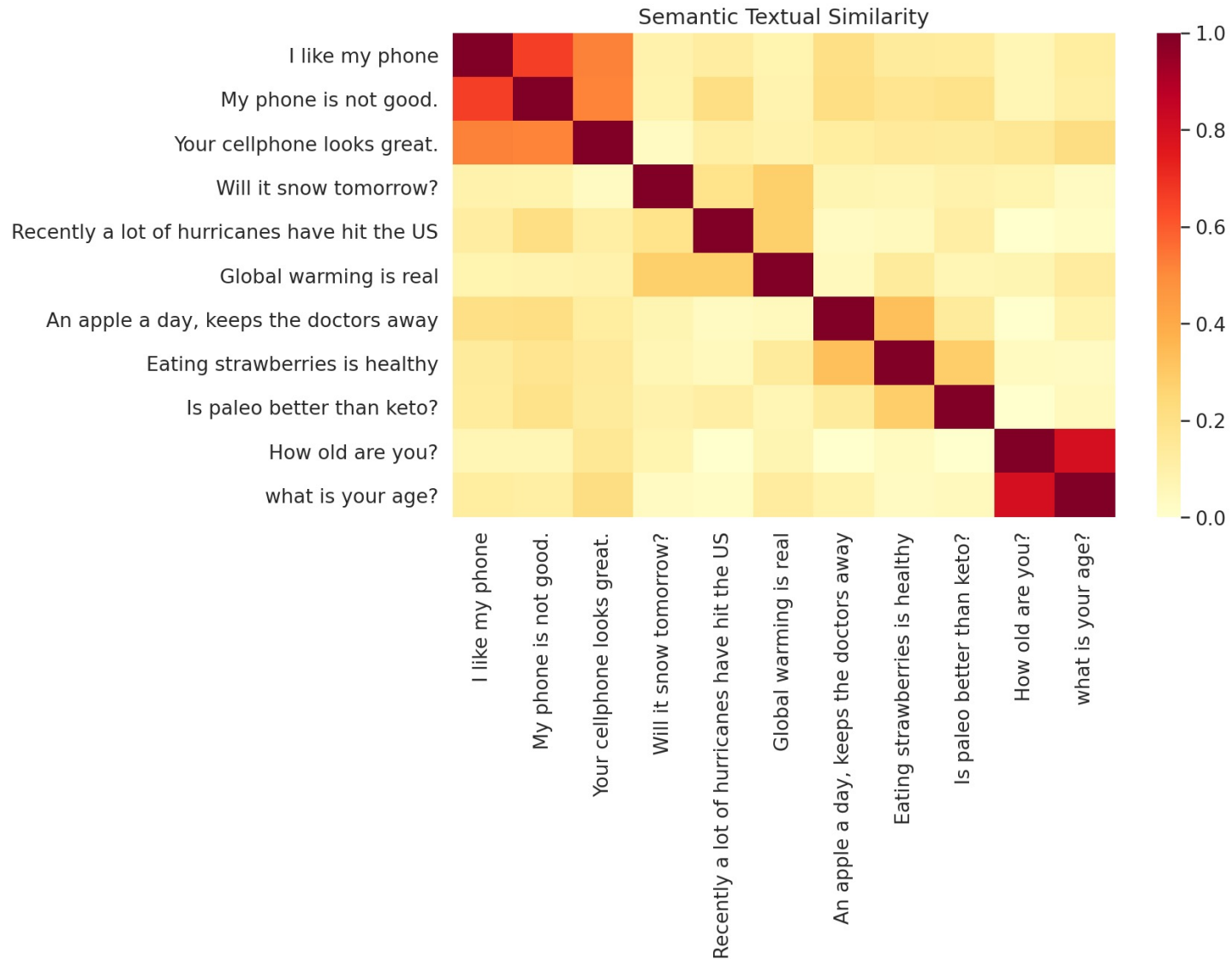
```
[ ] 1 word = "Elephant"
2 sentence = "I am a sentence for which I would like to get its embedding."
```

<https://tinyurl.com/aintpuppython101>



# Python in Google Colab (Python101)

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

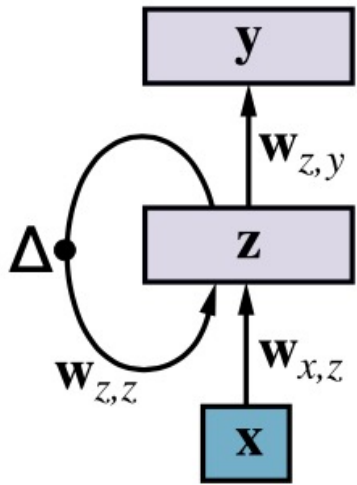


<https://tinyurl.com/aintpupython101>

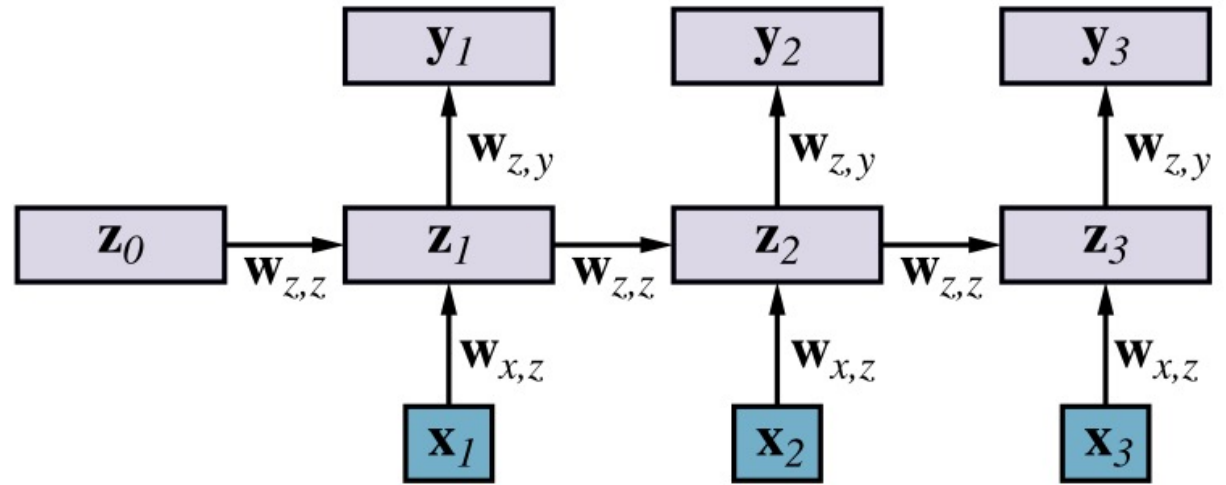
# Outline

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

# RNN

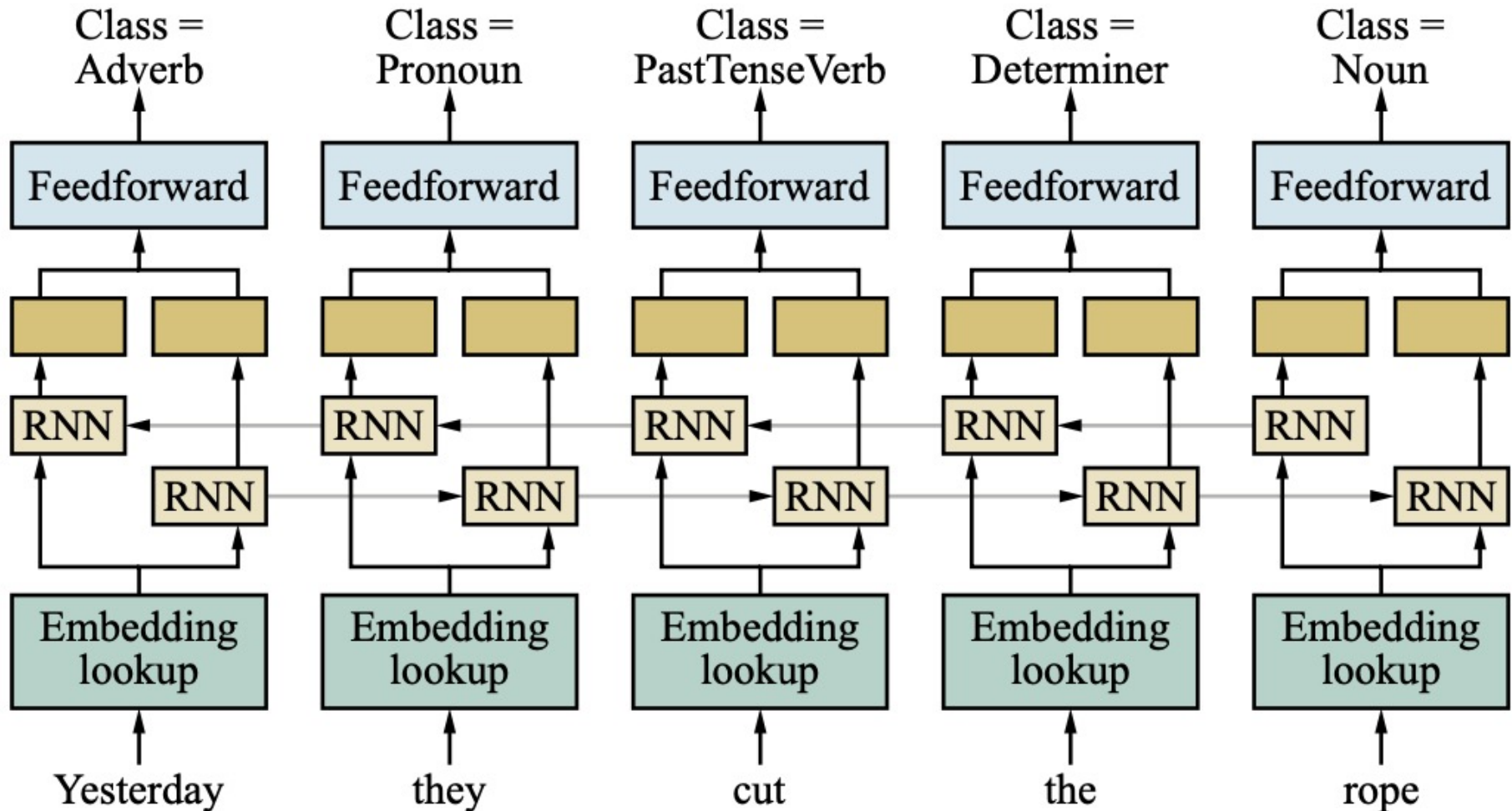


(a)



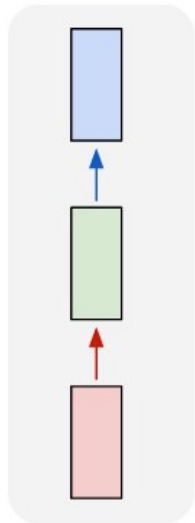
(b)

# Bidirectional RNN network for POS tagging



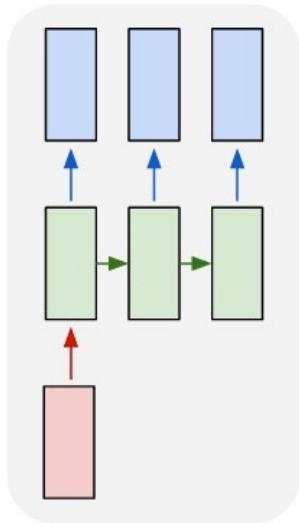
# LSTM Recurrent Neural Network

one to one



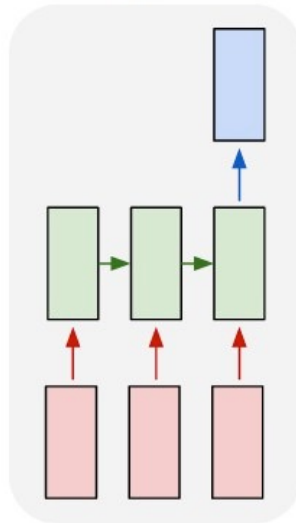
**Traditional  
Neural  
Network**

one to many



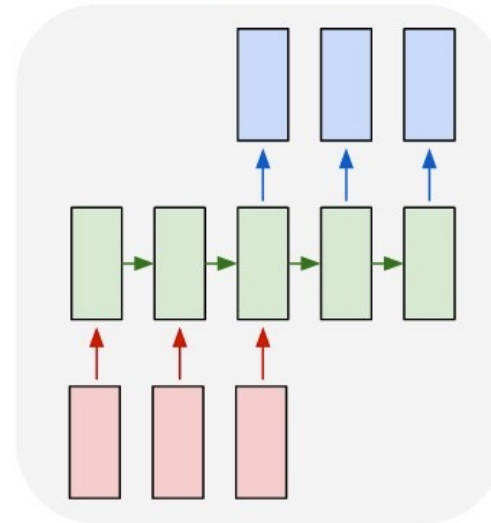
**Music  
Generation**

many to one



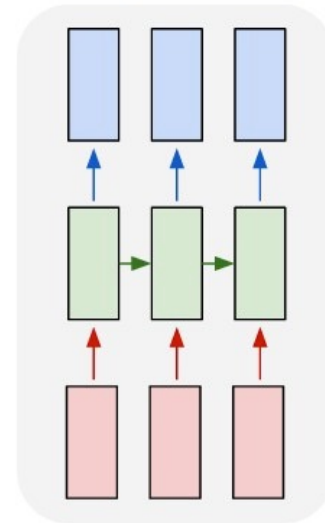
**Sentiment  
Classification**

many to many



**Name  
Entity  
Recognition**

many to many

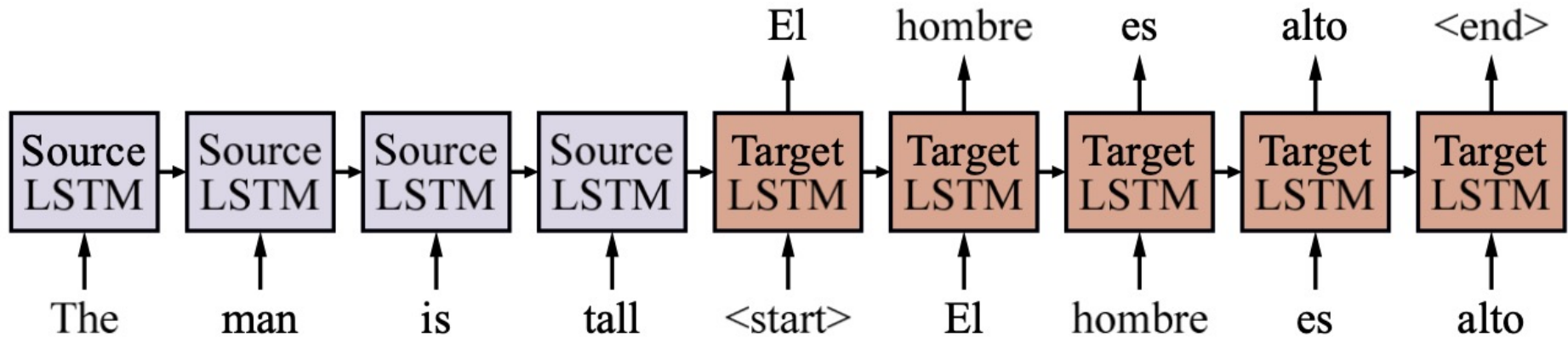


**Machine  
Translation**

# Outline

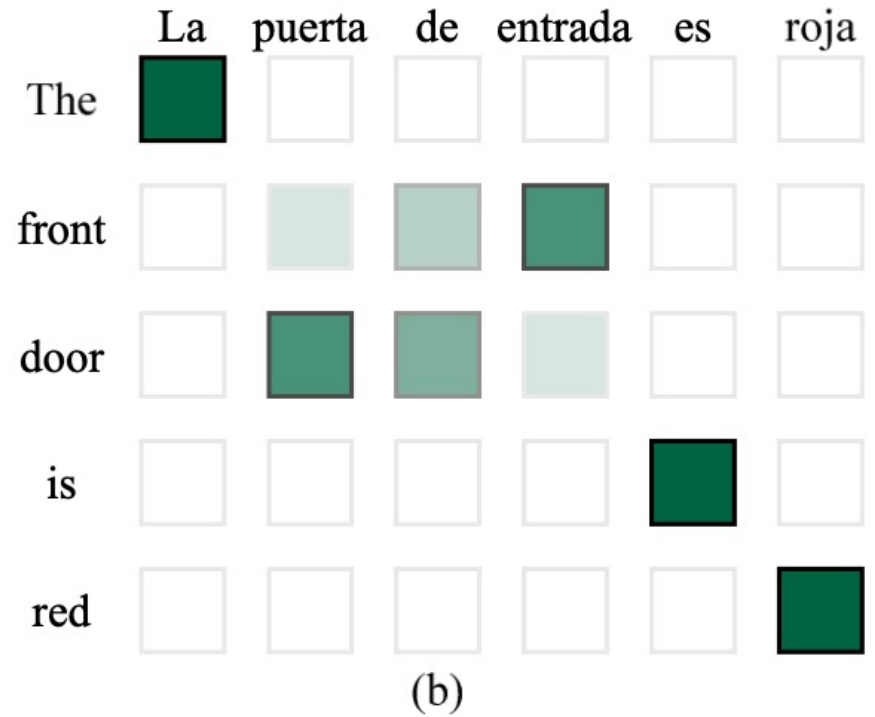
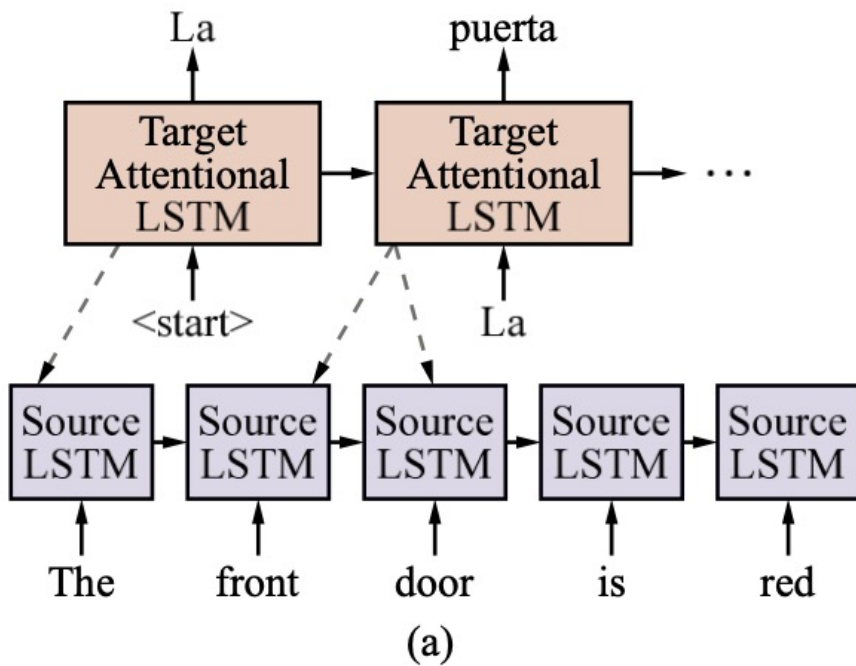
- Word Embeddings
- Recurrent Neural Networks for NLP
- **Sequence-to-Sequence Models**
- The Transformer Architecture
- Pretraining and Transfer Learning
- State of the art (SOTA)

# Sequence-to-Sequence model



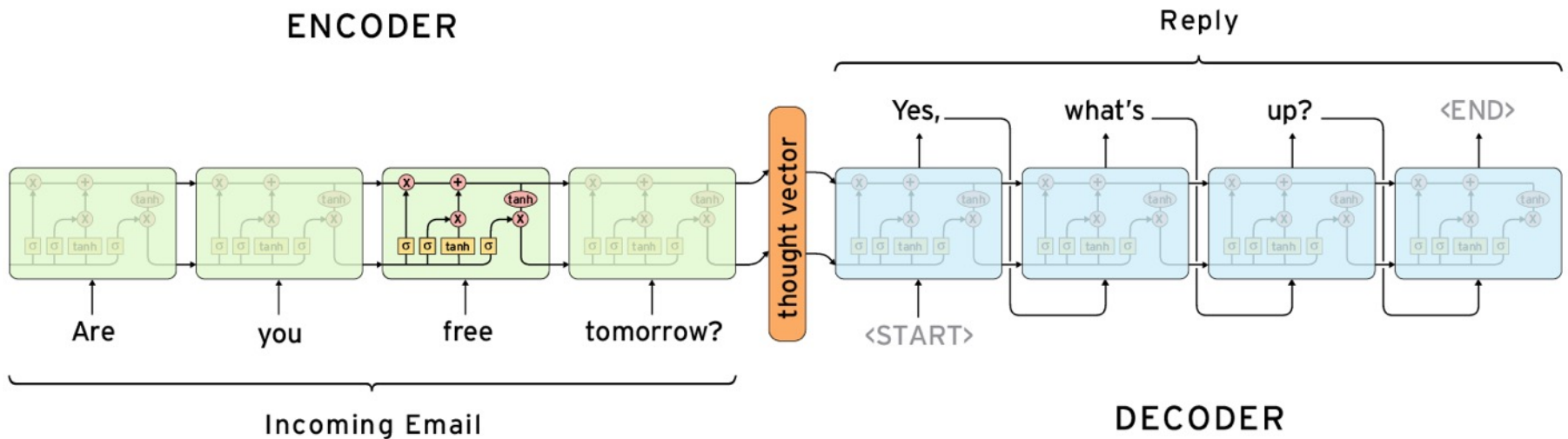
# Attentional Sequence-to-Sequence model

