

人工智慧文本分析



Tamkang
Universit
淡江大學

(Artificial Intelligence for Text Analytics)

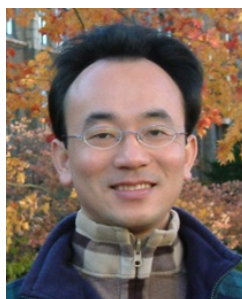
文本分類

(Text Classification)

1082AITA06

MBA, IMTKU (M2455) (8410) (Spring 2020)

Wed 8, 9 (15:10-17:00) (B605)



Min-Yuh Day

戴敏育

Associate Professor

副教授

Dept. of Information Management, Tamkang University

淡江大學 資訊管理學系

<http://mail.tku.edu.tw/myday/>

2020-04-15



課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
1	2020/03/04	人工智慧文本分析課程介紹 (Course Orientation on Artificial Intelligence for Text Analytics)
2	2020/03/11	文本分析的基礎：自然語言處理 (Foundations of Text Analytics: Natural Language Processing; NLP)
3	2020/03/18	Python自然語言處理 (Python for Natural Language Processing)
4	2020/03/25	處理和理解文本 (Processing and Understanding Text)
5	2020/04/01	文本表達特徵工程 (Feature Engineering for Text Representation)
6	2020/04/08	人工智慧文本分析個案研究 I (Case Study on Artificial Intelligence for Text Analytics I)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
7	2020/04/15	文本分類 (Text Classification)
8	2020/04/22	文本摘要和主題模型 (Text Summarization and Topic Models)
9	2020/04/29	期中報告 (Midterm Project Report)
10	2020/05/06	文本相似度和分群 (Text Similarity and Clustering)
11	2020/05/13	語意分析和命名實體識別 (Semantic Analysis and Named Entity Recognition; NER)
12	2020/05/20	情感分析 (Sentiment Analysis)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
13	2020/05/27	人工智慧文本分析個案研究 II (Case Study on Artificial Intelligence for Text Analytics II)
14	2020/06/03	深度學習和通用句子嵌入模型 (Deep Learning and Universal Sentence-Embedding Models)
15	2020/06/10	問答系統與對話系統 (Question Answering and Dialogue Systems)
16	2020/06/17	期末報告 I (Final Project Presentation I)
17	2020/06/24	期末報告 II (Final Project Presentation II)
18	2020/07/01	教師彈性補充教學

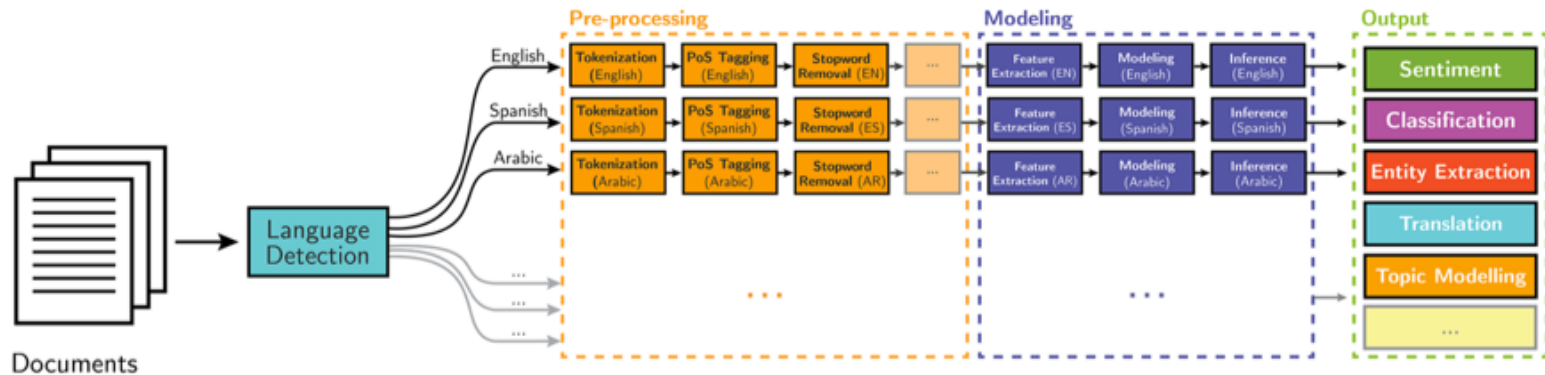
Outline

- Text Classification
- Classification Model Evaluation
 - Confusion Matrix
 - Accuracy
 - Precision
 - Recall (TPR) (Sensitivity) (Hit Rate)
 - F1 score (F-measure) (F-score)