## for English-to-Spanish translation

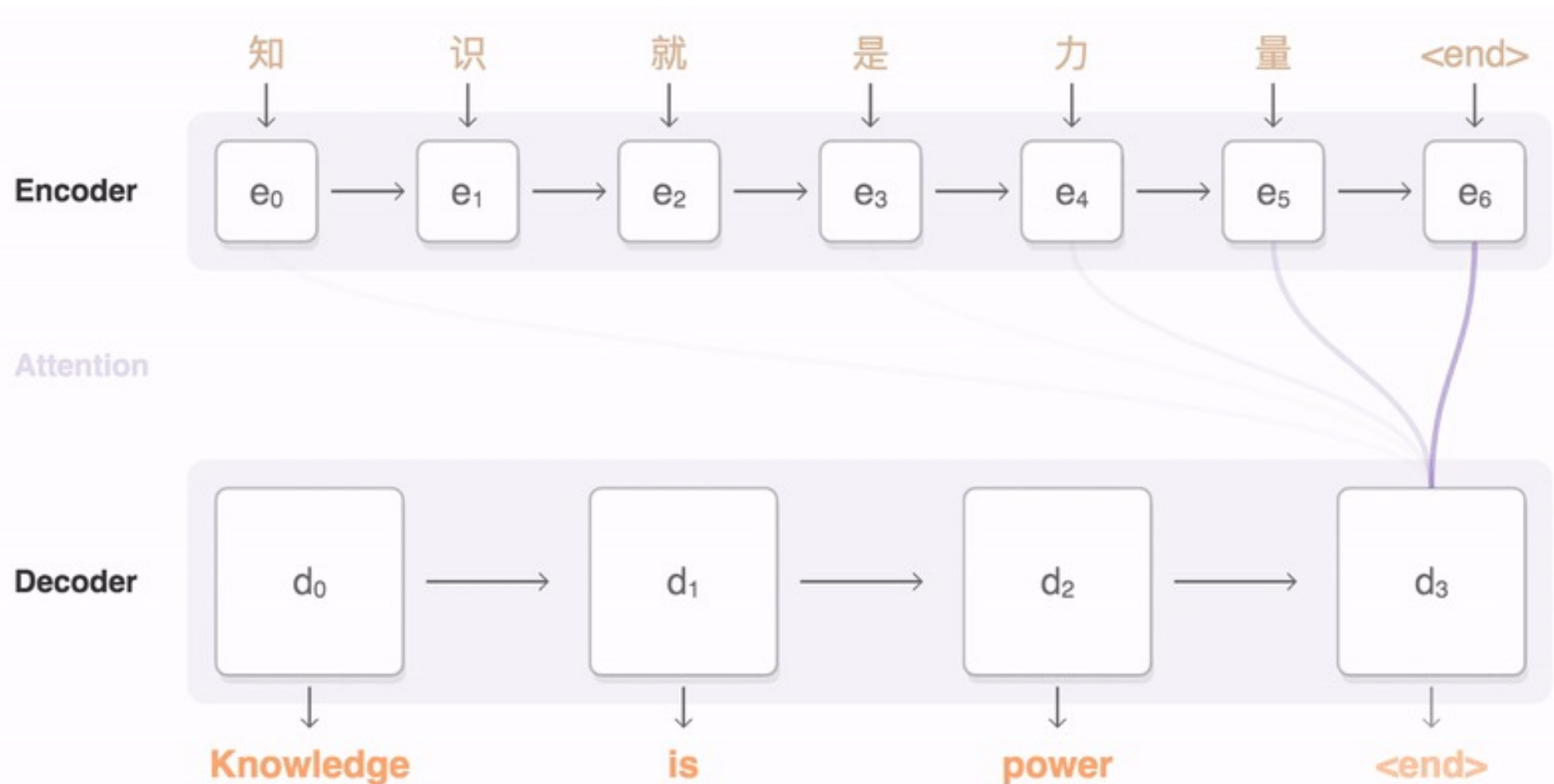




# The Sequence to Sequence model (seq2seq)



# Sequence to Sequence (Seq2Seq)

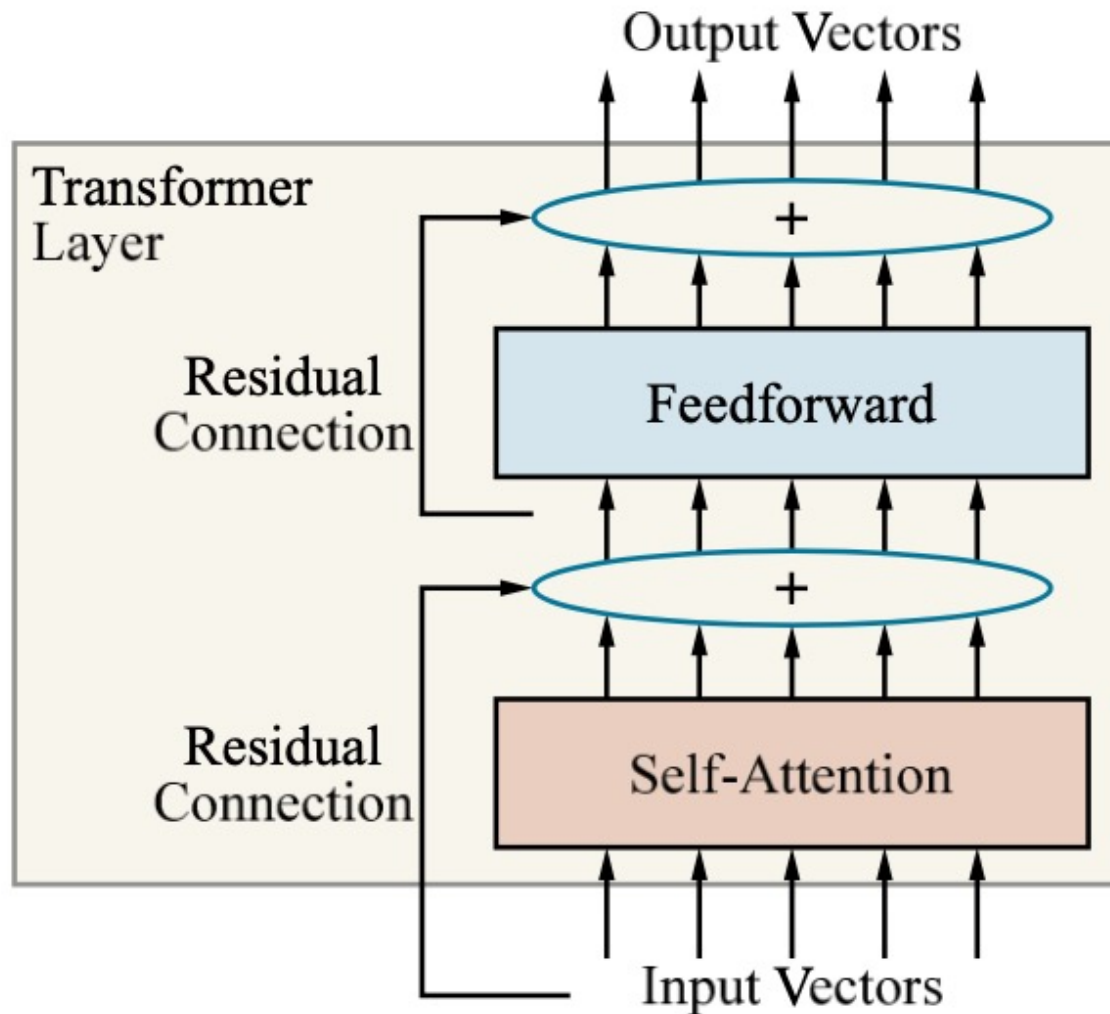


# Outline

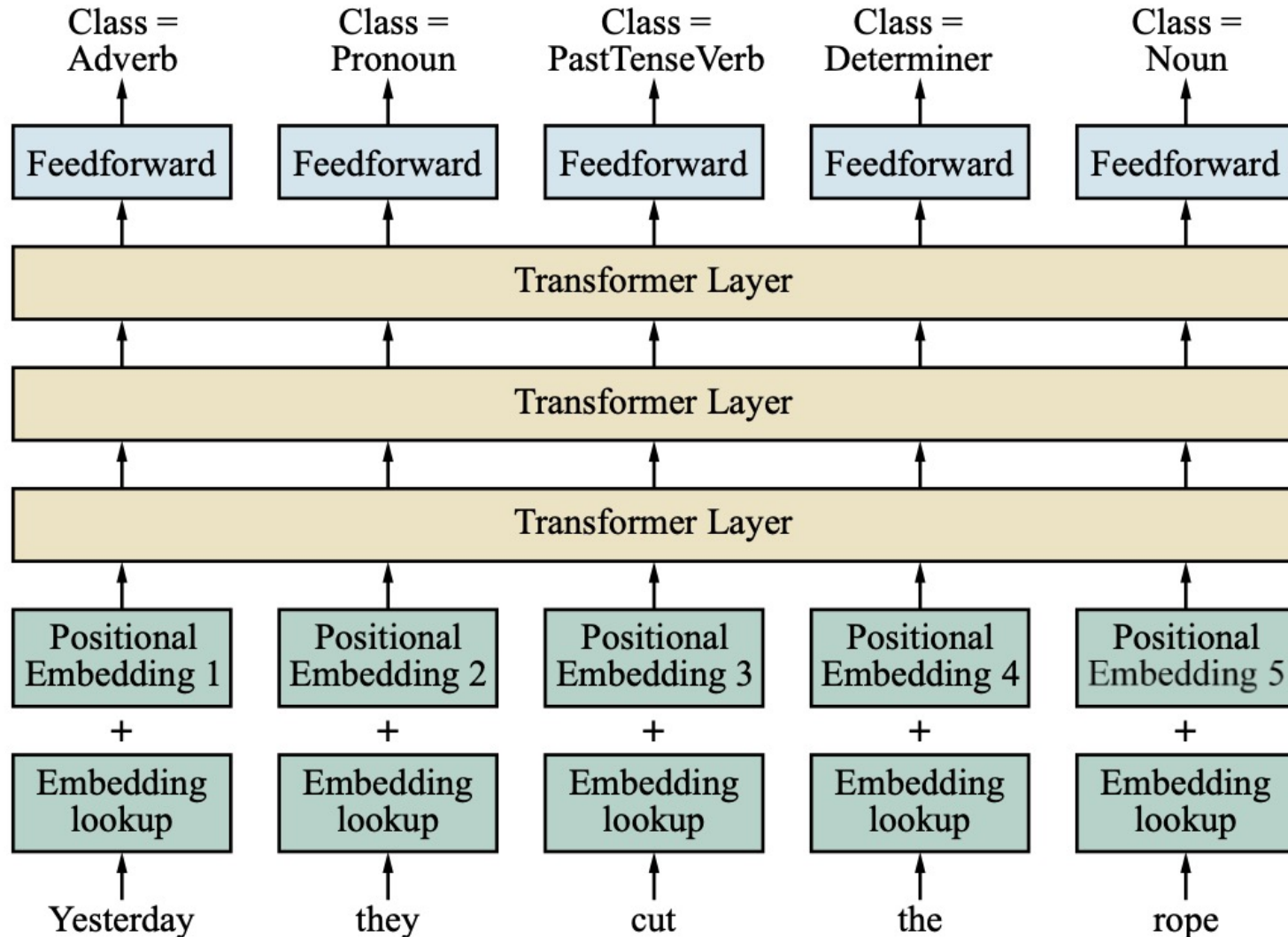
- Word Embeddings
- Recurrent Neural Networks for NLP
- Sequence-to-Sequence Models
- **The Transformer Architecture**
- Pretraining and Transfer Learning
- State of the art (SOTA)

# Single-layer Transformer

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

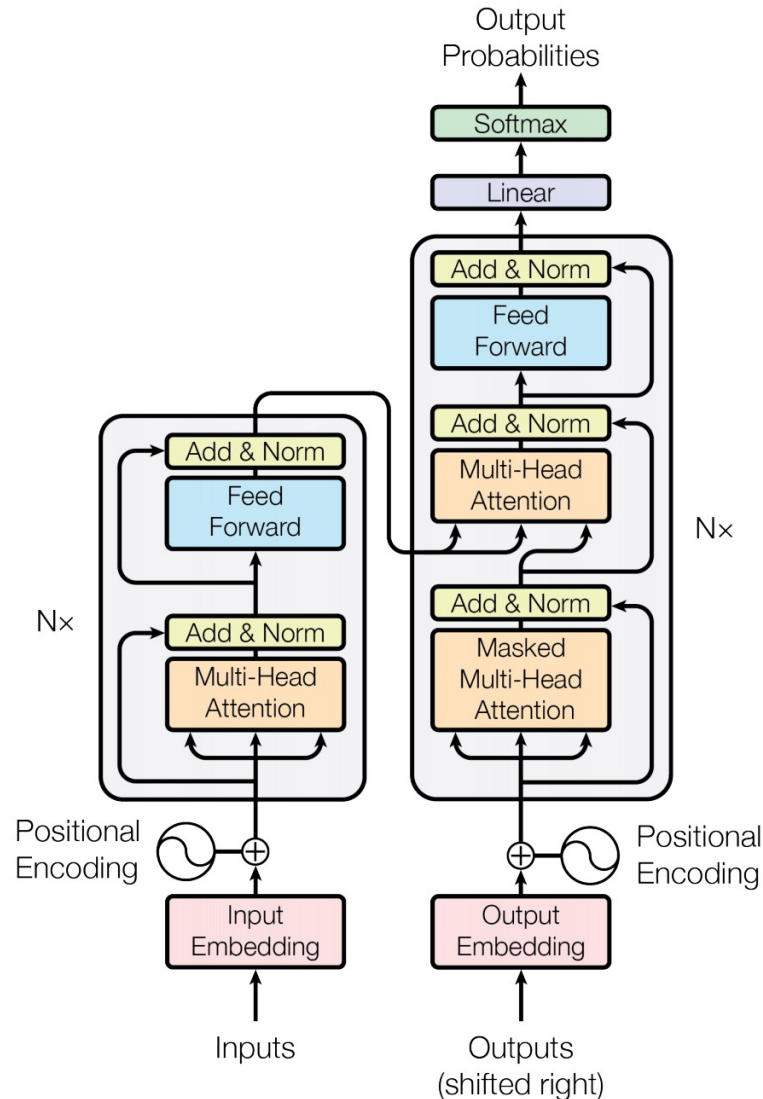


# Transformer Architecture for POS Tagging



# Transformer (Attention is All You Need)

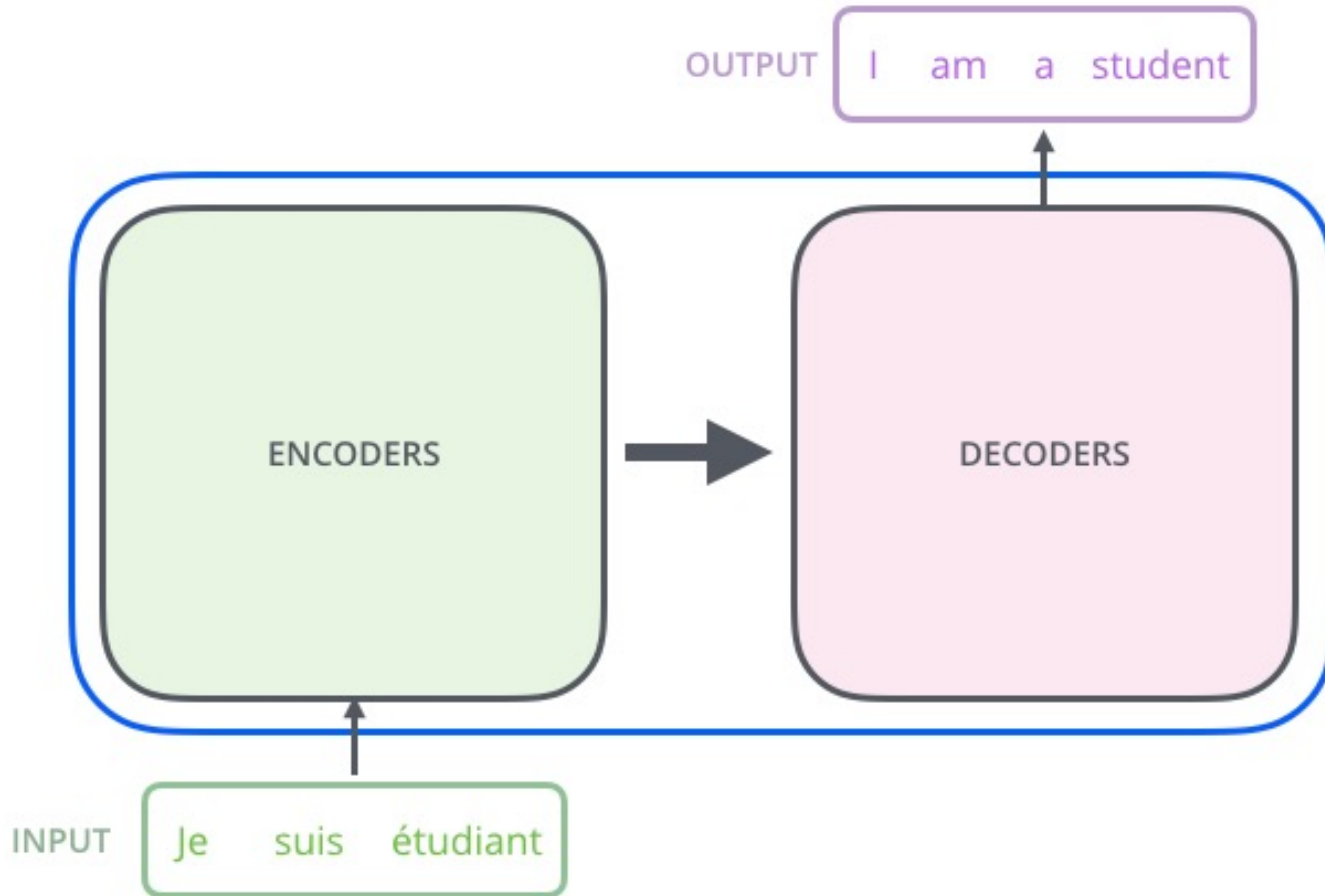
(Vaswani et al., 2017)



# Transformer



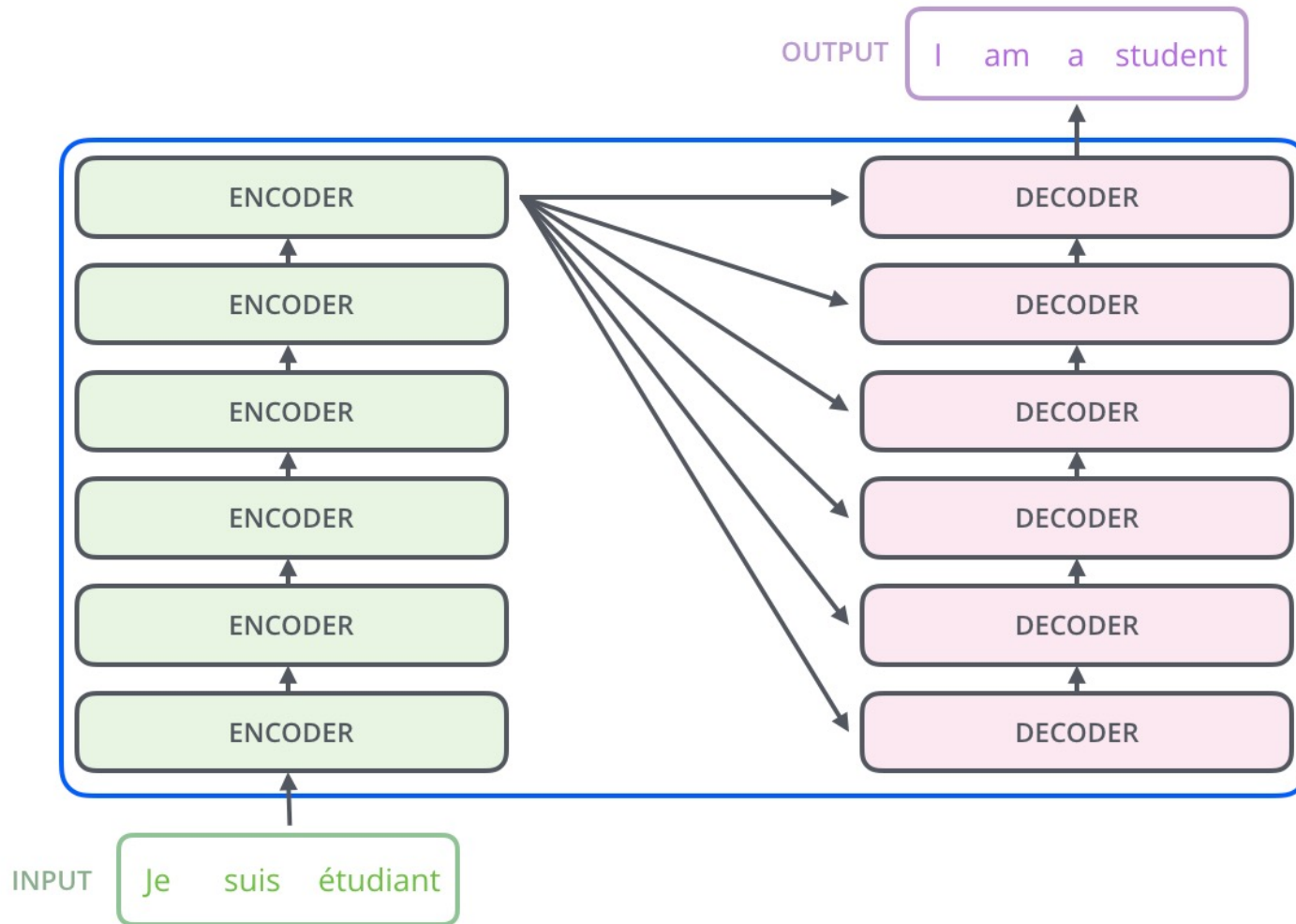
# Transformer Encoder Decoder





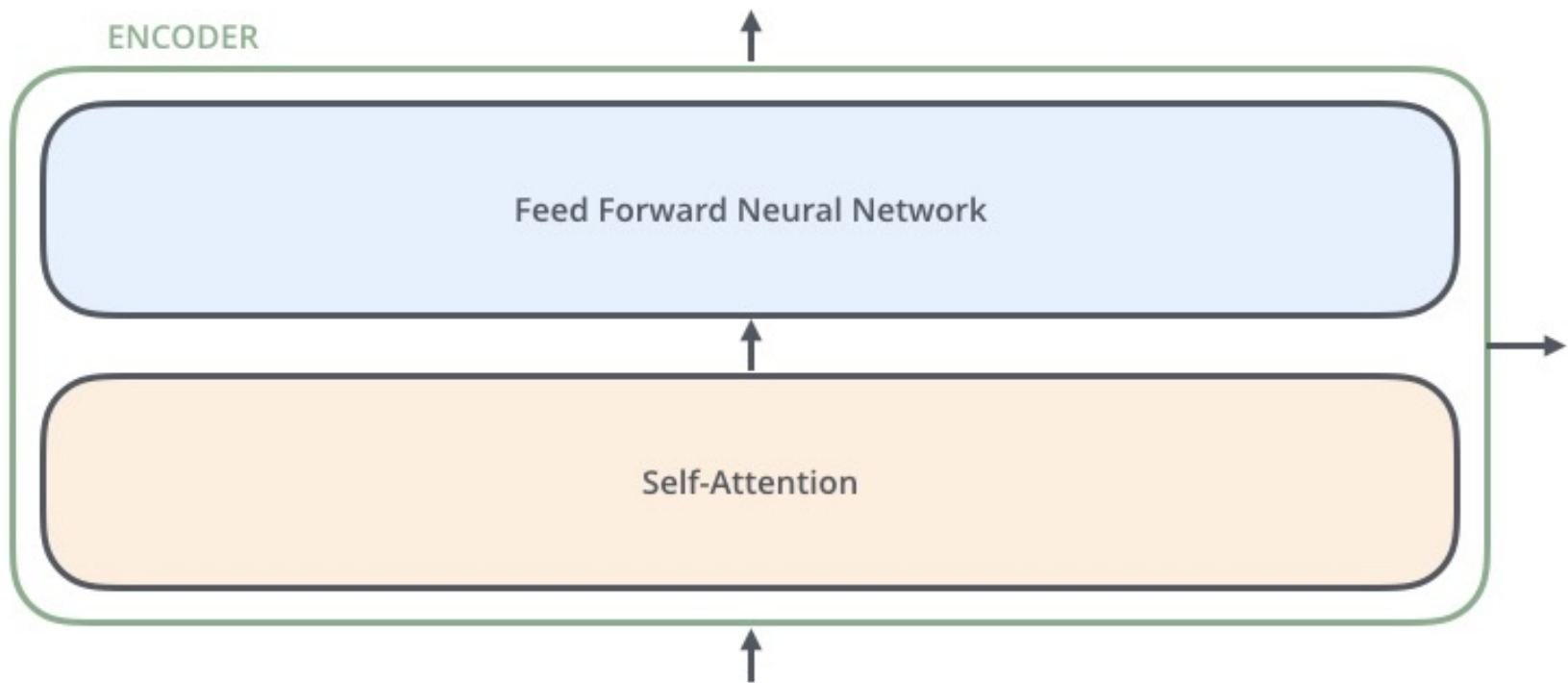
# Transformer

## Encoder Decoder Stack

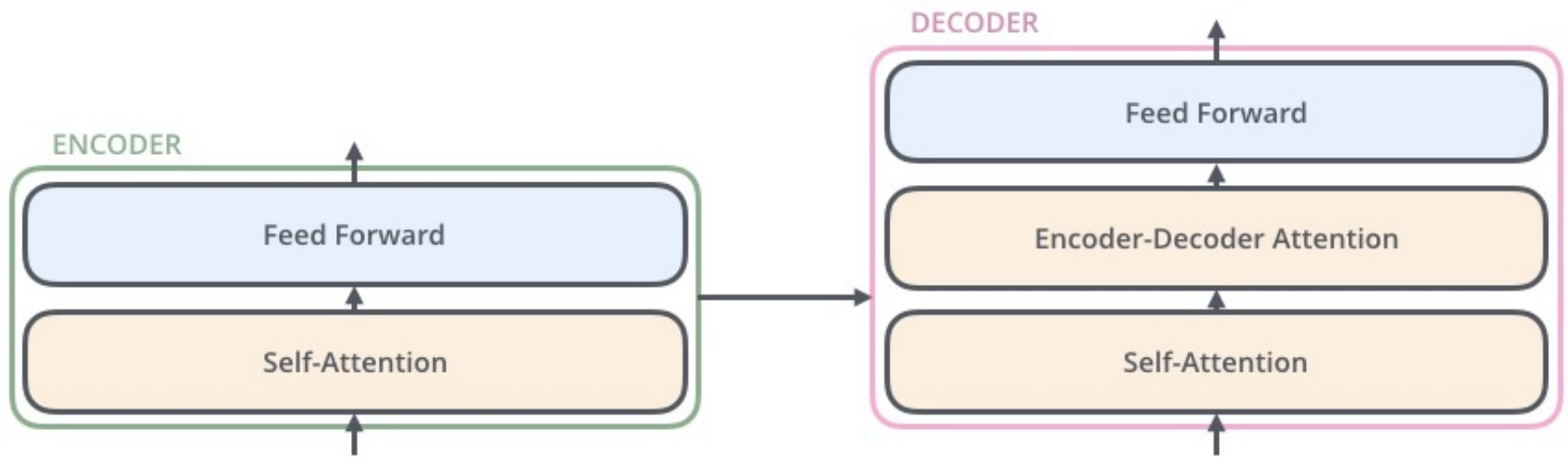


# Transformer

## Encoder Self-Attention



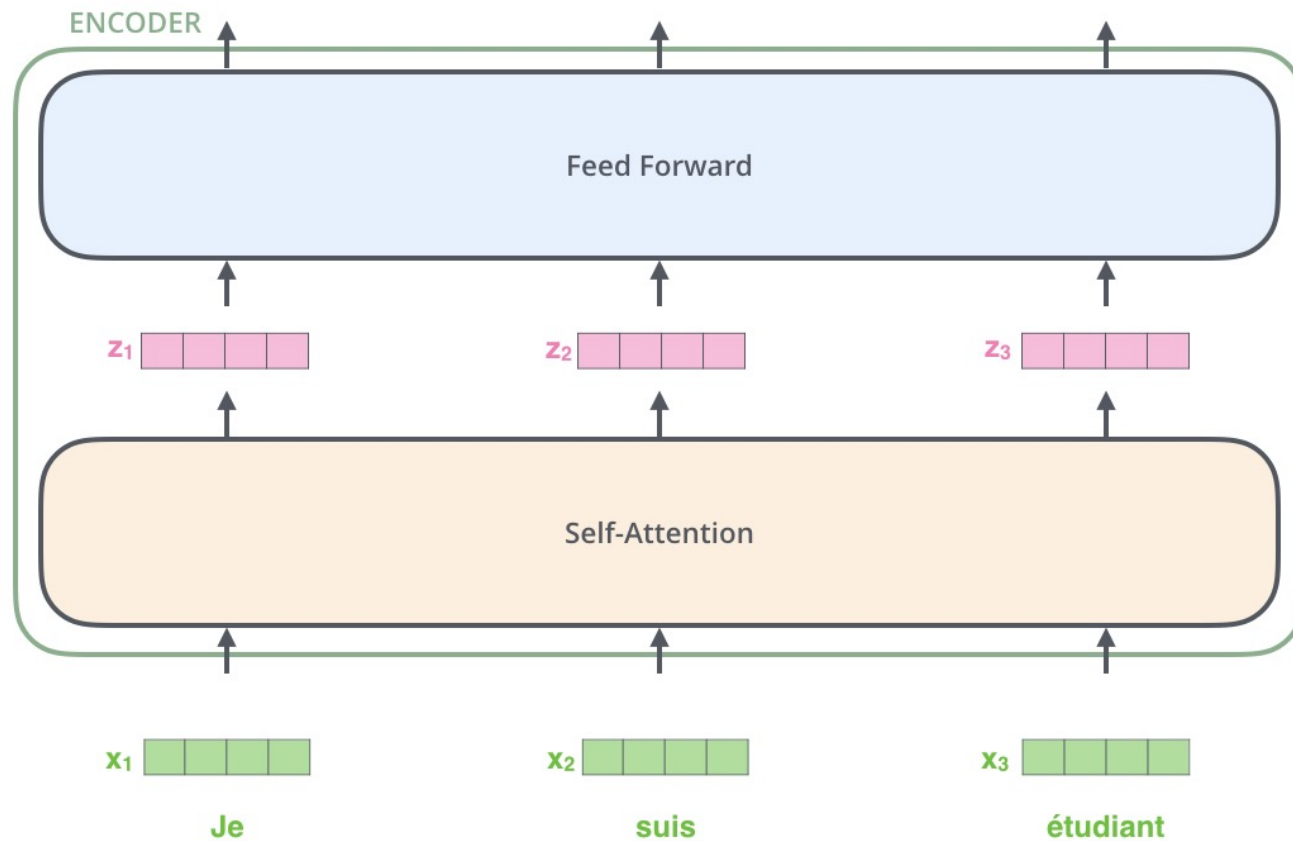
# Transformer Decoder



# Transformer

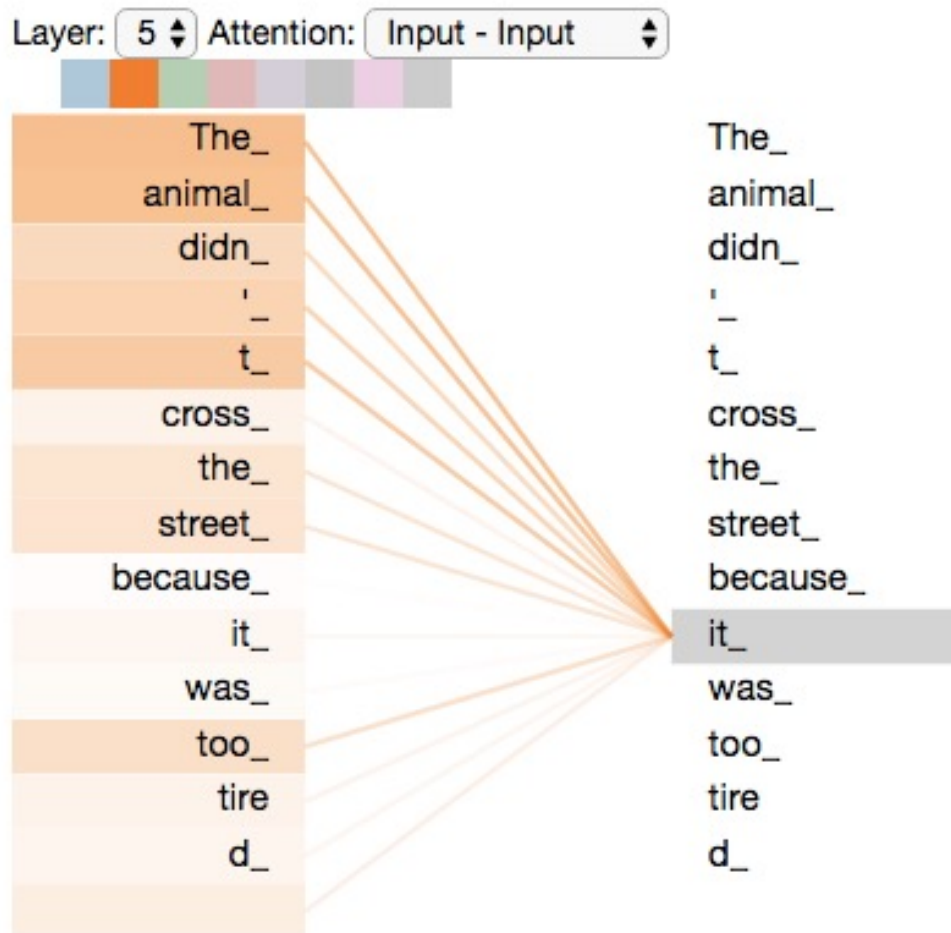
## Encoder with Tensors

### Word Embeddings



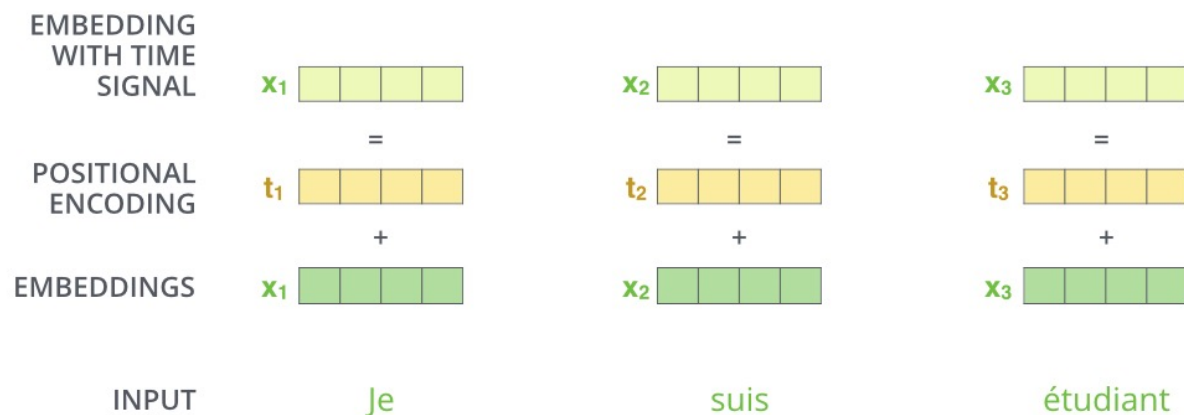
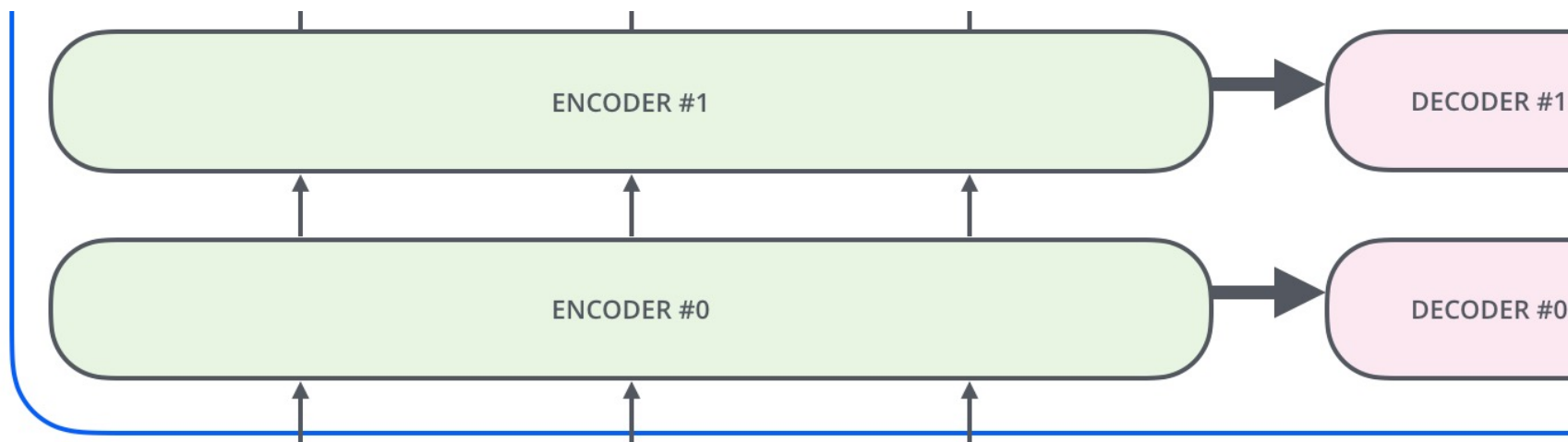
# Transformer

## Self-Attention Visualization



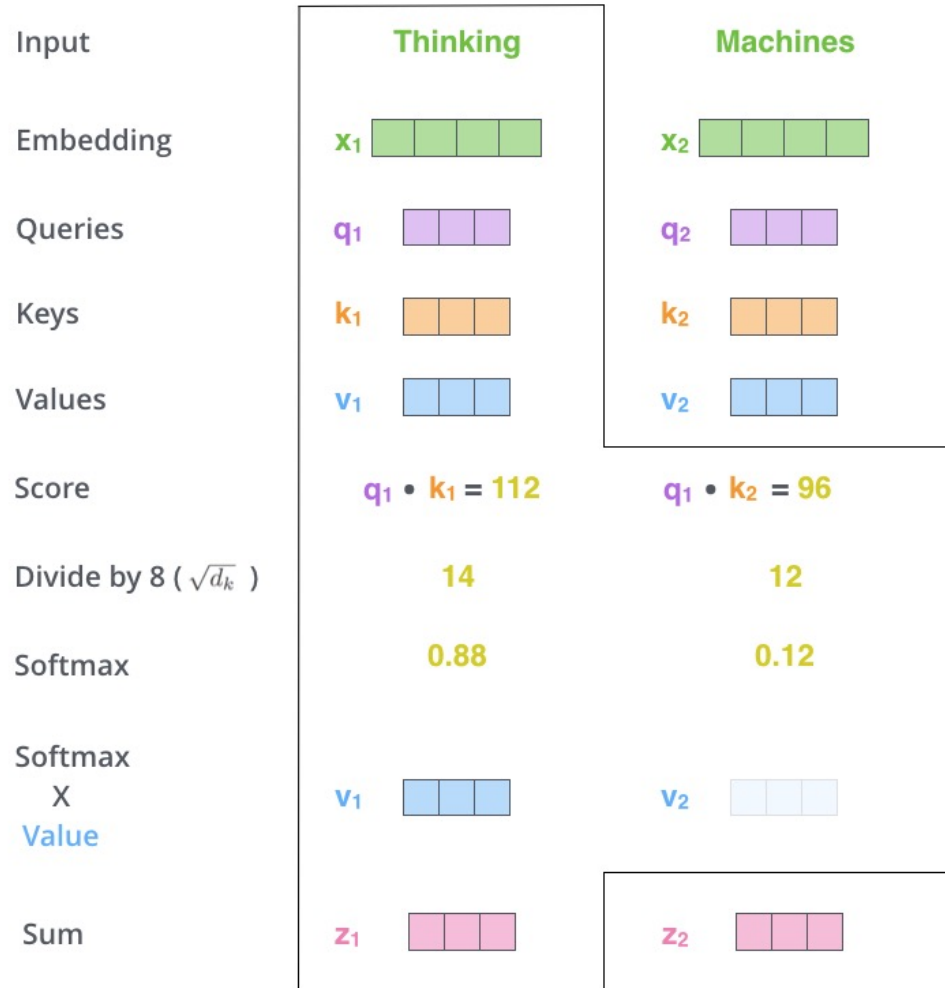
# Transformer

## 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

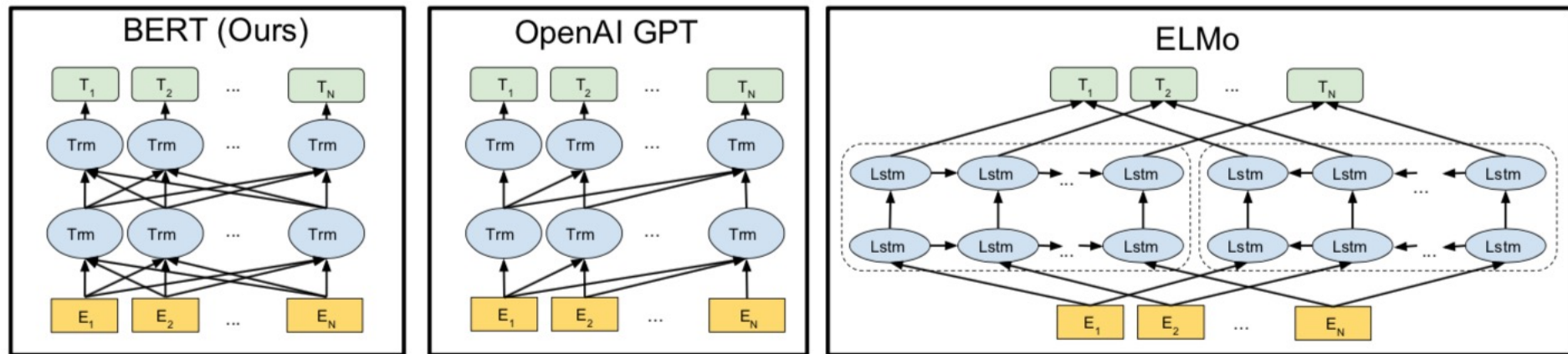
**BERT: Pre-training of Deep Bidirectional Transformers for  
Language Understanding**

**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.

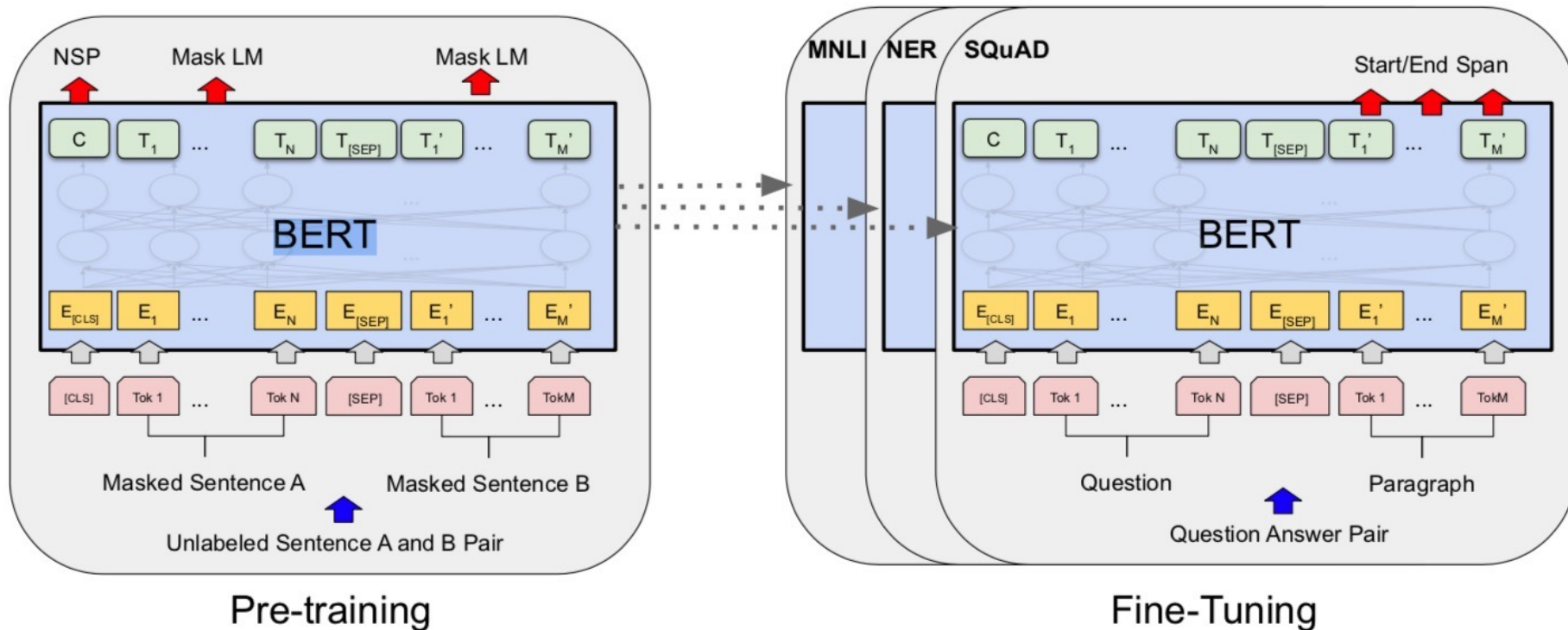
Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

# 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



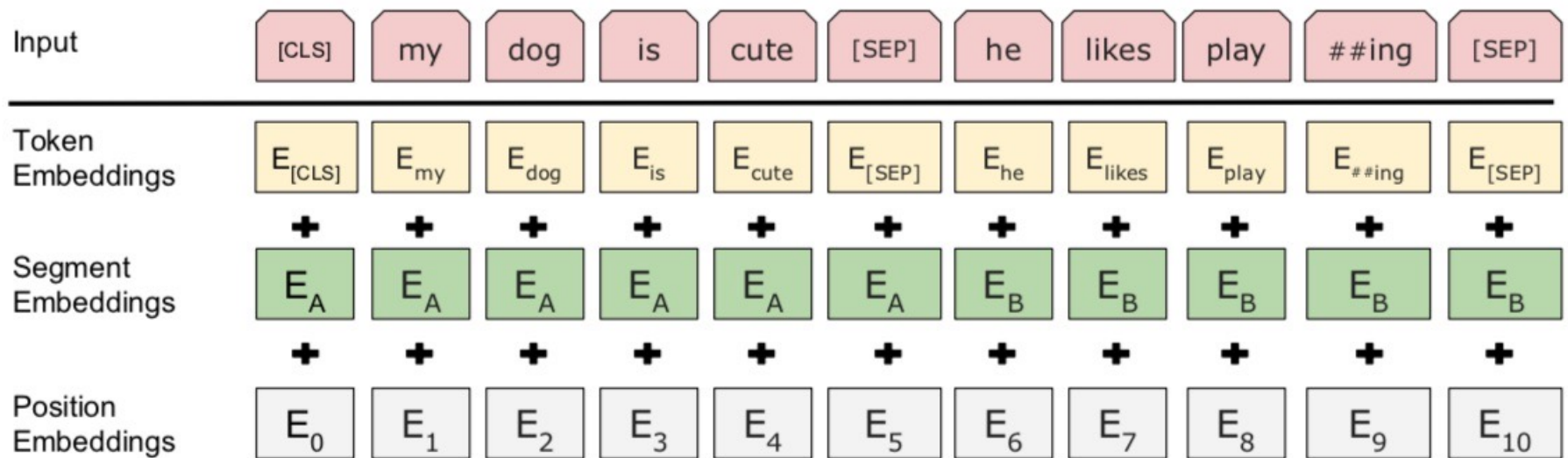
Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).

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

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

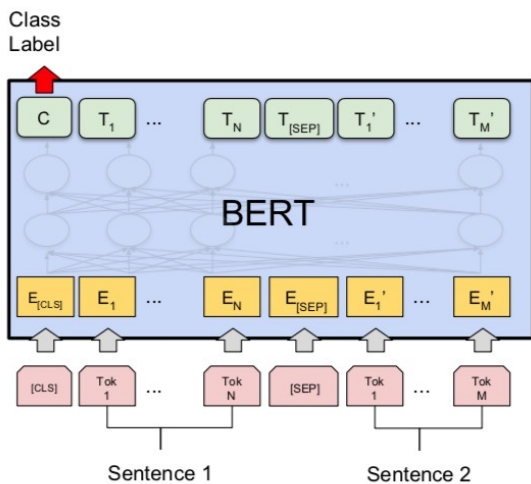
BERT (Bidirectional Encoder Representations from Transformers)

## BERT input representation

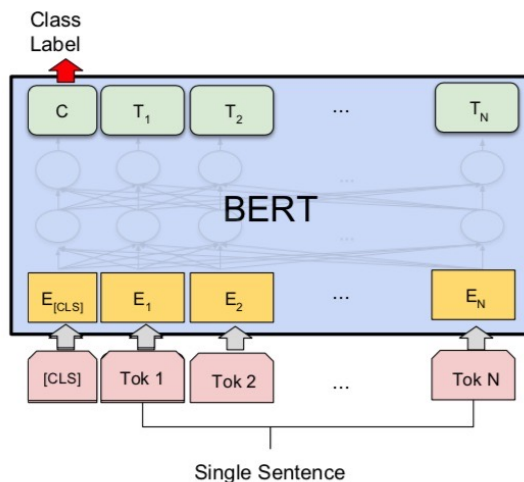


The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

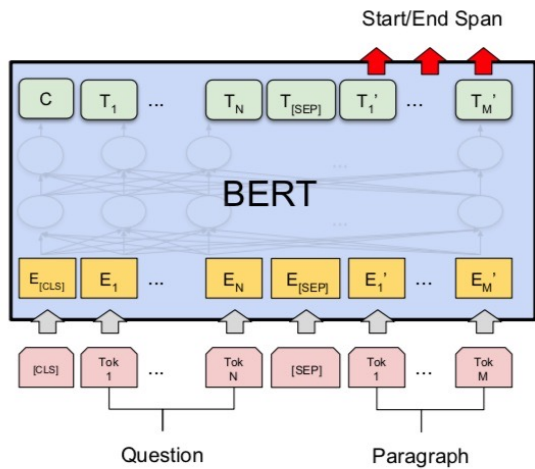
# Fine-tuning BERT on Different Tasks



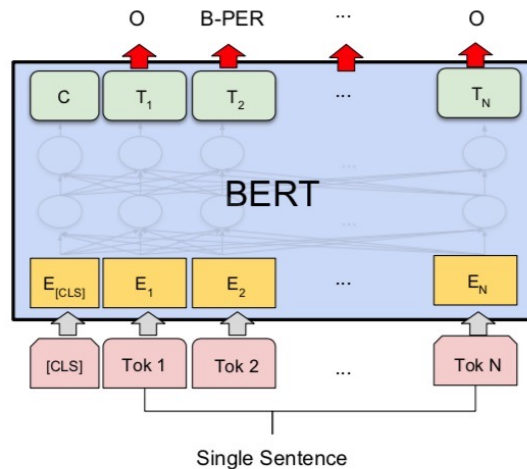
(a) Sentence Pair Classification Tasks:  
MNLI, QQP, QNLI, STS-B, MRPC,  
RTE, SWAG



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



(c) Question Answering Tasks:  
SQuAD v1.1

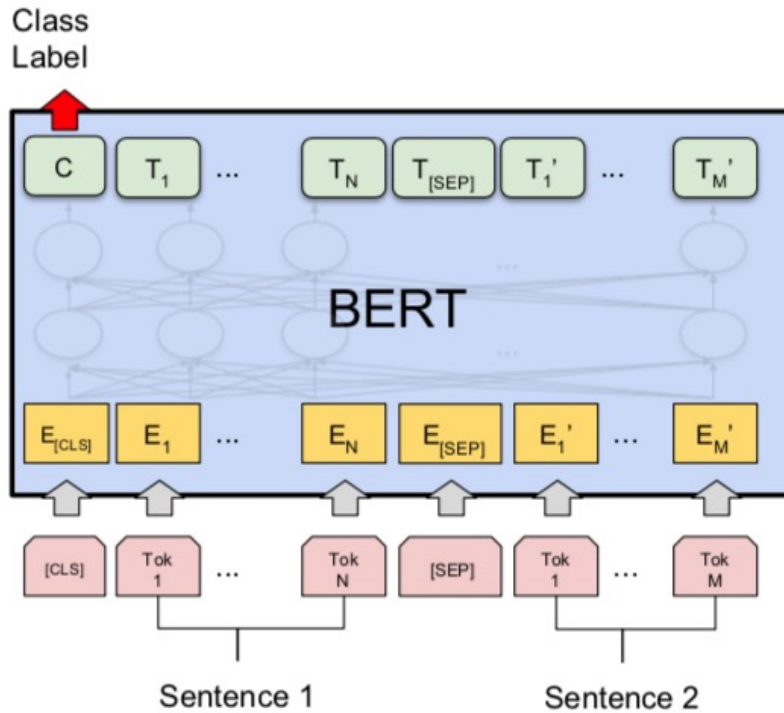


(d) Single Sentence Tagging Tasks:  
CoNLL-2003 NER

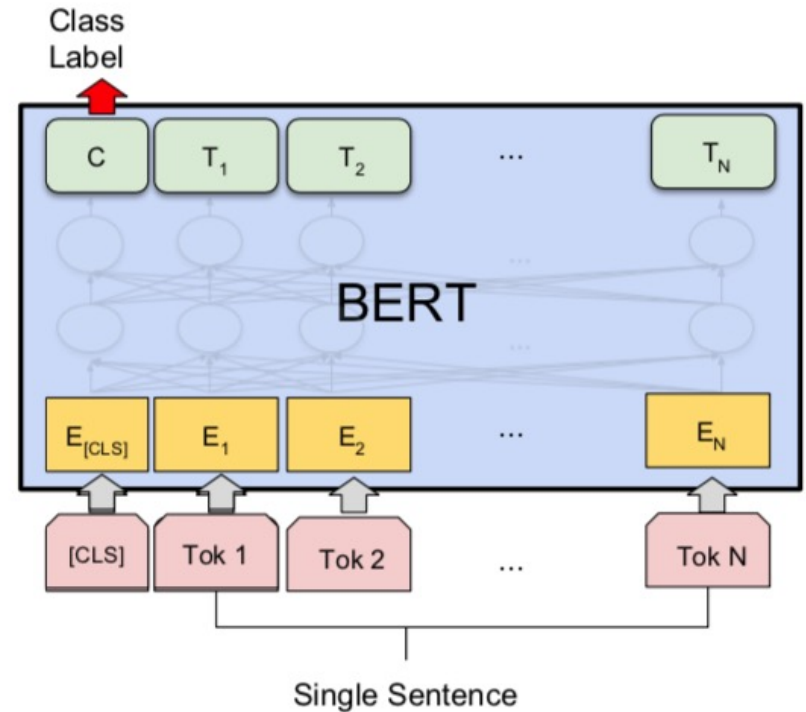
Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).

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

# BERT Sequence-level tasks



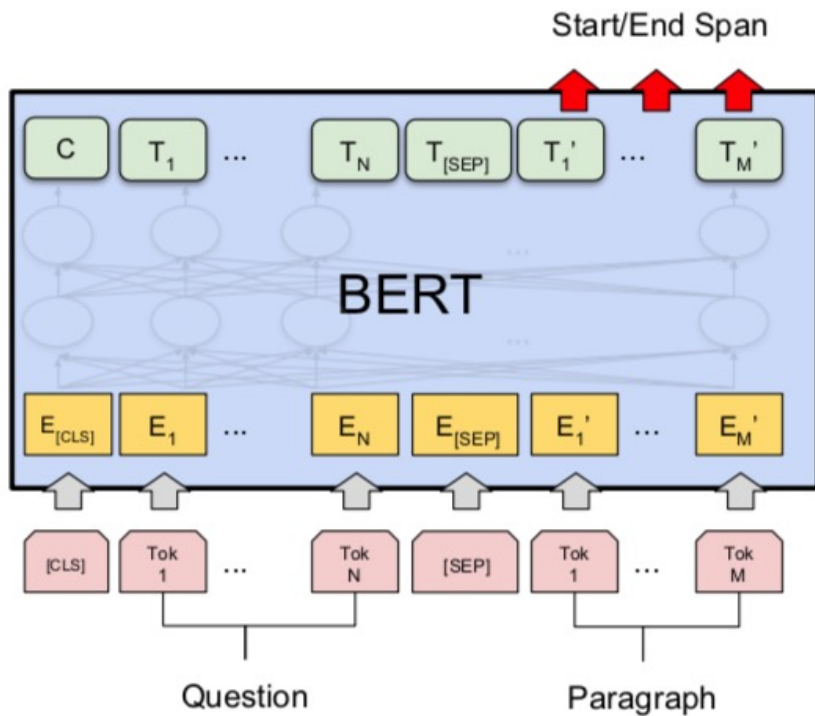
(a) Sentence Pair Classification Tasks:  
MNLI, QQP, QNLI, STS-B, MRPC,  
RTE, SWAG



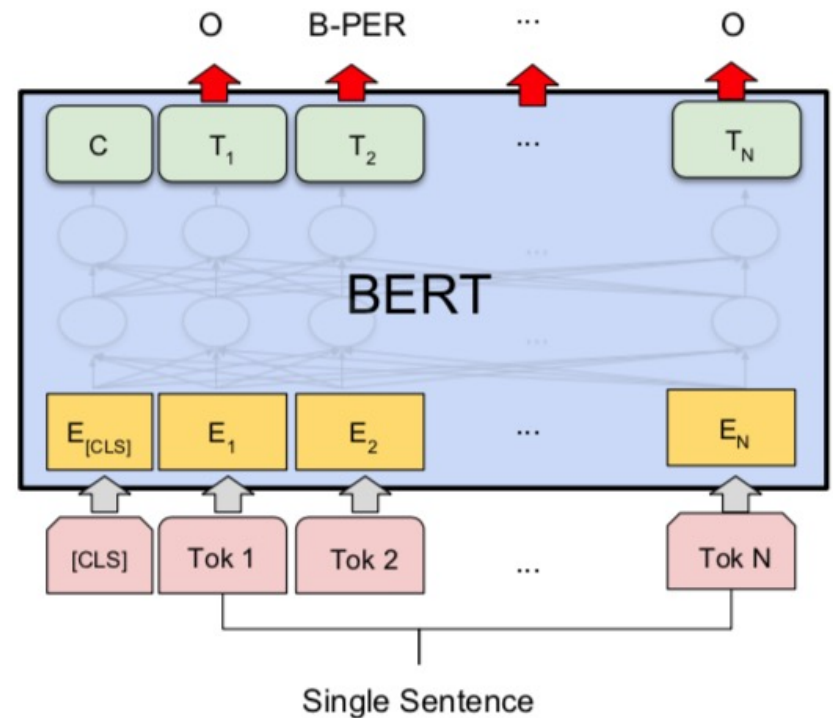
(b) Single Sentence Classification Tasks:  
SST-2, CoLA



# BERT Token-level tasks

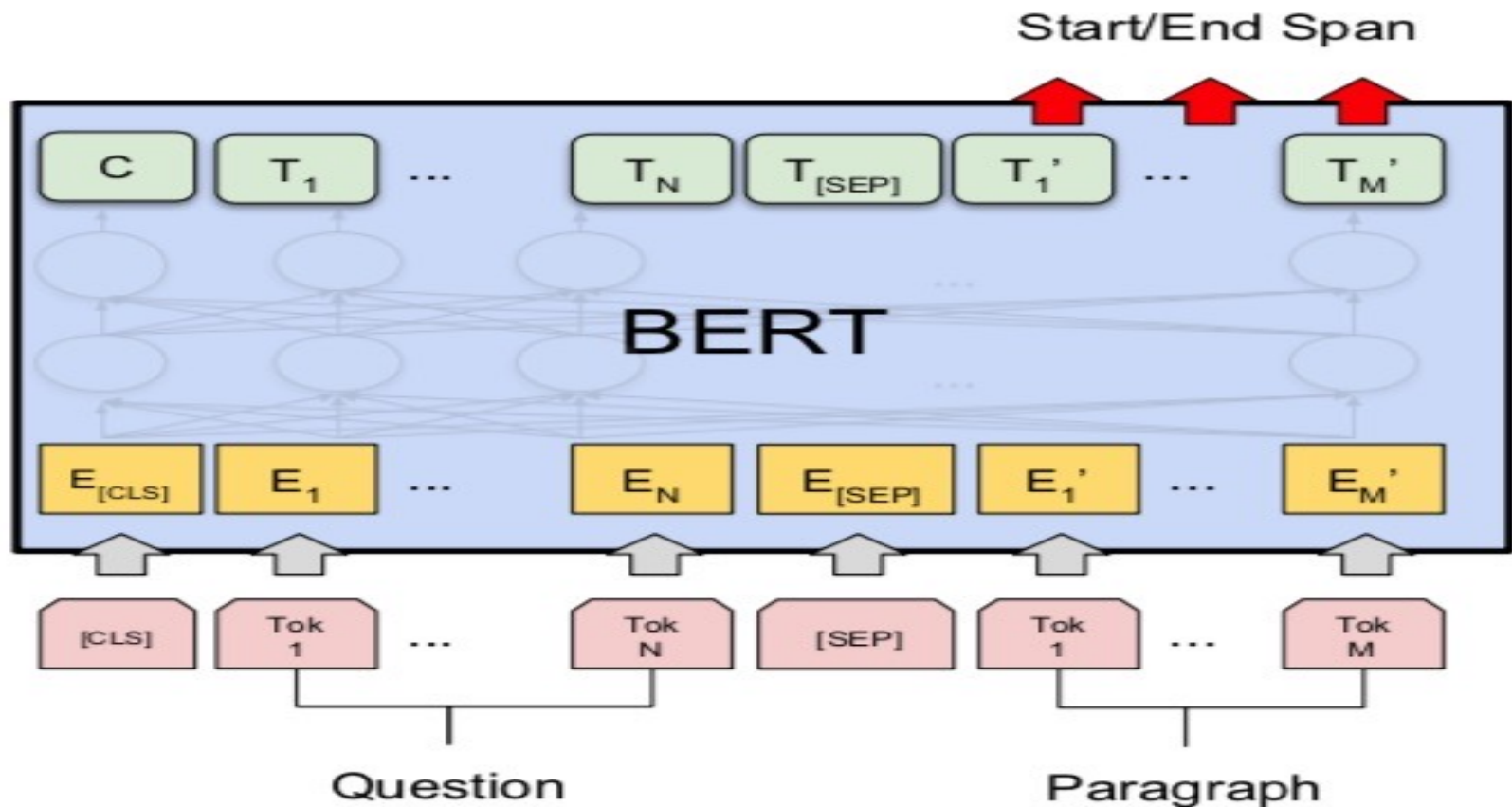


(c) Question Answering Tasks:  
SQuAD v1.1



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

# Fine-tuning BERT on Question Answering (QA)

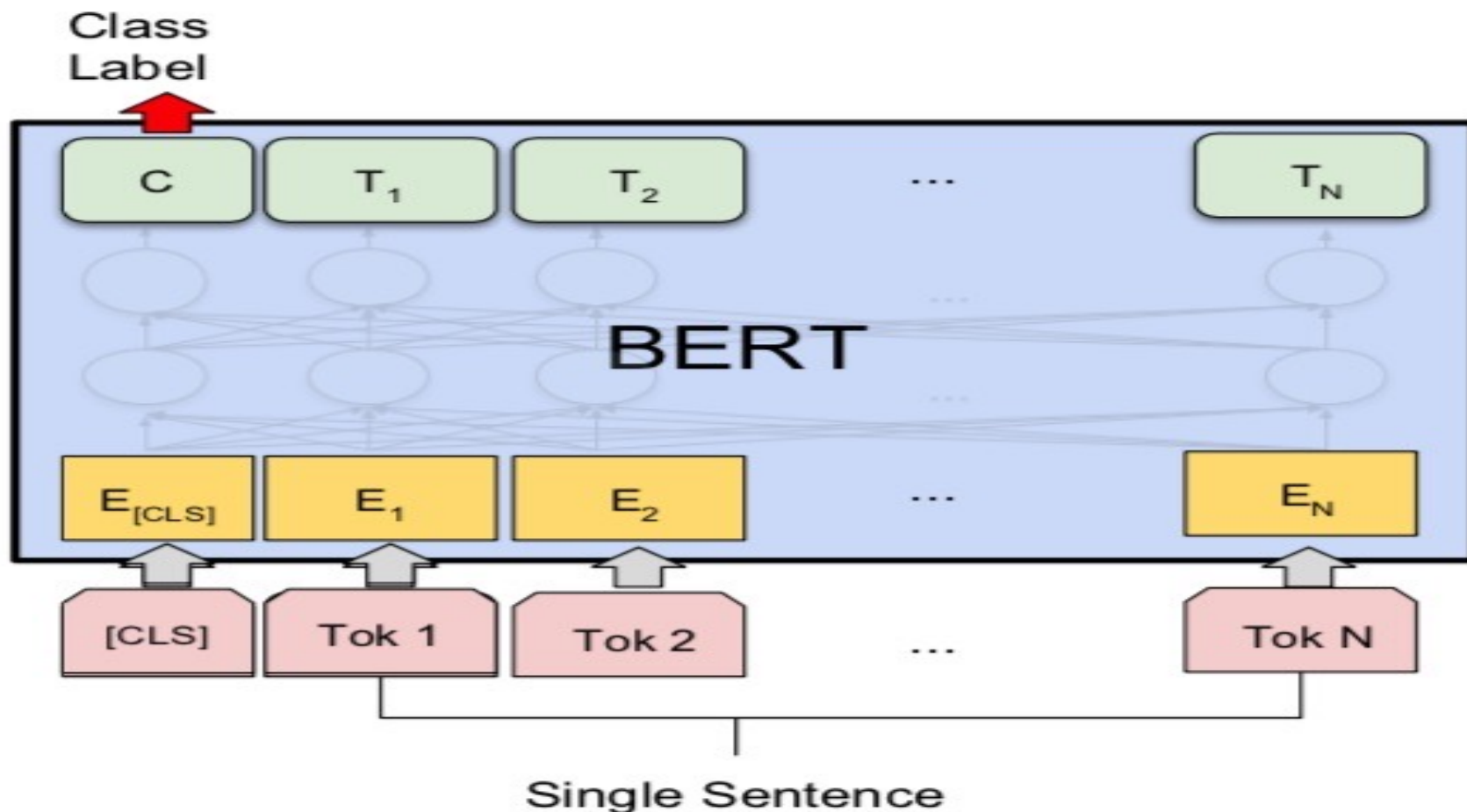


(c) Question Answering Tasks:  
SQuAD v1.1



# Fine-tuning BERT on Dialogue

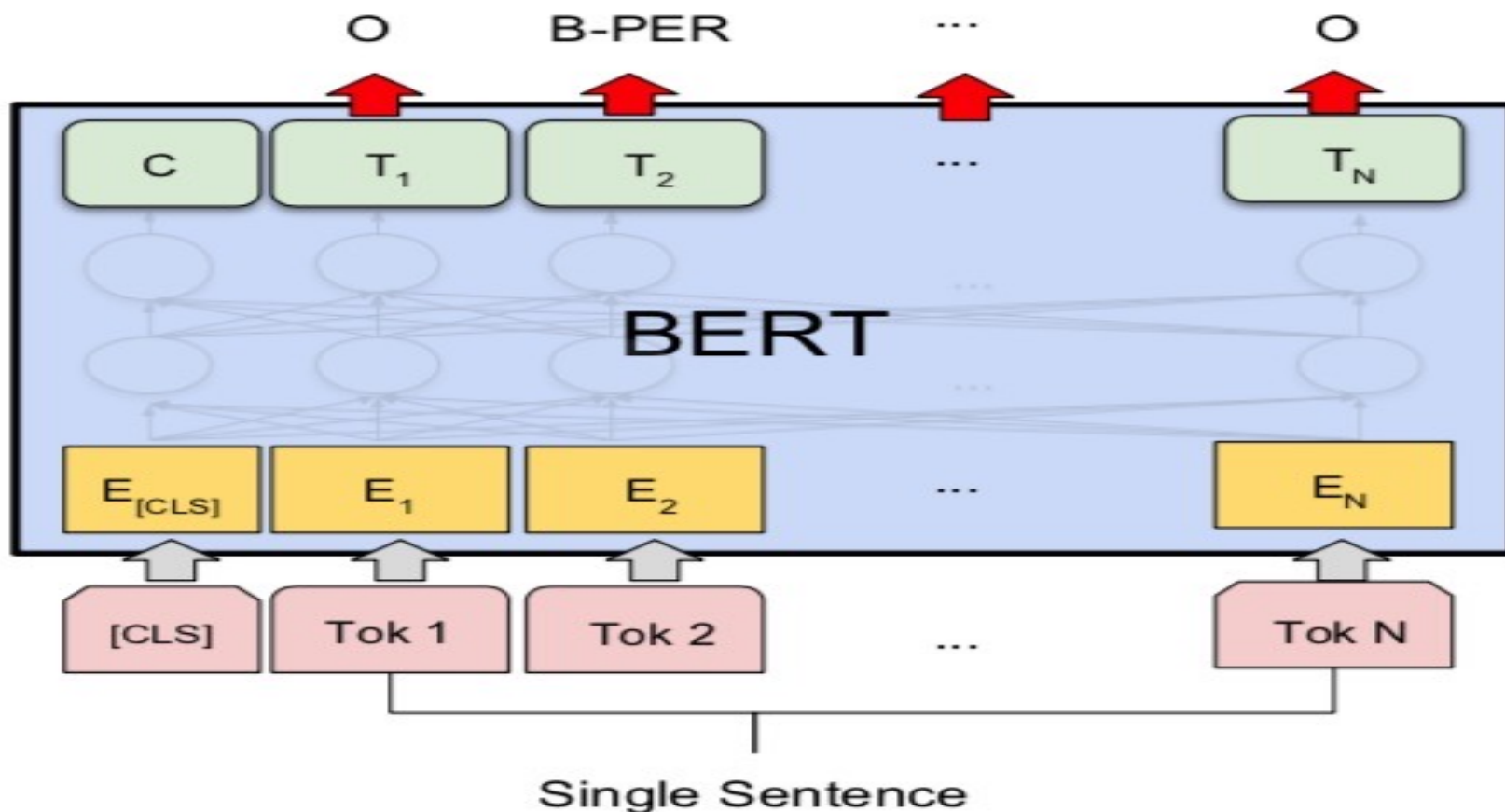
## Intent Detection (ID; Classification)



(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 <sub>BASE</sub>	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
BERT <sub>LARGE</sub>	<b>86.7/85.9</b>	<b>72.1</b>	<b>91.1</b>	<b>94.9</b>	<b>60.5</b>	<b>86.5</b>	<b>89.3</b>	<b>70.1</b>	<b>81.9</b>

**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

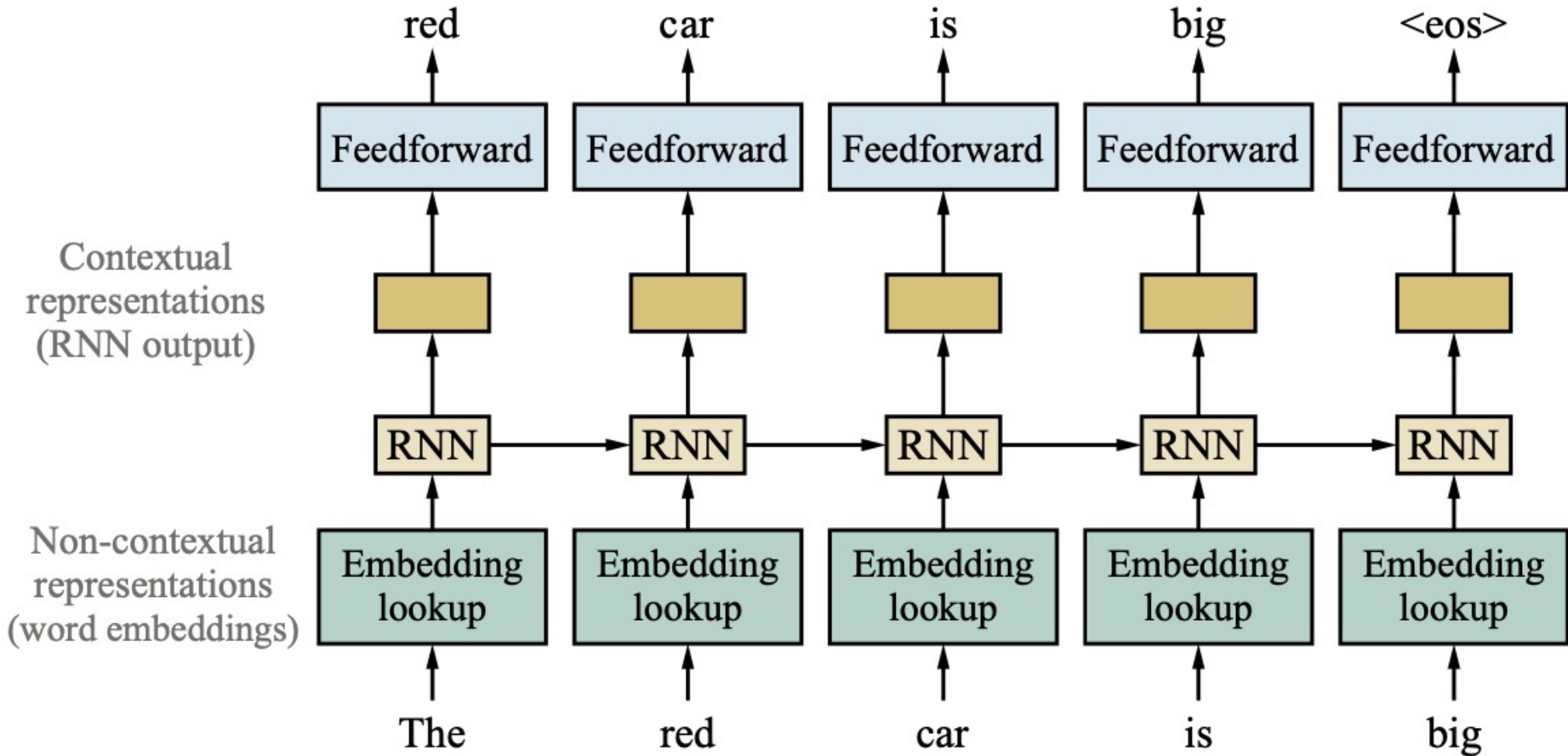
**RTE:** Recognizing Textual Entailment

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

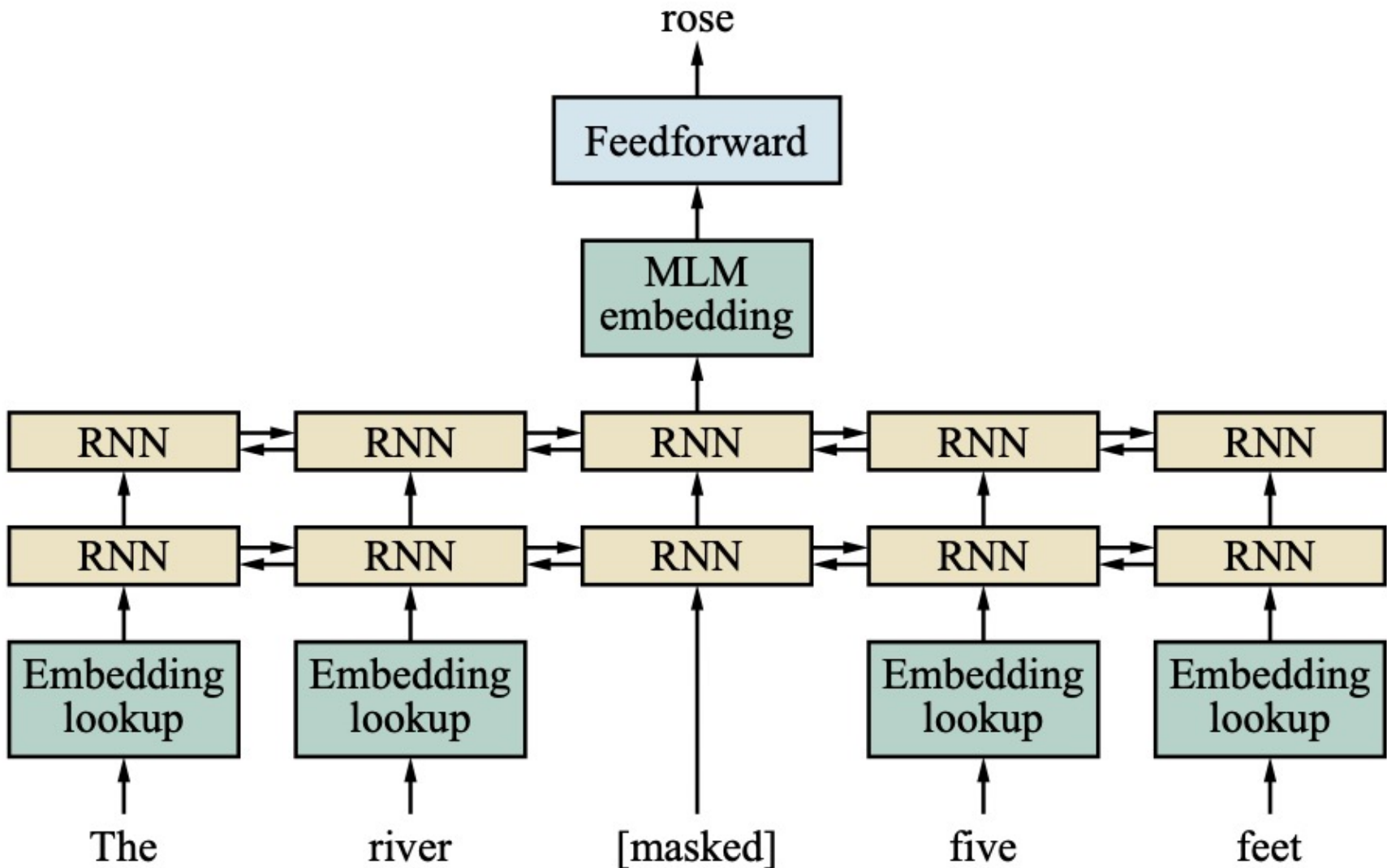
"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

# Training Contextual Representations

## using a left-to-right Language Model



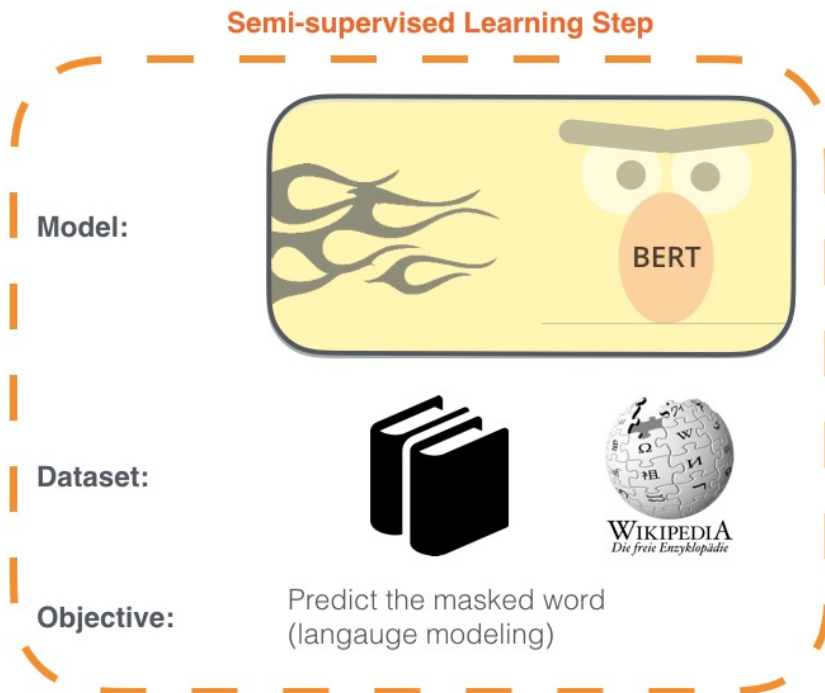
# Masked Language Modeling: Pretrain a Bidirectional Model



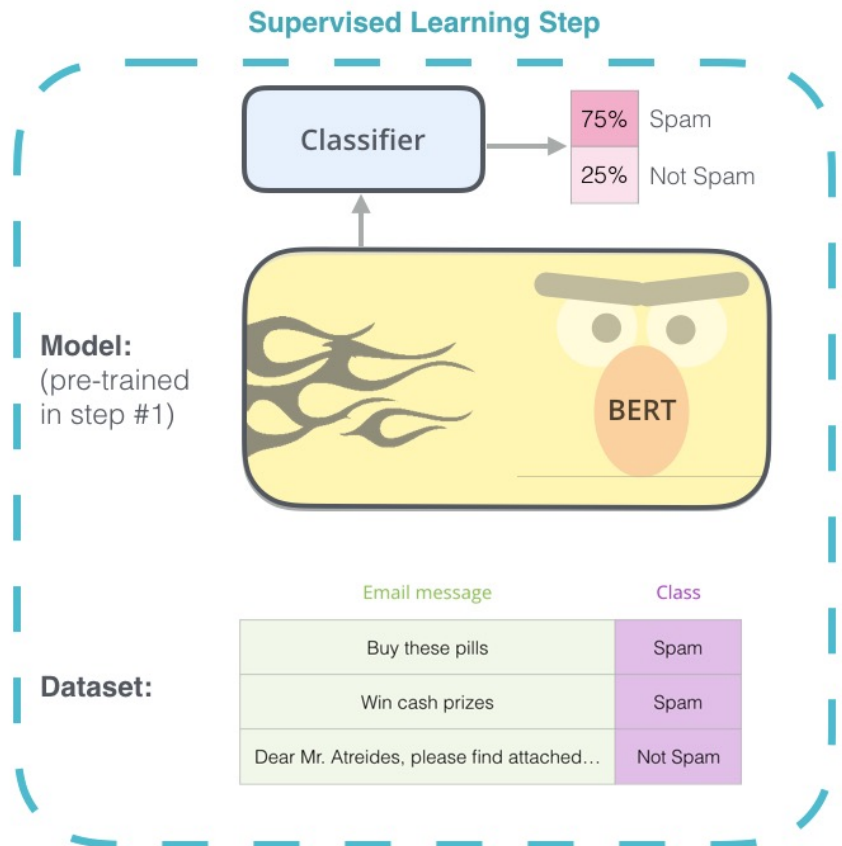
# Illustrated BERT

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

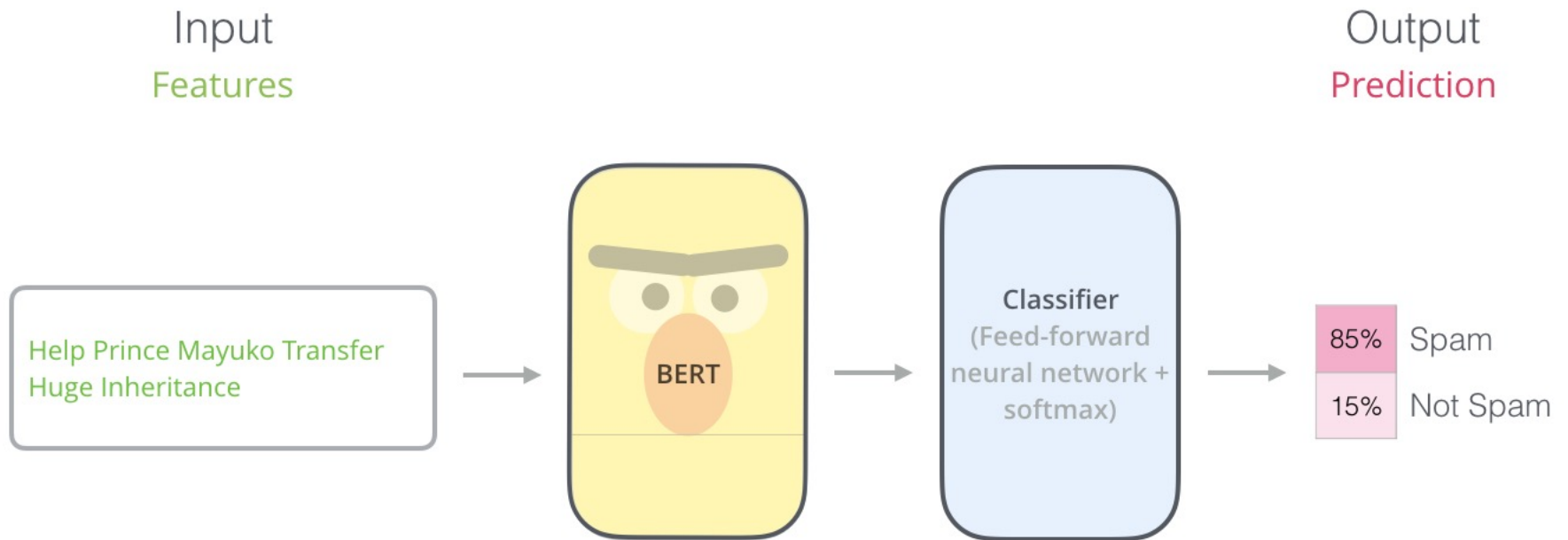
The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.



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

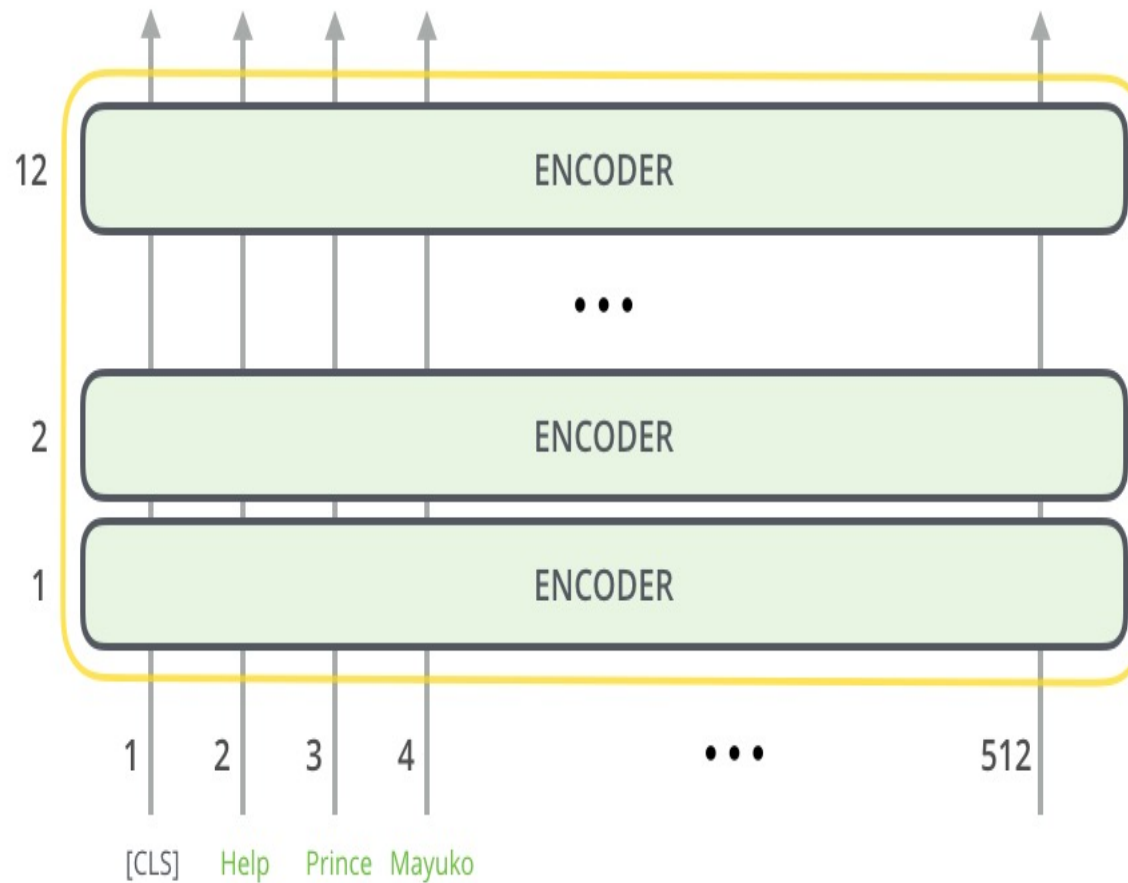


# BERT Classification Input Output





# BERT Encoder Input

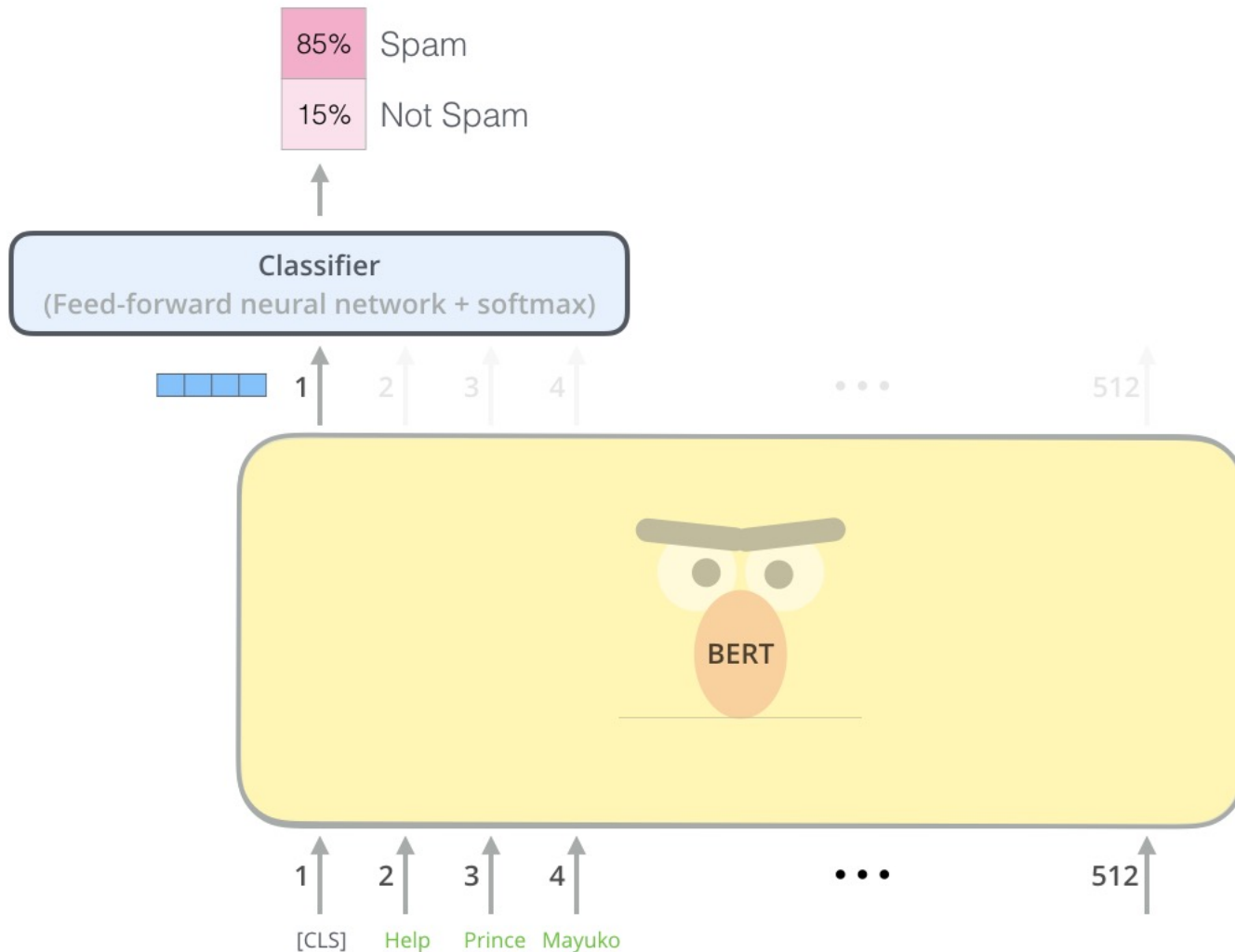


BERT

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

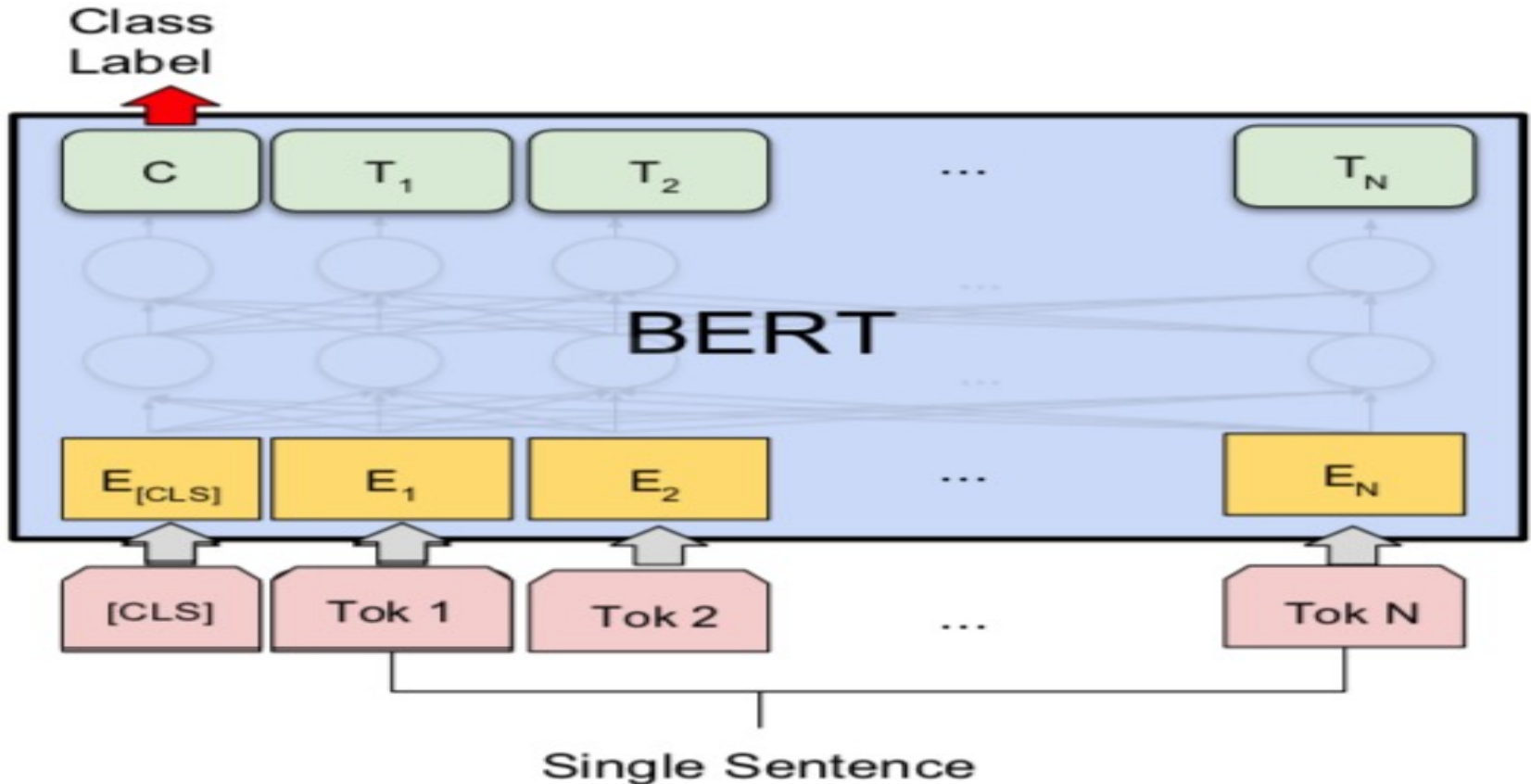


# BERT Classifier



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

# Sentiment Analysis: Single Sentence Classification



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

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

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

# A Visual Guide to Using BERT for the First Time

(Jay Alammar, 2019)

“a visually stunning  
ruminatiion on love”

Reviewer #1

That’s a **positive** thing to say



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

Reviewer #2

That’s **negative**

# Sentiment Classification: SST2

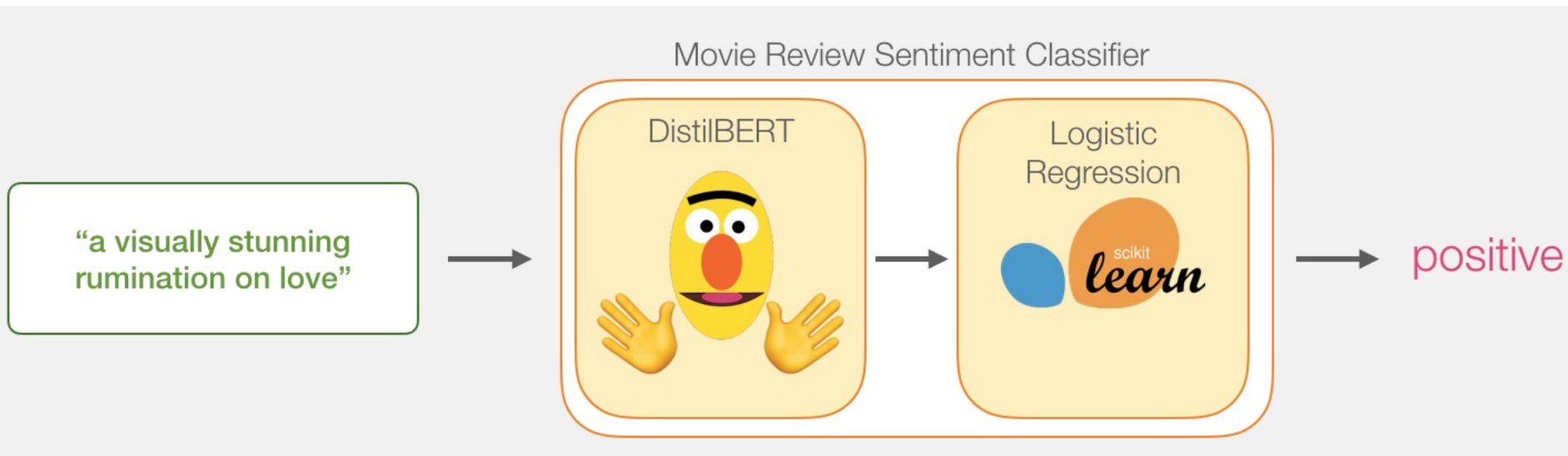
## Sentences from movie reviews

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

# Movie Review Sentiment Classifier



# Movie Review Sentiment Classifier



# Movie Review Sentiment Classifier

## Model Training

Movie Review Sentiment Classifier

DistilBERT

Already (pre-)trained



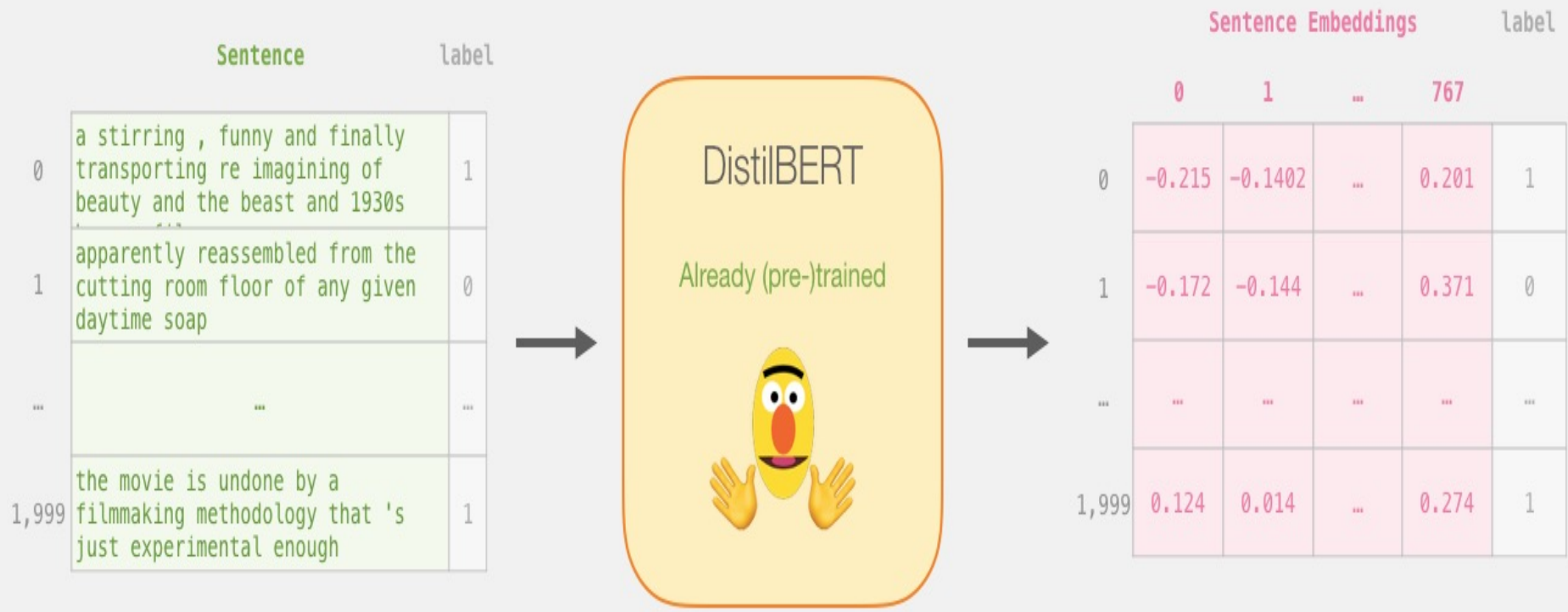
Logistic  
Regression

We will train in this tutorial



# Step # 1 Use distilBERT to Generate Sentence Embeddings

Step #1: Use DistilBERT to embed all the sentences





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

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



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

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

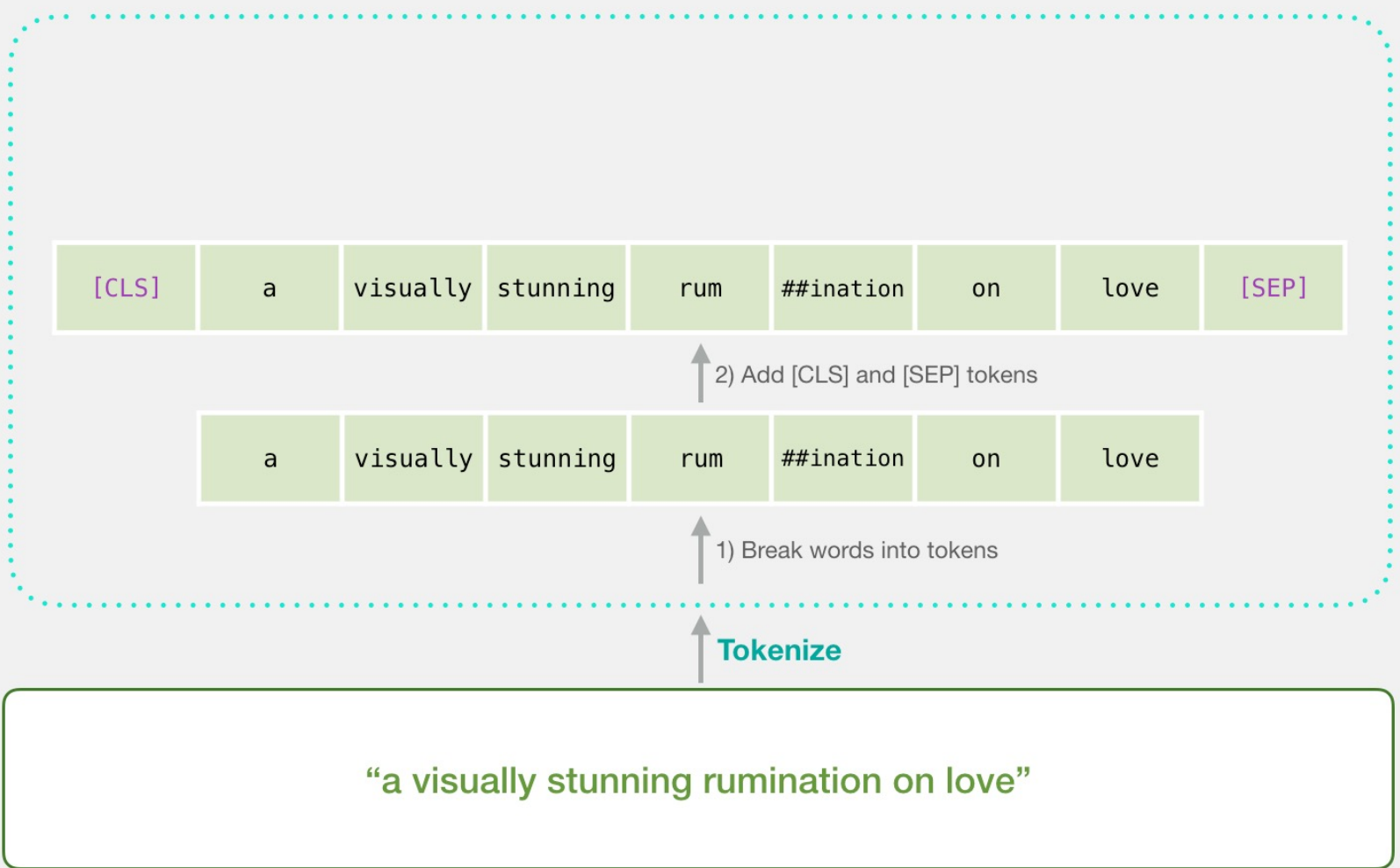
	Sentence Embeddings				label
	0	1	...	767	
0	-0.215	-0.1402	...	0.201	1
1					
2					
3					
4					
5					
6					
...					
1,499					



# Tokenization

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

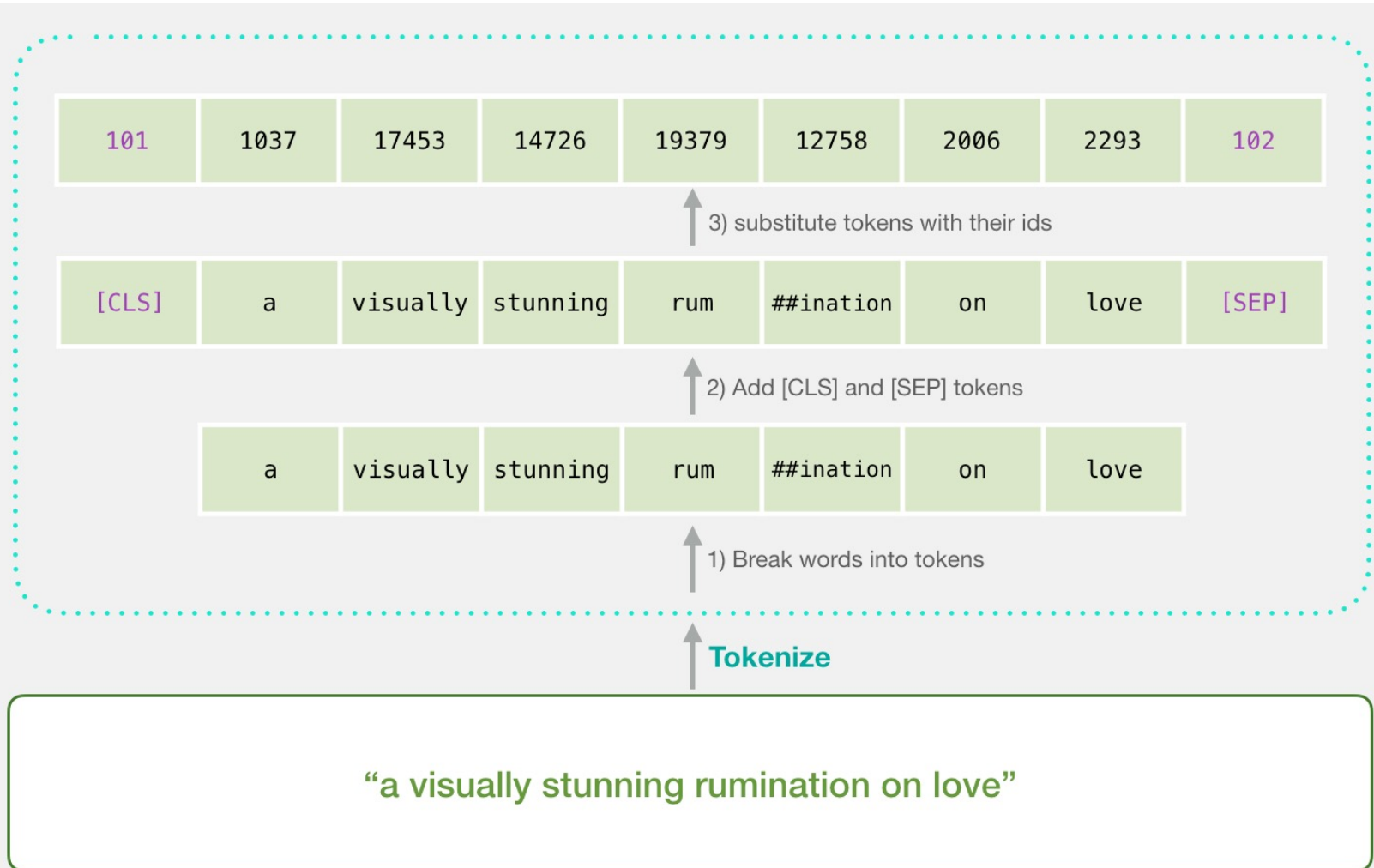
Tokenization  
DistilBertTokenizer



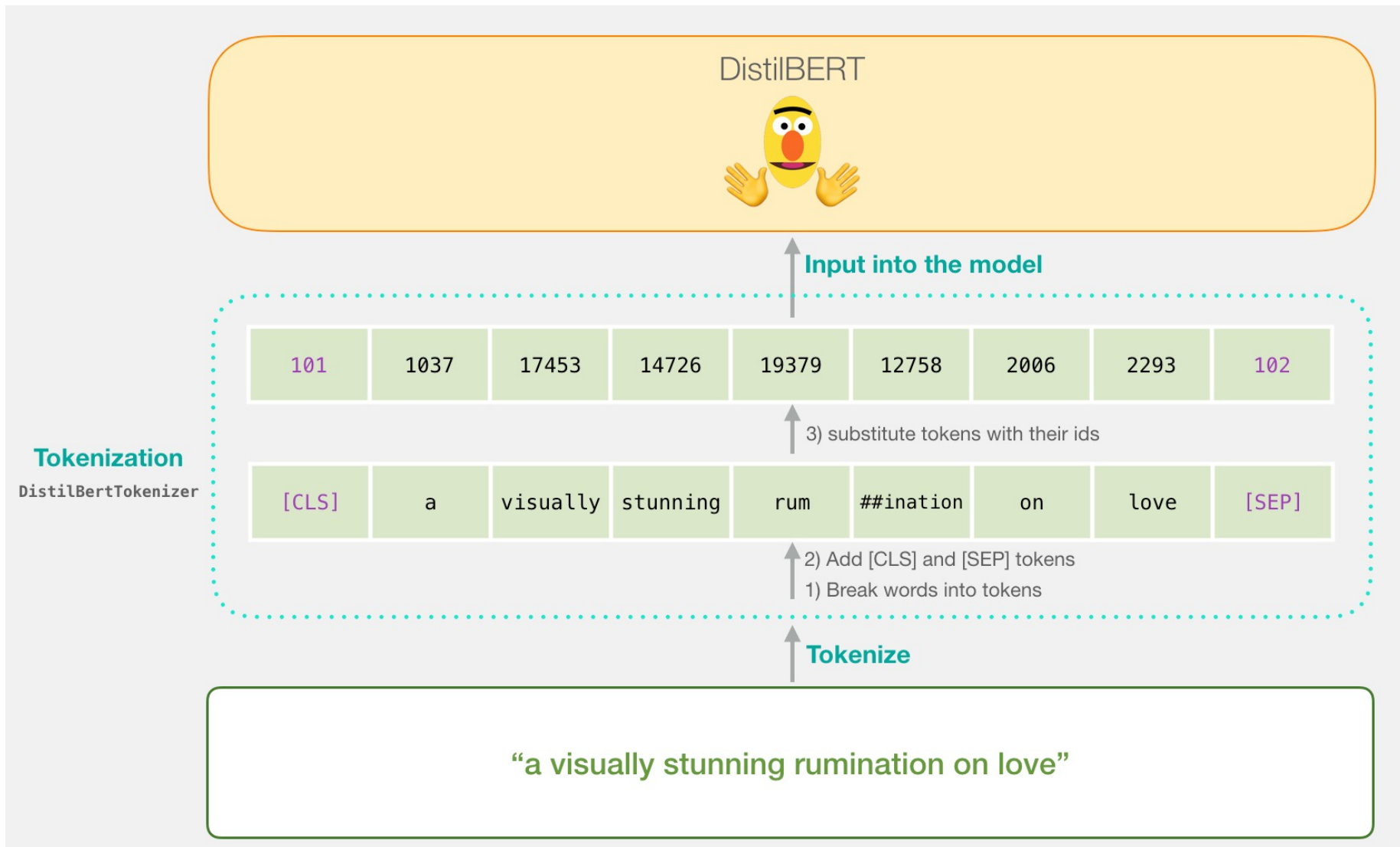
# Tokenization

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

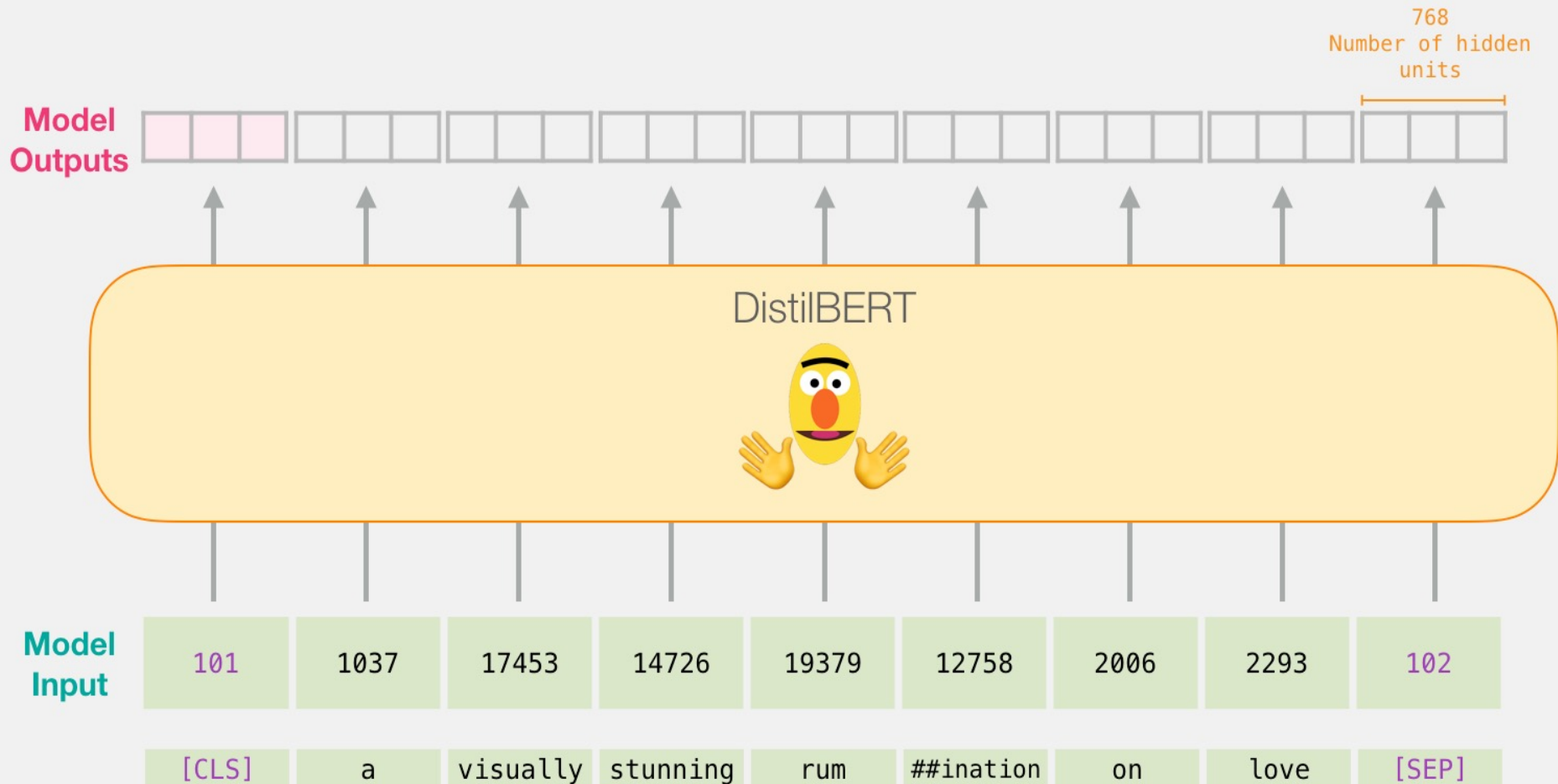
Tokenization  
DistilBertTokenizer



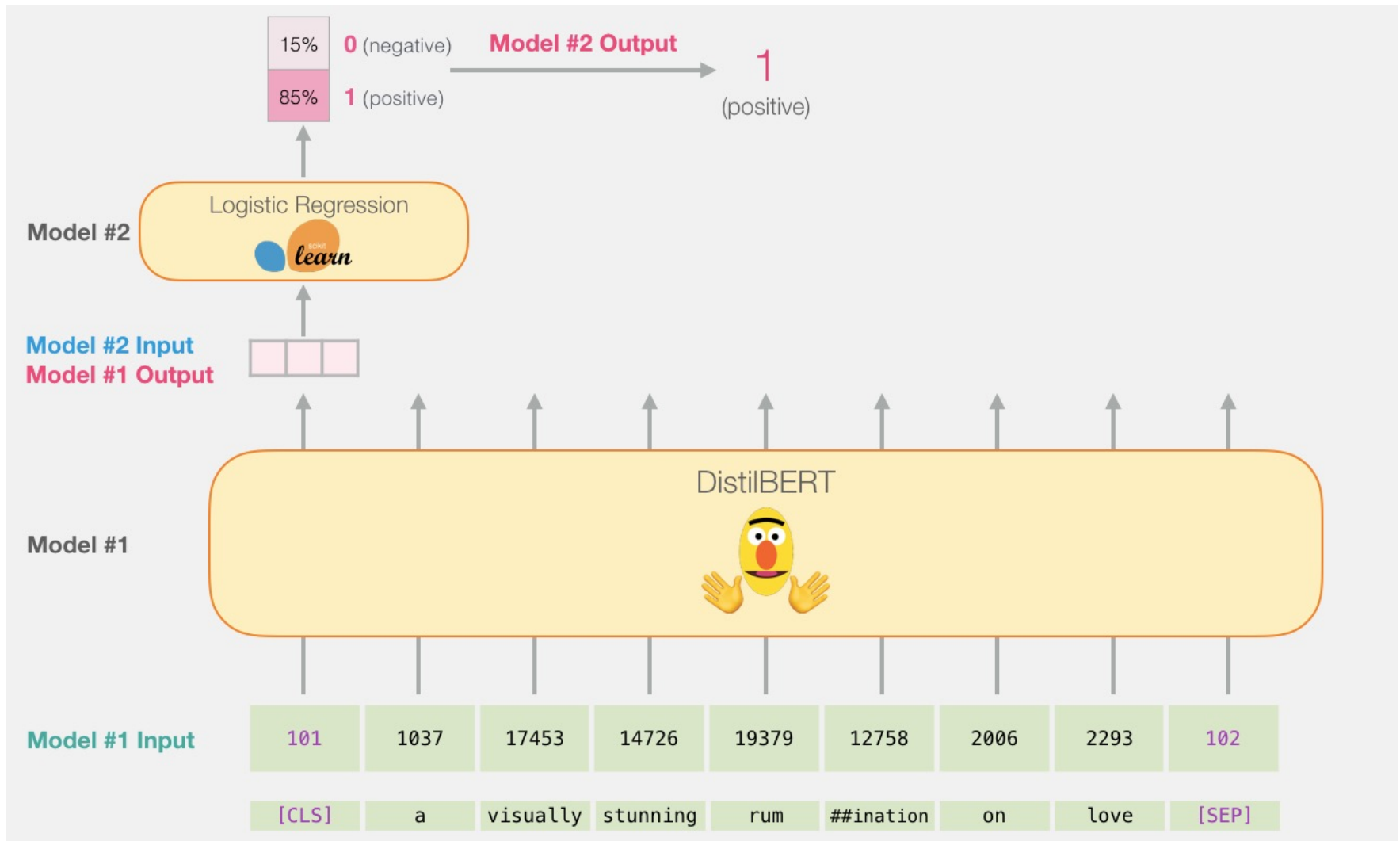
# Tokenization for BERT Model



# Flowing Through DistilBERT (768 features)

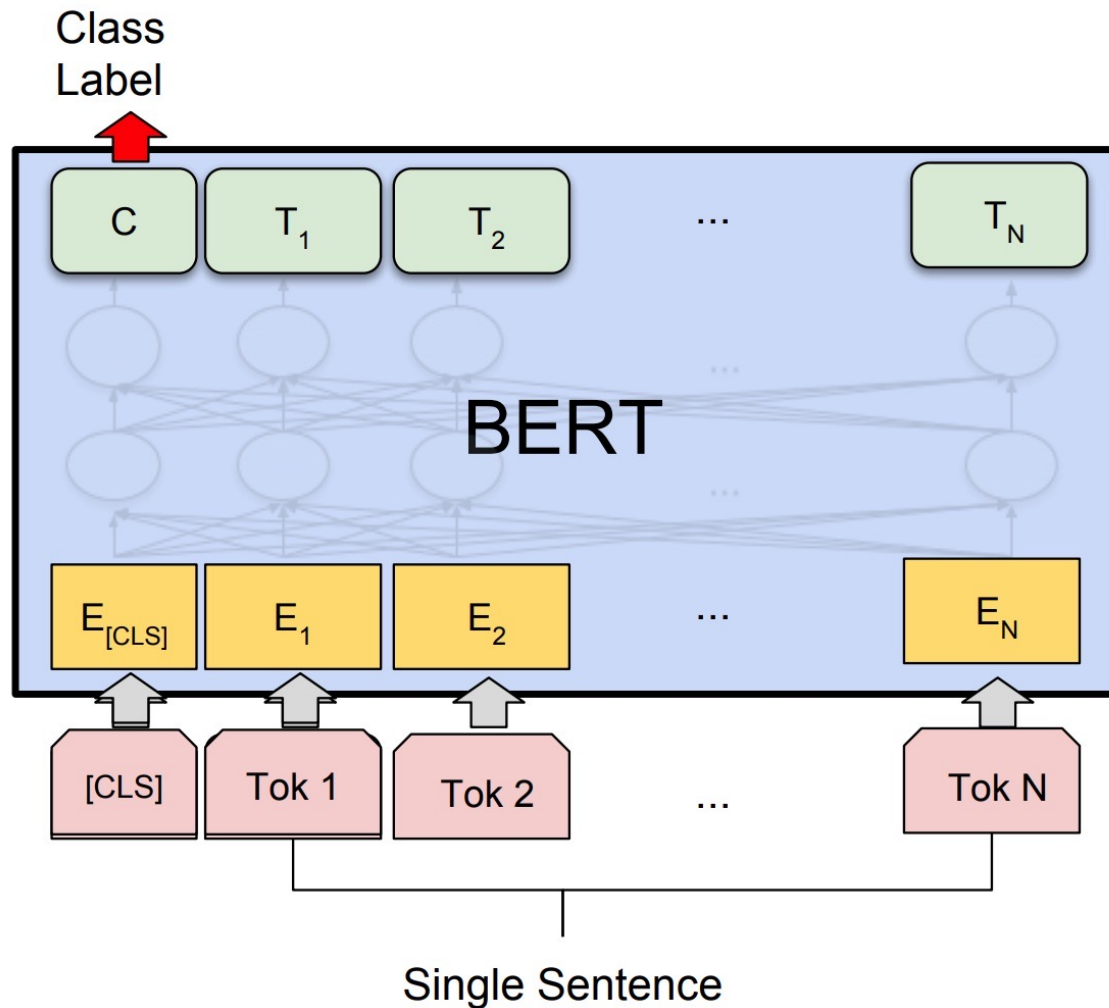


# Model #1 Output Class vector as Model #2 Input



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

# Fine-tuning BERT on Single Sentence Classification Tasks

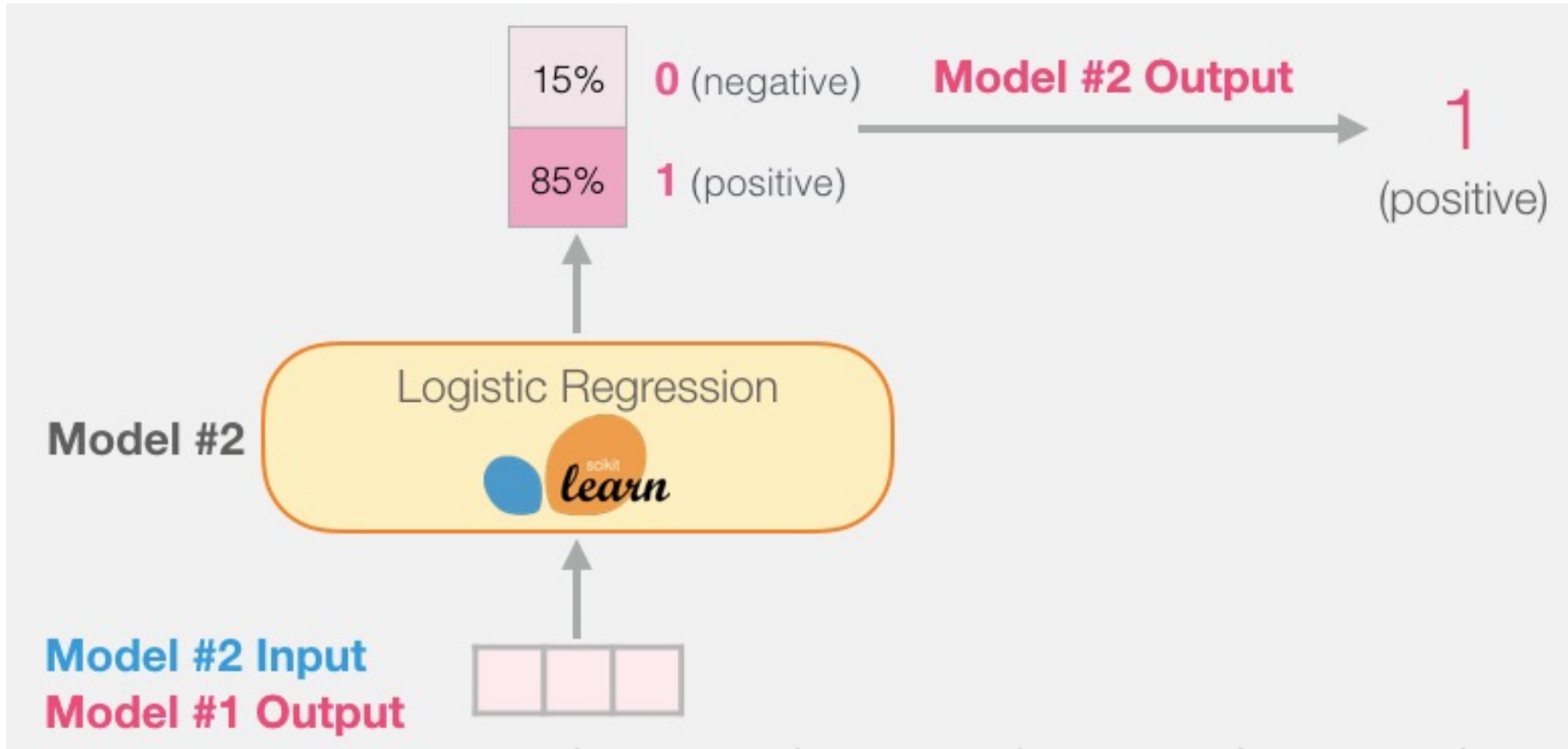


Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).

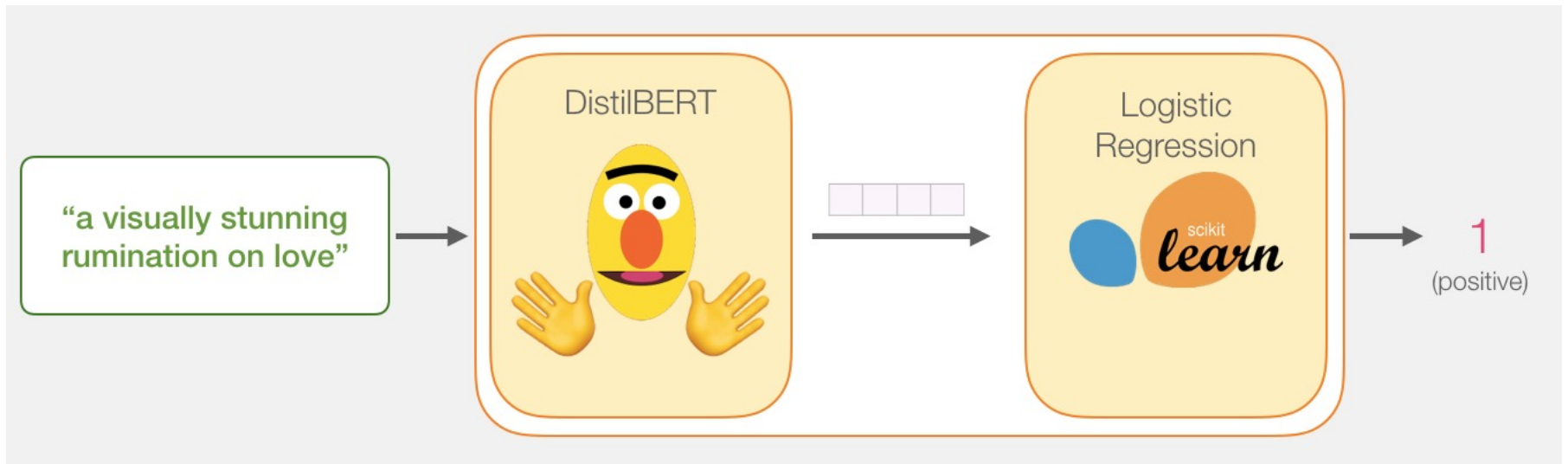
"Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.



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



# Logistic Regression Model to classify **Class** vector



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

```
df.head()
```

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

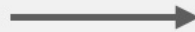
# Tokenization

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

Raw Dataset

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

Tokenize



Sequences of Token IDs

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

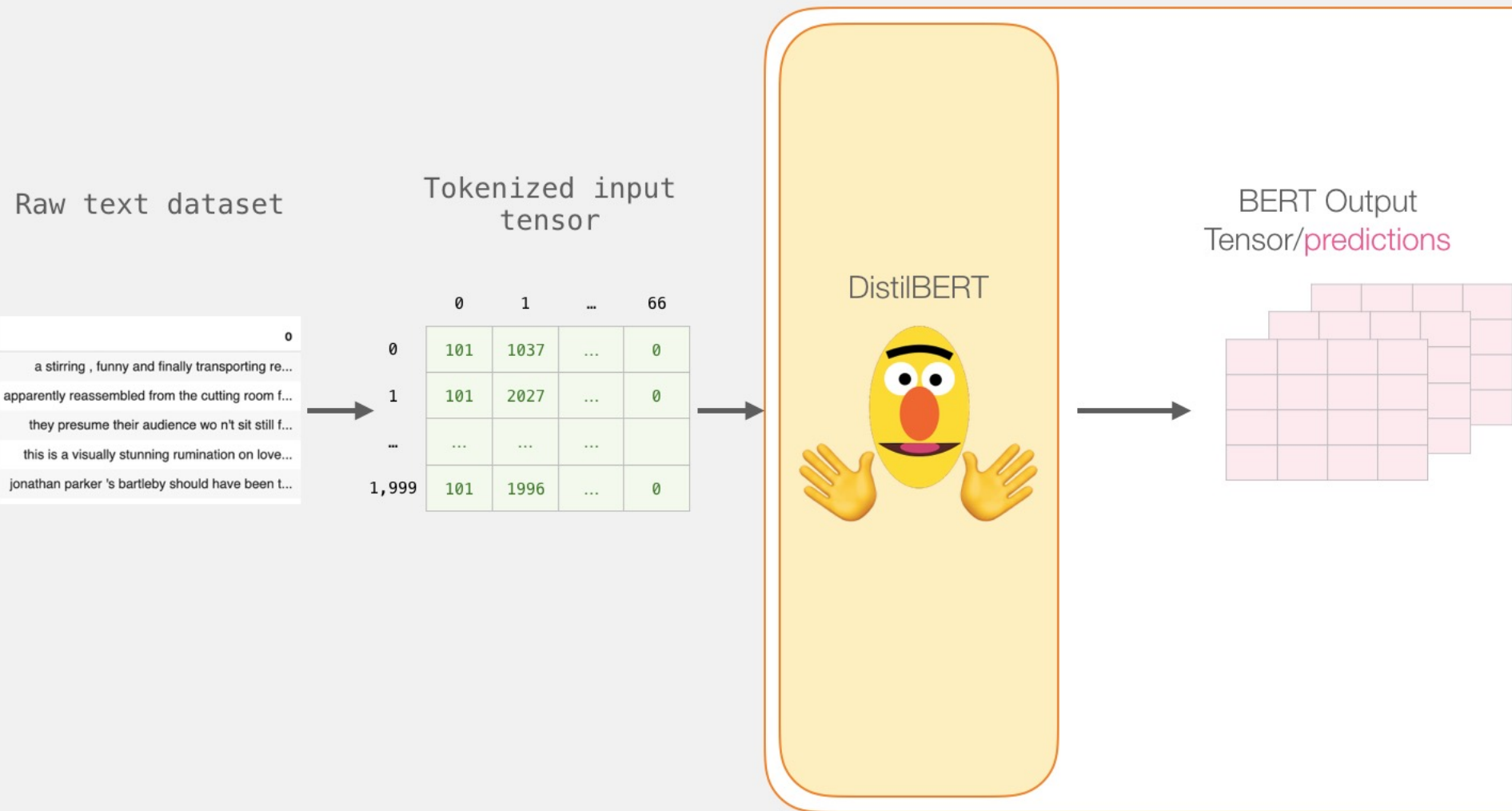
# BERT Input Tensor

## BERT/DistilBERT Input Tensor

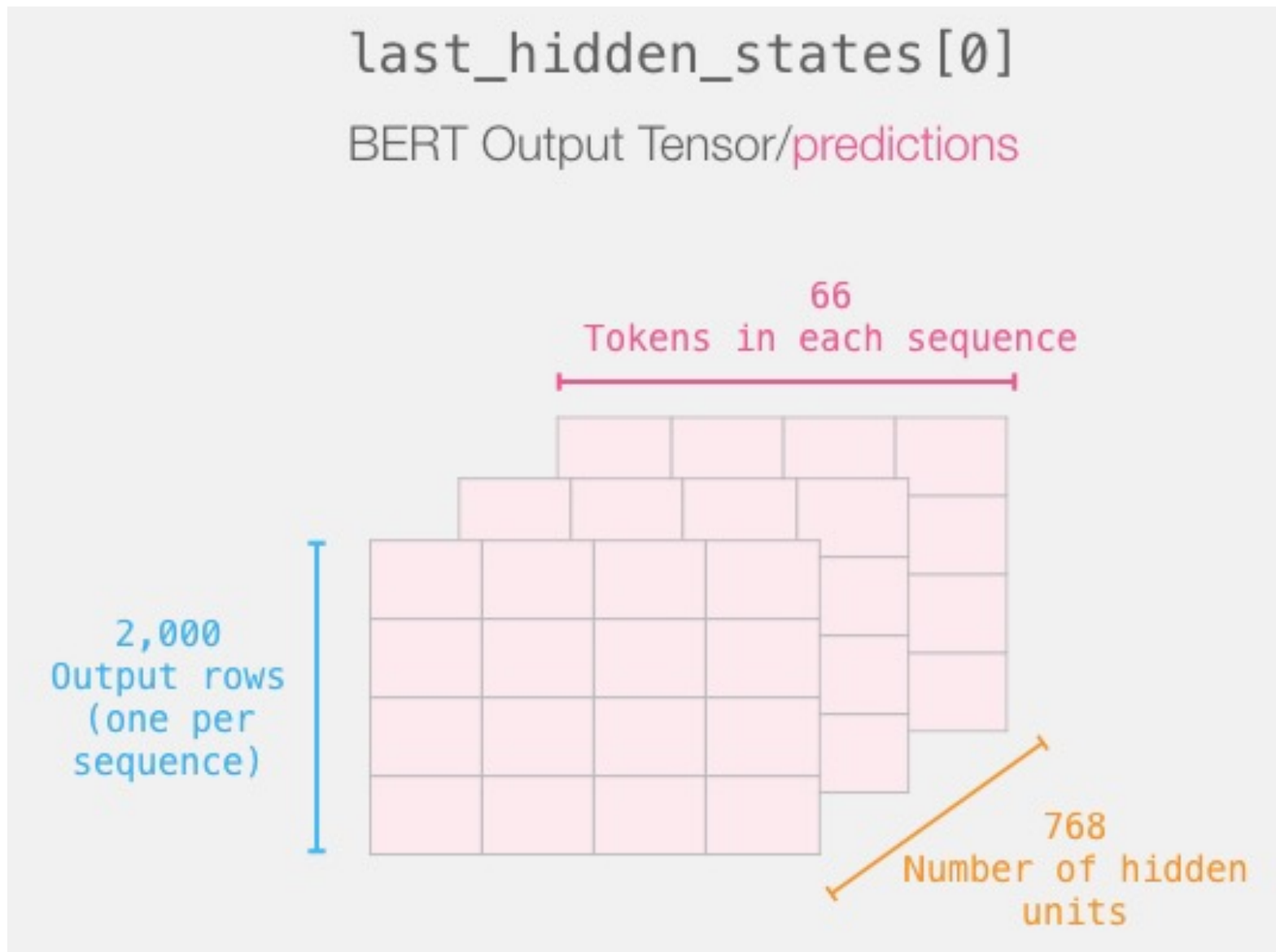


# Processing with DistilBERT

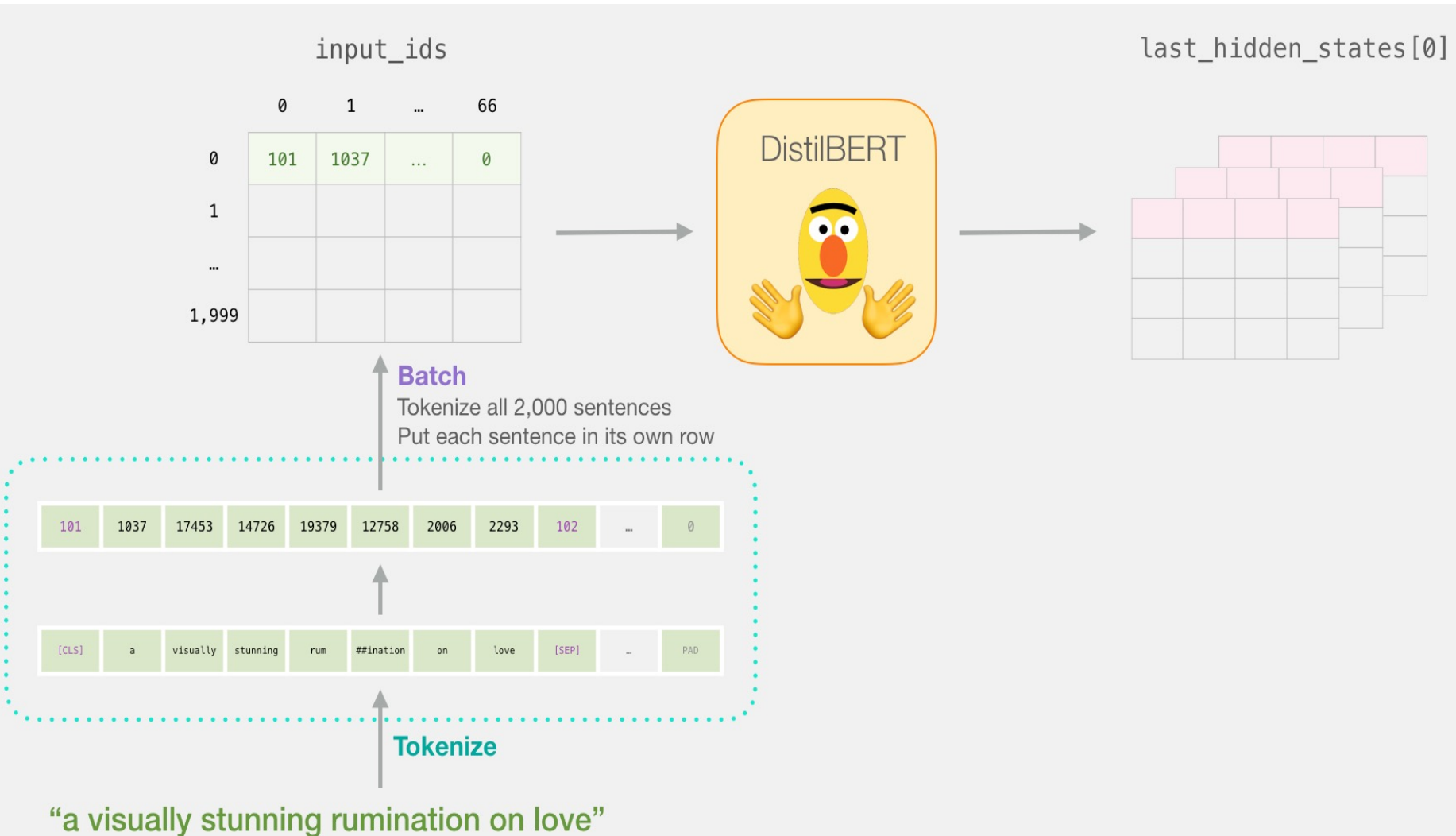
```
input_ids = torch.tensor(np.array(padded))  
last_hidden_states = model(input_ids)
```



# Unpacking the BERT output tensor



# Sentence to last\_hidden\_state[0]





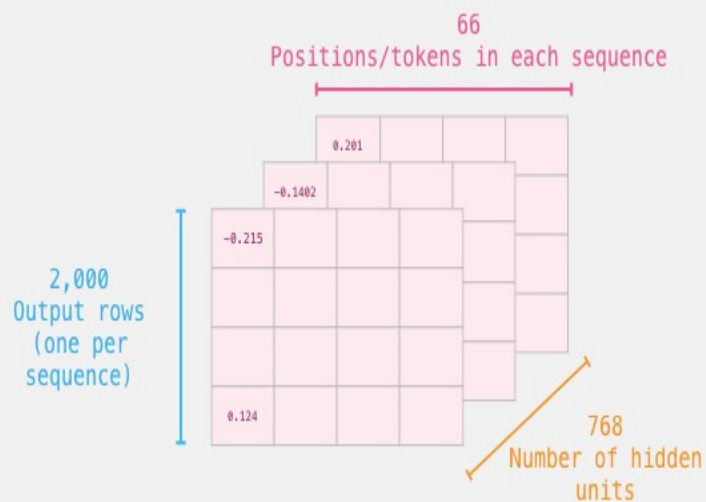
# BERT's output for the [CLS] tokens

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

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

`last_hidden_states[0]`

BERT Output Tensor/predictions

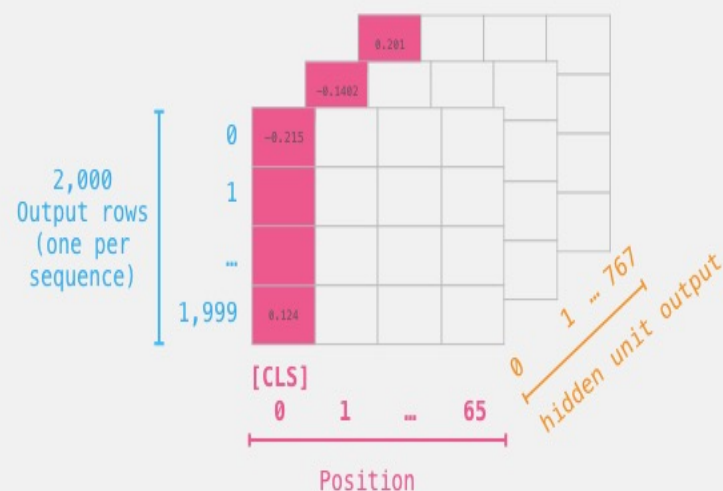


only the first position: [CLS]

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

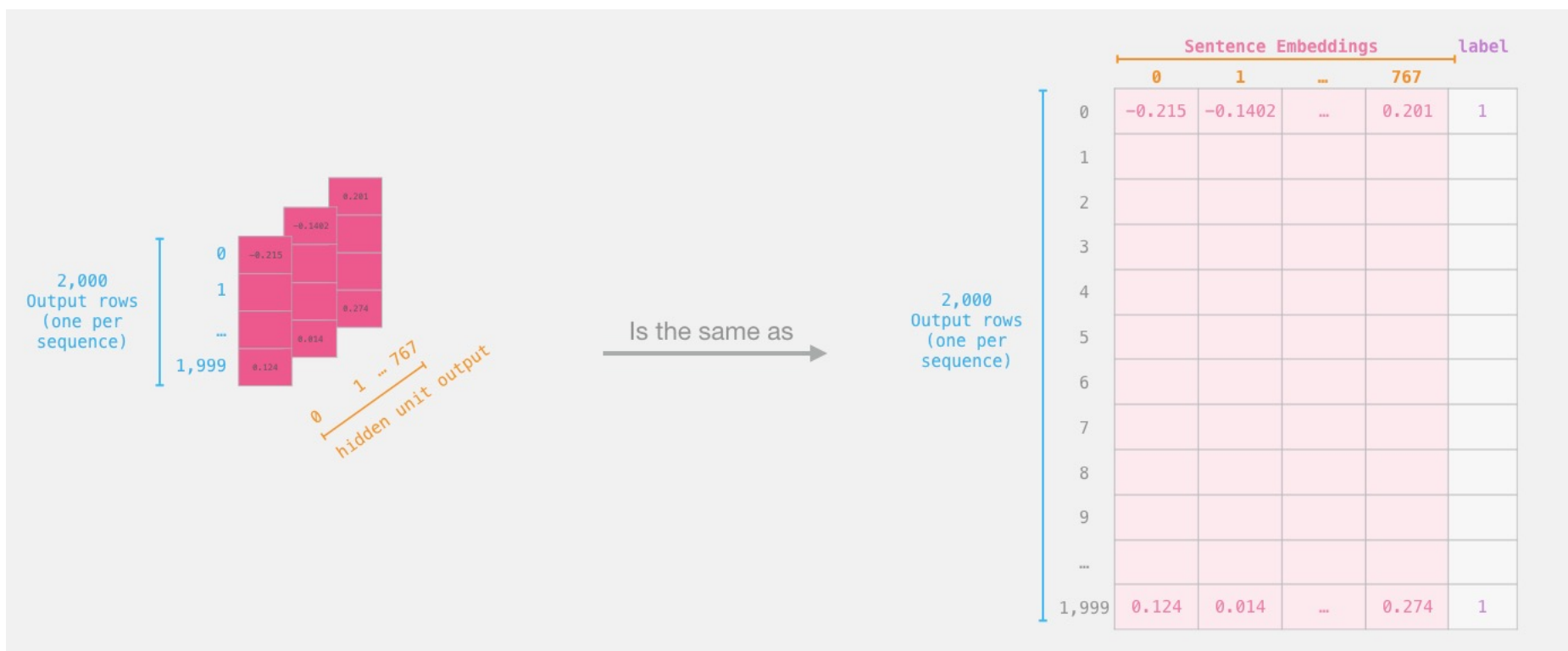
all sentences

all hidden unit outputs



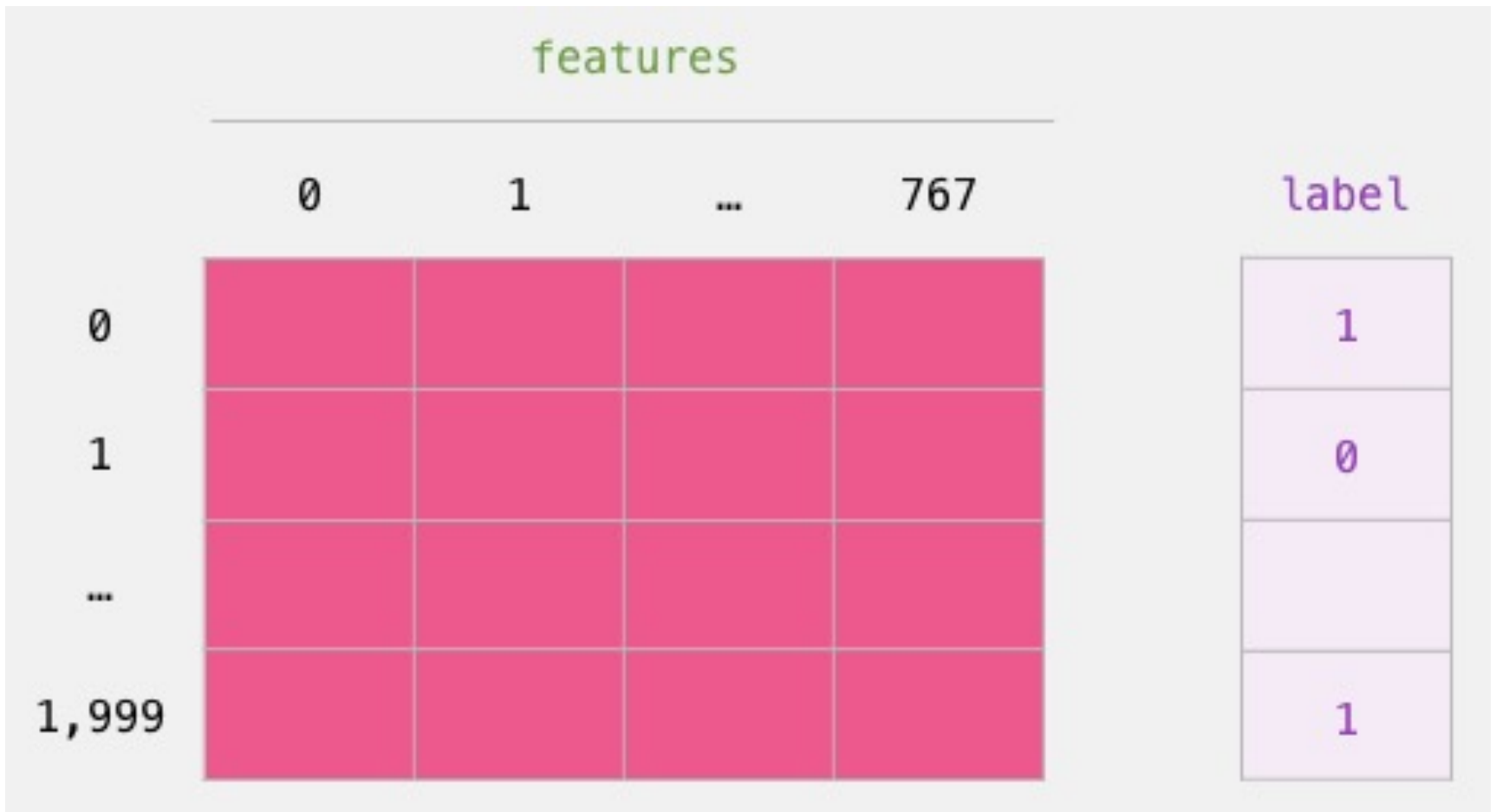
# The tensor sliced from BERT's output

## Sentence Embeddings



# Dataset for Logistic Regression (768 Features)

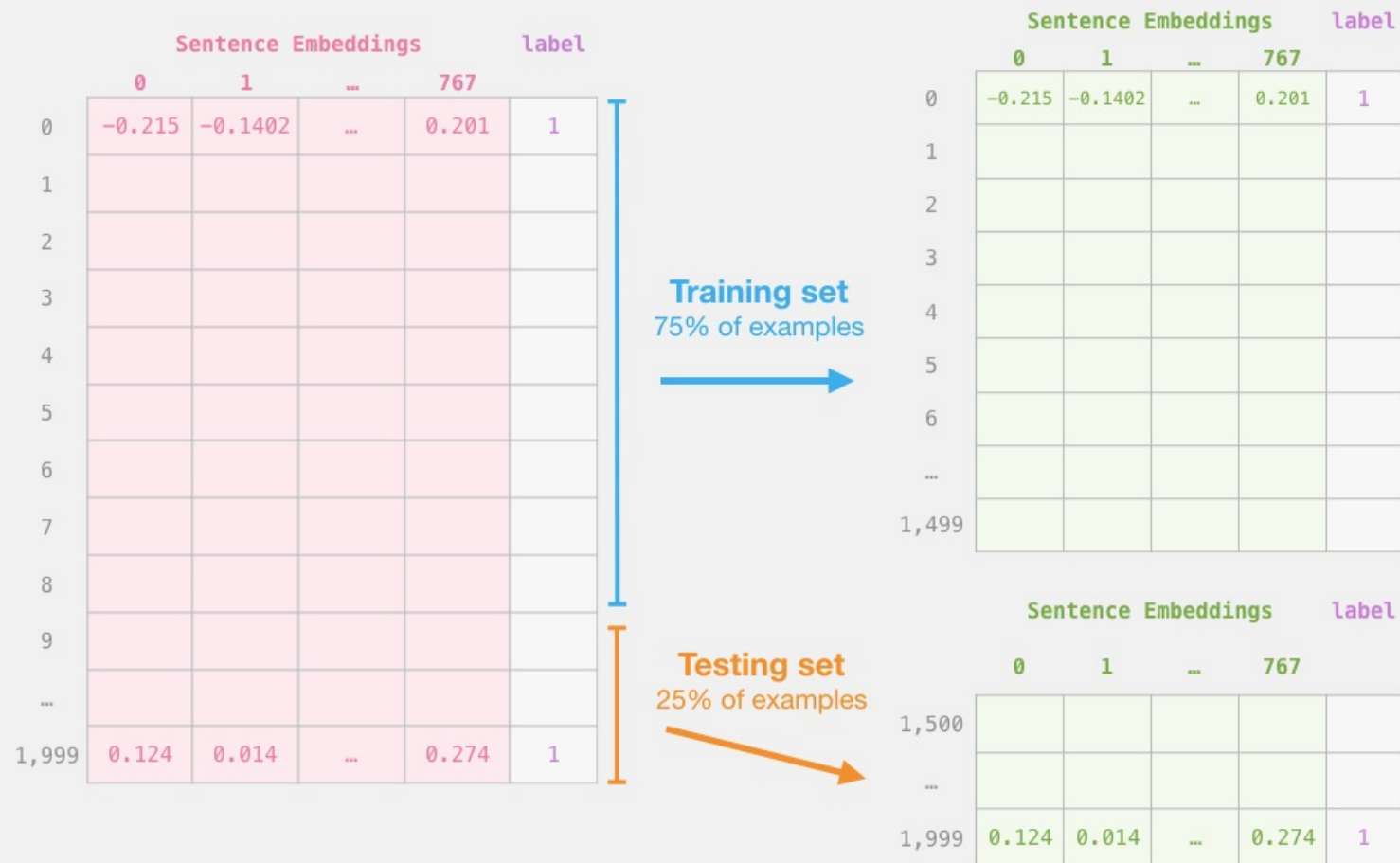
The features are the output vectors of BERT for the [CLS] token (position #0)



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

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

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



# Score Benchmarks

## Logistic Regression Model on SST-2 Dataset

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

#Testing
lr_clf.score(test_features, test_labels)

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

# Sentiment Classification: SST2

## Sentences from movie reviews

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

# A Visual Notebook to Using BERT for the First Time



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

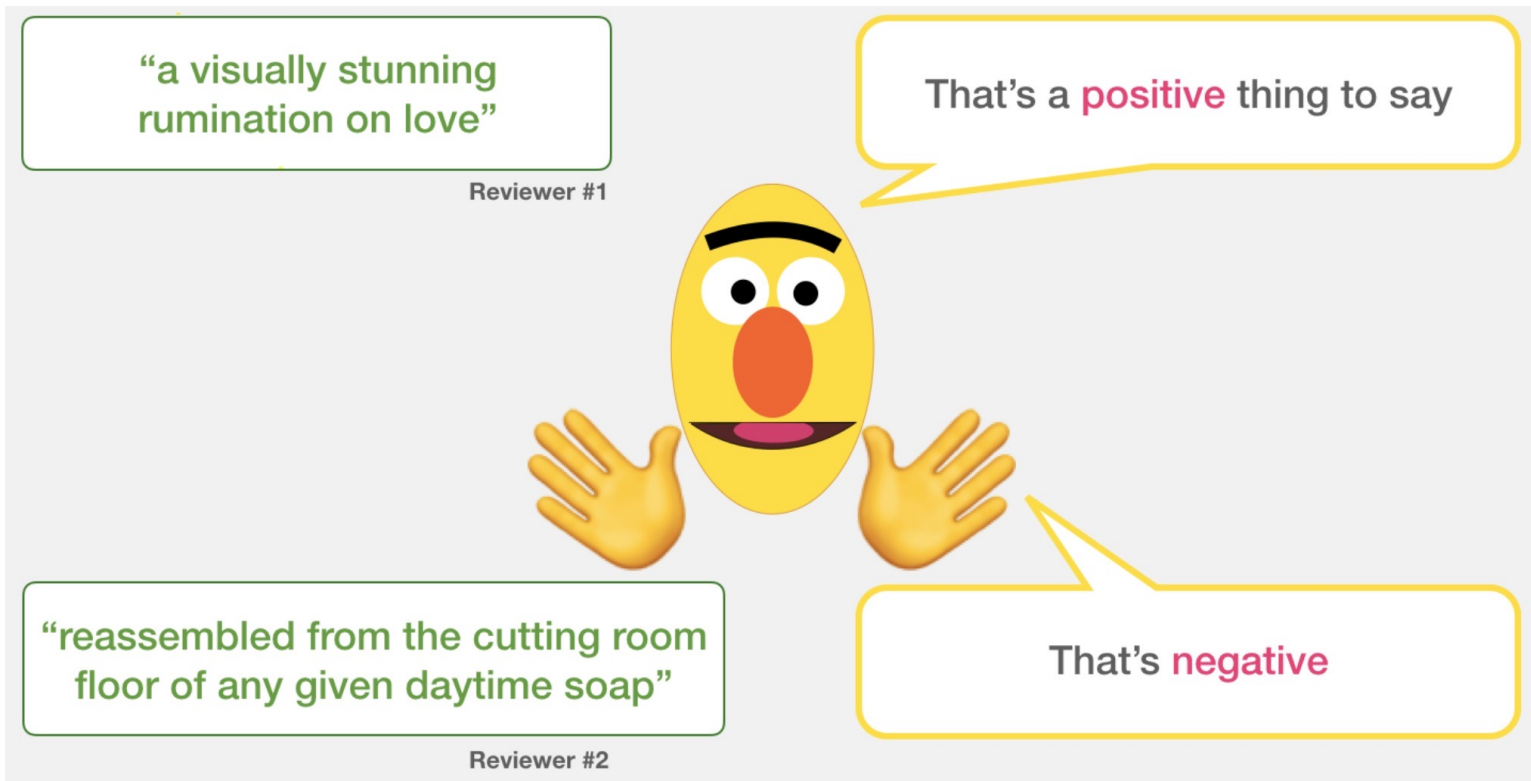
Share

File Edit View Insert Runtime Tools Help Last edited on Nov 26, 2019

+ Code + Text Copy to Drive

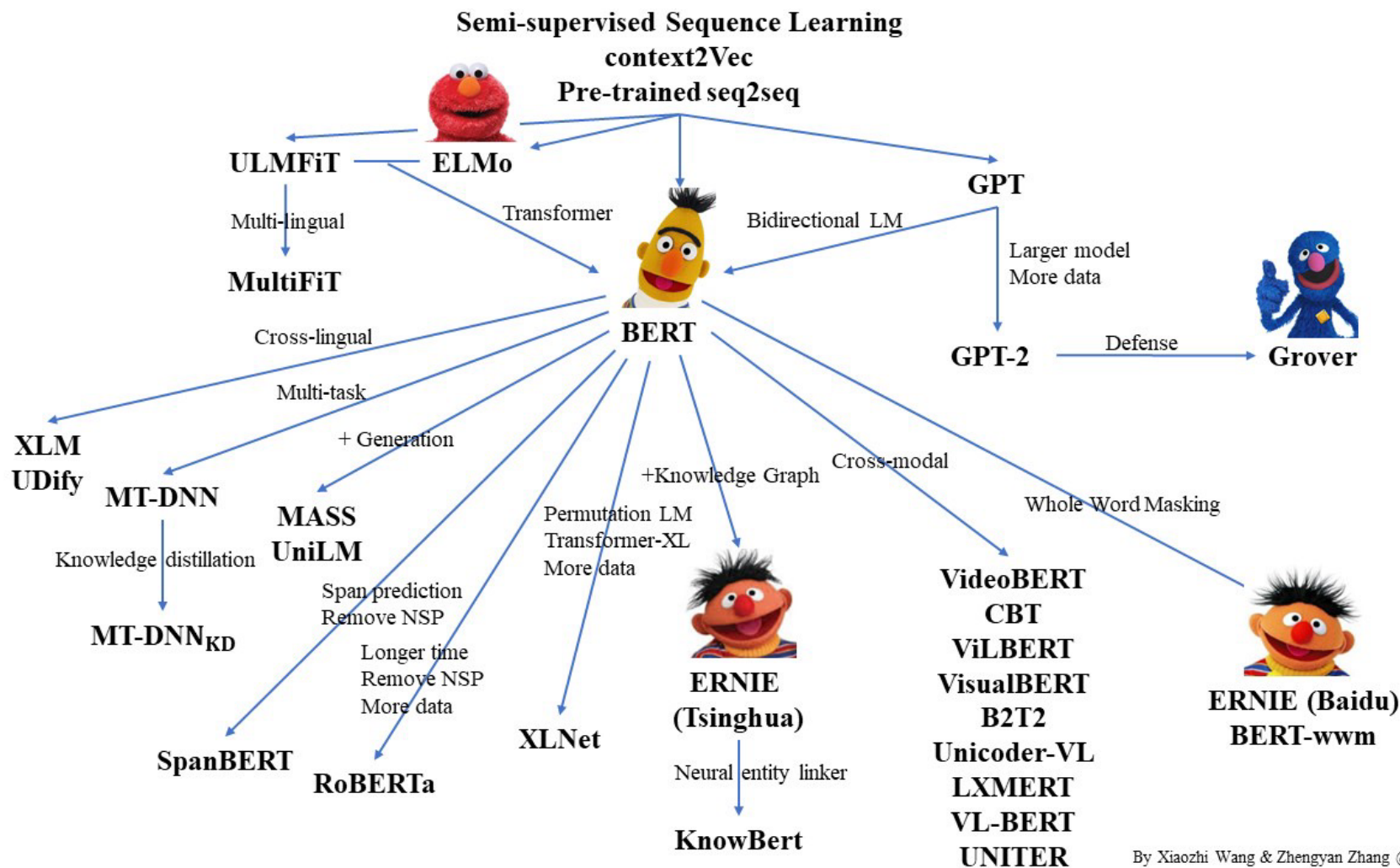
Connect Editing

▼ A Visual Notebook to Using BERT for the First Time.ipynb



[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](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)

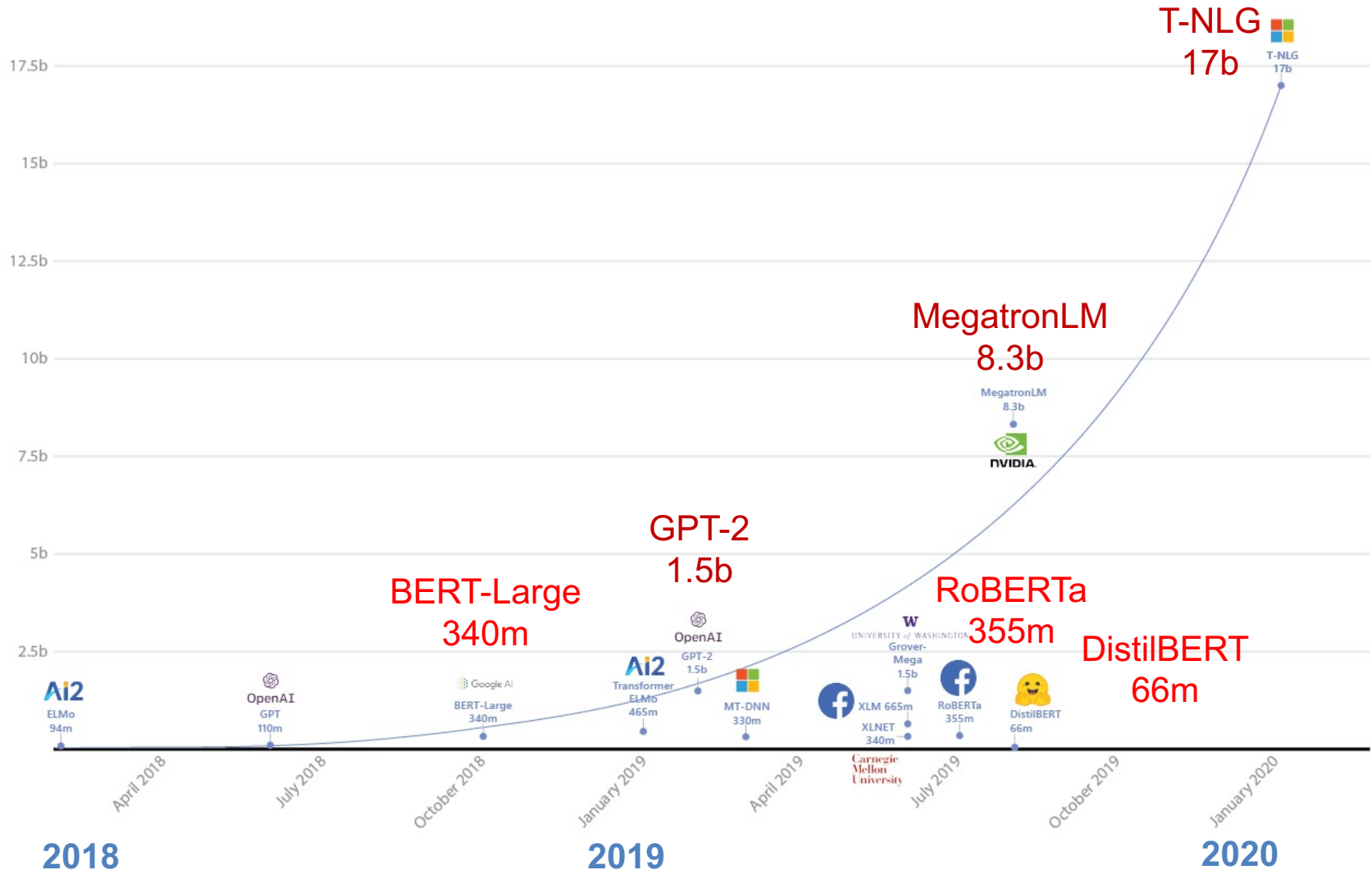
# Pre-trained Language Model (PLM)



By Xiaozhi Wang & Zhengyan Zhang @THUNLP



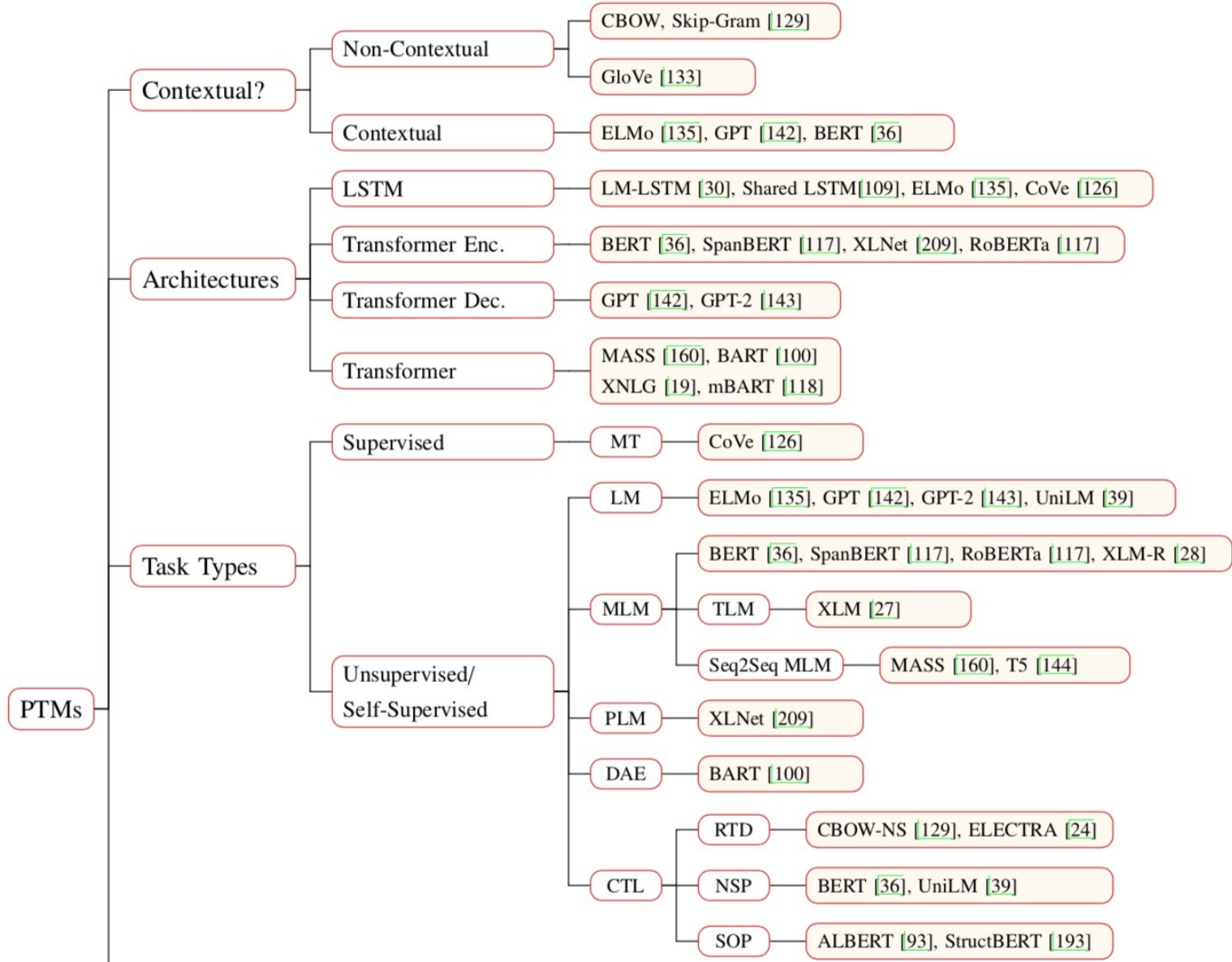
# Turing Natural Language Generation (T-NLG)



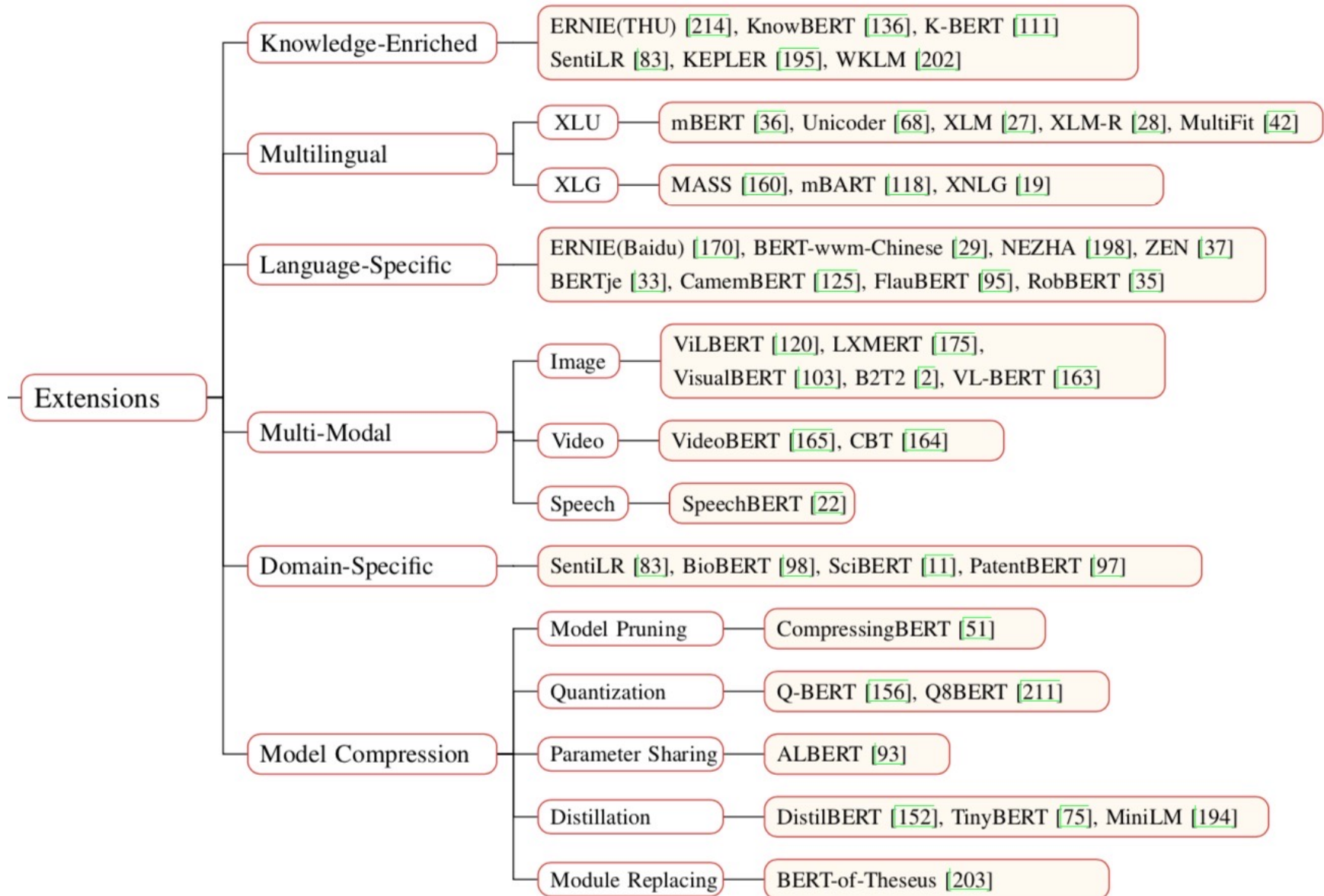
# Outline

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

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

# Question Answering

## (QA)

## SQuAD

**S**tanford **Q**uestion **A**nswering **D**ataset



# SQuAD

# SQuAD2.0

The Stanford Question Answering Dataset

## What is SQuAD?

Stanford **Q**uestion **A**nswering **D**ataset (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

{pranavs, zjian, klopyrev, pliang}@cs.stanford.edu

Computer Science Department

Stanford University

### Abstract

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

---

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

What causes precipitation to fall?

**gravity**

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

**graupel**

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

**within a cloud**

---

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

Source: Rajpurkar, Pranav, Jian Zhang, Konstantin Lopyrev, and Percy Liang.

"Squad: 100,000+ questions for machine comprehension of text." arXiv preprint arXiv:1606.05250 (2016).

# SQuAD (Question Answering)

Q: What causes precipitation to fall?

## Precipitation

---

From Wikipedia, the free encyclopedia

*For other uses, see [Precipitation \(disambiguation\)](#).*

In meteorology, **precipitation** is any product of the condensation of atmospheric water vapor that falls under gravity from clouds.<sup>[2]</sup> 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."**<sup>[3]</sup>

# 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/](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

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.

<https://spacy.io/>

# 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

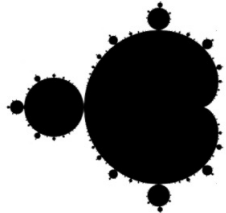


Analyze plain-text documents for semantic structure



Retrieve semantically similar documents

# TextBlob



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)



# Text Classification with TF Hub

TensorFlow tutorials  
 Quickstart for beginners  
 Quickstart for experts

## BEGINNER

### ML basics with Keras

Basic image classification

**Text classification with TF Hub**

Text classification with preprocessed text

Regression

Overfit and underfit

Save and load

### Load and preprocess data

CSV

NumPy

pandas.DataFrame

Images

Text

Unicode

TF.Text


TfRecord and tf.Example


Additional formats with tf.io


TensorFlow > Learn > TensorFlow Core > Tutorials



## Text classification with TensorFlow Hub: Movie reviews

 Run in Google Colab

 View source on GitHub

 Download notebook

This notebook classifies movie reviews as *positive* or *negative* using the text of the review. This is an example of *binary*—or two-class—classification, an important and widely applicable kind of machine learning problem.

The tutorial demonstrates the basic application of transfer learning with TensorFlow Hub and Keras.

We'll use the [IMDB dataset](#) that contains the text of 50,000 movie reviews from the [Internet Movie Database](#). These are split into 25,000 reviews for training and 25,000 reviews for testing. The training and testing sets are *balanced*, meaning they contain an equal number of positive and negative reviews.


This notebook uses [tf.keras](#), a high-level API to build and train models in TensorFlow, and [TensorFlow Hub](#), a library and platform for transfer learning. For a more advanced text classification tutorial using [tf.keras](#), see the [MLCC Text Classification Guide](#).

### Contents

- Download the IMDB dataset
- Explore the data
- Build the model
  - Loss function and optimizer
- Train the model
- Evaluate the model
- Further reading

```
from __future__ import absolute_import, division, print_function, unicode_literals
```

# Text Classification with Pre Text



Install
Learn ▾
API ▾
Resources ▾
More ▾

🔍

🌐

GitHub
Sign in

TensorFlow tutorials

Quickstart for beginners

Quickstart for experts

---

BEGINNER

---

ML basics with Keras ^

- Basic image classification
- Text classification with TF Hub
- Text classification with preprocessed text
- Regression
- Overfit and underfit
- Save and load

---

Load and preprocess data ^


- CSV
- NumPy
- pandas.DataFrame
- Images
- Text
- Unicode
- TF.Text
- TFRecord and tf.Example
- Additional formats with tf.io [🔗](#)

---


Estimator v

TensorFlow > Learn > TensorFlow Core > Tutorials ☆☆☆☆☆


## Text classification with preprocessed text: Movie reviews



Run in Google Colab



View source on GitHub



Download notebook

This notebook classifies movie reviews as *positive* or *negative* using the text of the review. This is an example of *binary*—or two-class—classification, an important and widely applicable kind of machine learning problem.

We'll use the [IMDB dataset](#) that contains the text of 50,000 movie reviews from the [Internet Movie Database](#). These are split into 25,000 reviews for training and 25,000 reviews for testing. The training and testing sets are *balanced*, meaning they contain an equal number of positive and negative reviews.

This notebook uses [tf.keras](#), a high-level API to build and train models in TensorFlow. For a more advanced text classification tutorial using `tf.keras`, see the [MLCC Text Classification Guide](#).

### Setup

```
from __future__ import absolute_import, division, print_function, unicode_literals
```

**Contents**

- Setup
- Download the IMDB dataset
- Try the encoder
- Explore the data
- Prepare the data for training
- Build the model
  - Hidden units
  - Loss function and optimizer
- Train the model
- Evaluate the model
- Create a graph of accuracy and loss over time

# Text Classification

## IMDB Movie Reviews

[https://colab.research.google.com/drive/1x16h1GhHsLrLYtPCvCHaoO1W-i\\_gror](https://colab.research.google.com/drive/1x16h1GhHsLrLYtPCvCHaoO1W-i_gror)

The screenshot shows a Google Colab notebook titled "tf02\_basic-text-classification.ipynb". The interface includes a top navigation bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help" menus. On the right, there are buttons for "COMMENT", "SHARE", and a user profile icon. Below the navigation bar, there are tabs for "+ CODE", "+ TEXT", and "CELL" controls. A sidebar on the left contains a "Table of contents" with the following items: "Copyright 2018 The TensorFlow Authors.", "Licensed under the Apache License, Version 2.0 (the 'License');", "MIT License", "Text classification with movie reviews" (highlighted), "Download the IMDB dataset", "Explore the data", "Convert the integers back to words", "Prepare the data", "Build the model", "Hidden units", "Loss function and optimizer", "Create a validation set", "Train the model", and "Evaluate the model".

The main content area of the notebook shows a copyright notice for 2018 The TensorFlow Authors, followed by a section titled "Text classification with movie reviews". This section includes three links: "View on TensorFlow.org", "Run in Google Colab", and "View source on GitHub". Below the links, there is a paragraph explaining that the notebook classifies movie reviews as *positive* or *negative* using the text of the review, which is an example of *binary* or two-class classification. It then states that the [IMDB dataset](#) (containing 50,000 reviews) is used, split into 25,000 training and 25,000 testing reviews, which are *balanced*. The text concludes by mentioning that the notebook uses [tf.keras](#) and provides a link to the [MLCC Text Classification Guide](#).

At the bottom of the notebook, a code cell is visible with the following Python code:

```
1 # memory footprint support libraries/code
2 !ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
3 !pip install gputil
4 !pip install psutil
5 !pip install humanize
6 import psutil
7 import humanize
8 import os
9 import GPUtil as GPU
10 GPUs = GPU.getGPUs()
11 gpu = GPUs[0]
12 def printm():
13     process = psutil.Process(os.getpid())
```

Source: [https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic\\_text\\_classification.ipynb](https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_text_classification.ipynb)

# Papers with Code State-of-the-Art (SOTA)



Search for papers, code and tasks



[Browse State-of-the-Art](#)

[Follow](#)

[Discuss](#)

[Trends](#)

[About](#)

[Log In/Register](#)

## Browse State-of-the-Art

1509 leaderboards • 1327 tasks • 1347 datasets • 17810 papers with code

Follow on [Twitter](#) for updates

## Computer Vision



Semantic Segmentation

33 leaderboards  
667 papers with code



Image Classification

52 leaderboards  
564 papers with code



Object Detection

54 leaderboards  
467 papers with code



Image Generation

51 leaderboards  
231 papers with code



Pose Estimation

40 leaderboards  
231 papers with code

[See all 707 tasks](#)

## Natural Language Processing



Machine Translation



Language Modelling



Question Answering



Sentiment Analysis

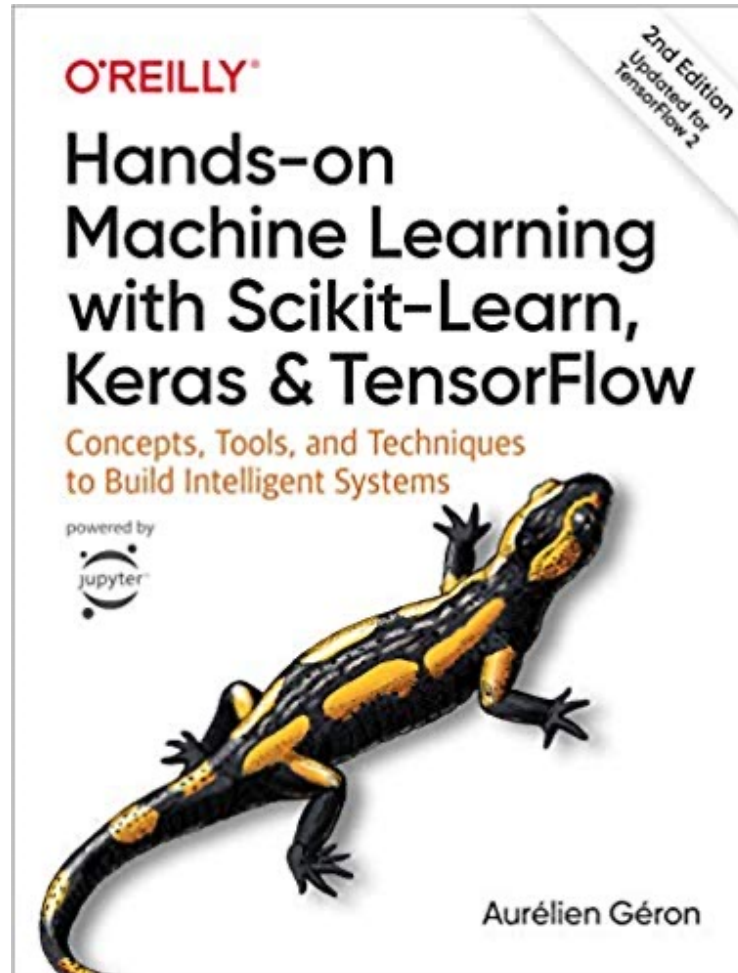


Text Generation



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, 2019

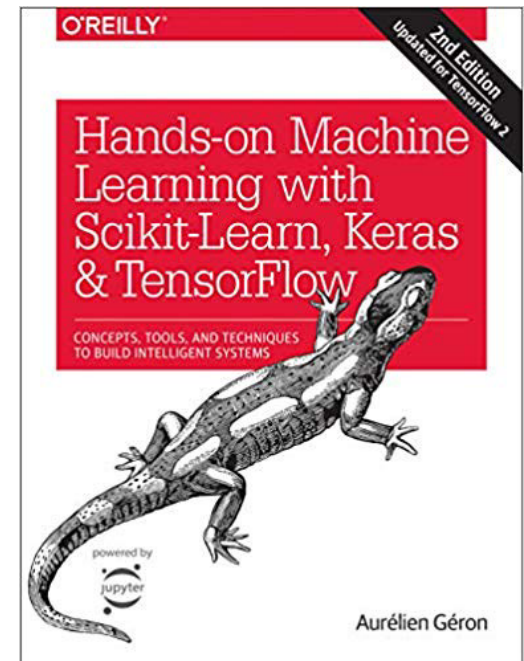


<https://github.com/ageron/handson-ml2>

# Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

## Notebooks

- [1. The Machine Learning landscape](#)
- [2. End-to-end Machine Learning project](#)
- [3. Classification](#)
- [4. Training Models](#)
- [5. Support Vector Machines](#)
- [6. Decision Trees](#)
- [7. Ensemble Learning and Random Forests](#)
- [8. Dimensionality Reduction](#)
- [9. Unsupervised Learning Techniques](#)
- [10. Artificial Neural Nets with Keras](#)
- [11. Training Deep Neural Networks](#)
- [12. Custom Models and Training with TensorFlow](#)
- [13. Loading and Preprocessing Data](#)
- [14. Deep Computer Vision Using Convolutional Neural Networks](#)
- [15. Processing Sequences Using RNNs and CNNs](#)
- [16. Natural Language Processing with RNNs and Attention](#)
- [17. Representation Learning Using Autoencoders](#)
- [18. Reinforcement Learning](#)
- [19. Training and Deploying TensorFlow Models at Scale](#)



# Python in Google Colab (Python101)

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

The screenshot displays the Google Colab interface for a notebook titled "python101.ipynb". The top navigation bar includes "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help", with a status indicator "All changes saved". On the right, there are icons for "Comment", "Share", and "Settings", along with a RAM and Disk usage monitor and an "Editing" mode indicator.

The left sidebar shows a "Table of contents" with a search icon and a list of topics. The "Text Classification: IMDB Movie Review" section is highlighted in yellow. Other topics include "Machine Learning with scikit-learn", "Deep Learning", "Portfolio Optimization and Algorithmic Trading", and "Text Analytics and Natural Language Processing (NLP)".

The main workspace shows three code cells:

- Cell [1]:** Installation of TensorFlow Hub and TensorFlow Datasets.

```
[1] 1 !pip install -q tensorflow-hub
    2 !pip install -q tensorflow-datasets
```
- Cell [2]:** Importing necessary libraries and checking their versions.

```
[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
- Cell [3]:** Splitting the training set into 60% and 40% for training and testing.

```
[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/aintpuppython101>

# 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

- 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)
      - Python for Natural Language Processing
        - spaCy Chinese Model

+ Code + Text

- Huggingface Transformers: <https://github.com/huggingface/transformers>

```
[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.')
```

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]

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

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

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

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

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

<https://tinyurl.com/aintpuppython101>

# Python in Google Colab (Python101)

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

The screenshot displays a Google Colab notebook titled "python101.ipynb". The interface includes a top navigation bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help" menus, along with a "Comment" button, a "Share" button, and a settings gear. A "Table of contents" sidebar on the left lists various topics such as "Mall 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", and "Text Analytics and Natural Language Processing (NLP)".

The main workspace shows three code execution cells:

- Cell 1:** Imports the `pipeline` function from `transformers` and defines a pipeline for question-answering. It then uses the pipeline to answer the question "What is the name of the repository?". The output is a dictionary: `{'answer': 'huggingface/transformers', 'end': 58, 'score': 0.309702068567276, 'start': 34}`.
- Cell 2:** Defines a context string: `ontext = '''In meteorology, precipitation is any product of the condensation of atmospheric water vapor`. The output shows the start and end indices of the context within the text.
- Cell 3:** Uses the pipeline to answer the question "Where do water droplets collide with ice crystals to form precipitation?". The output is a dictionary: `{'answer': 'within a cloud', 'end': 321, 'score': 0.5175967812538147, 'start': 307}`.

At the bottom of the notebook, there is a fourth code execution cell:

```
[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>

The screenshot shows a Google Colab notebook titled "python101.ipynb". The interface includes a top navigation bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help" menus. A "Table of contents" sidebar on the left lists various topics, with "Universal Sentence Encoder (USE)" highlighted. The main workspace contains a code cell with the following Python code:

```
[ ] 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)
```

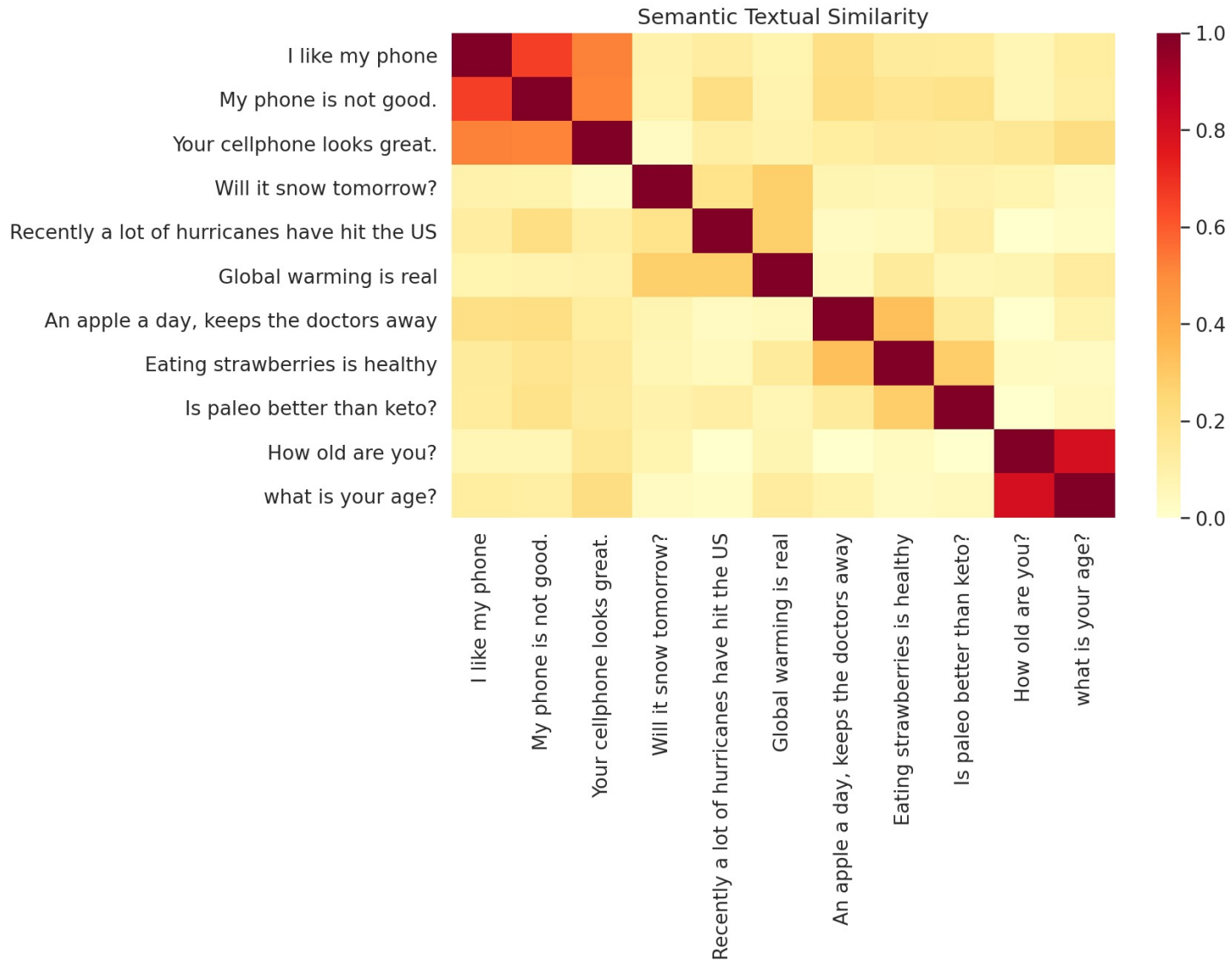
Below the code cell, the output shows: `module https://tfhub.dev/google/universal-sentence-encoder/4 loaded`. A second code cell is partially visible at the bottom:

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

<https://tinyurl.com/aintpuppython101>

# Python in Google Colab (Python101)

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



<https://tinyurl.com/aintpupython101>

# Summary

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



# References

- Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson.
- Aurélien Géron (2019), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd Edition, O'Reilly Media.
- 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 Alamar (2019), The Illustrated Transformer, <http://jalamar.github.io/illustrated-transformer/>
- Jay Alamar (2019), A Visual Guide to Using BERT for the First Time, <http://jalamar.github.io/a-visual-guide-to-using-bert-for-the-first-time/>
- 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 (2021), Python 101, <https://tinyurl.com/aintpupython101>