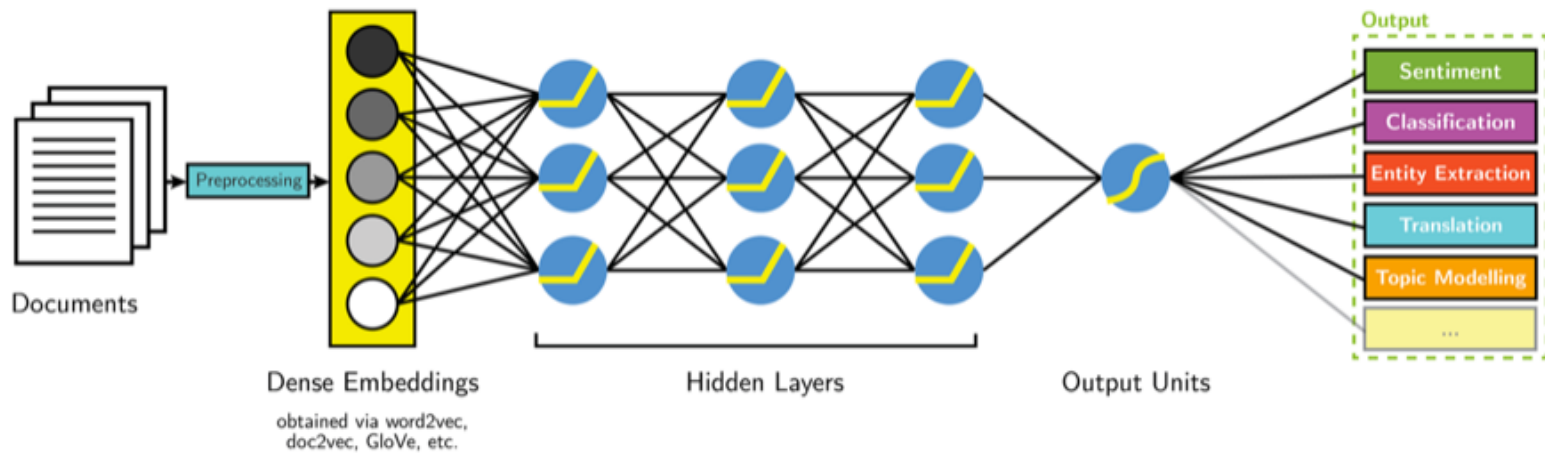
Text Classification

NLP

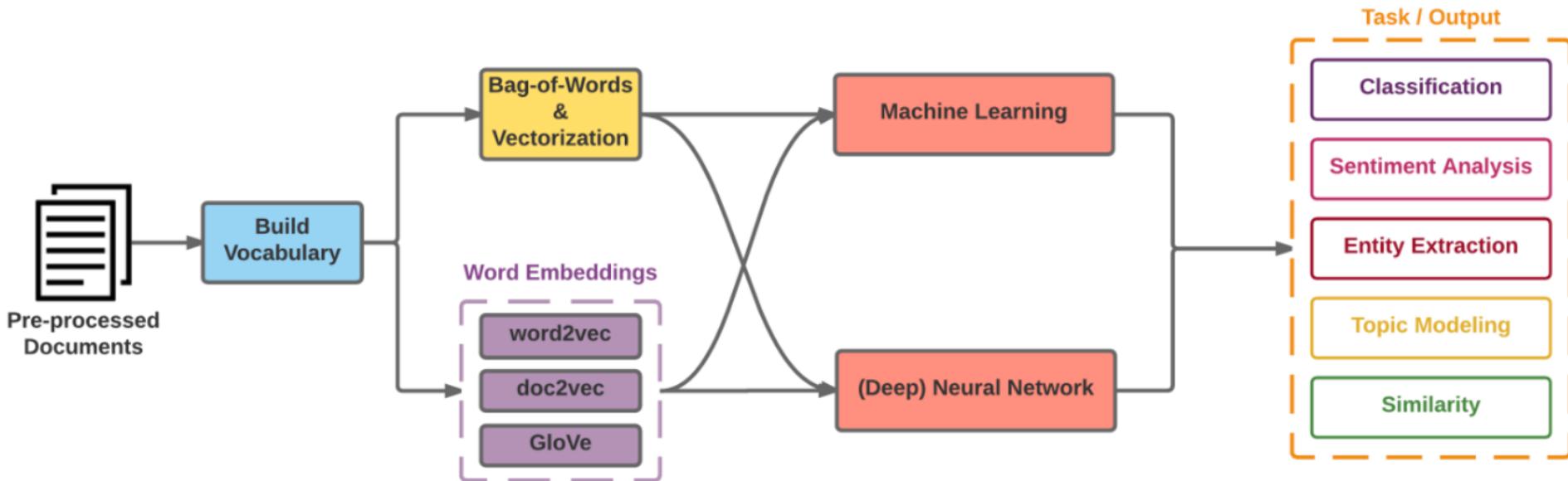
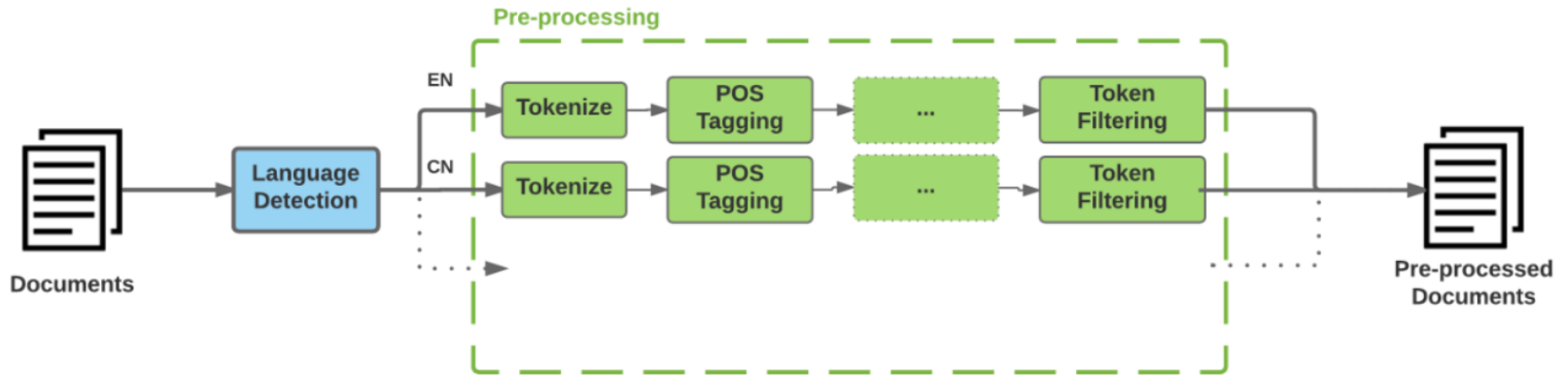
Classical NLP



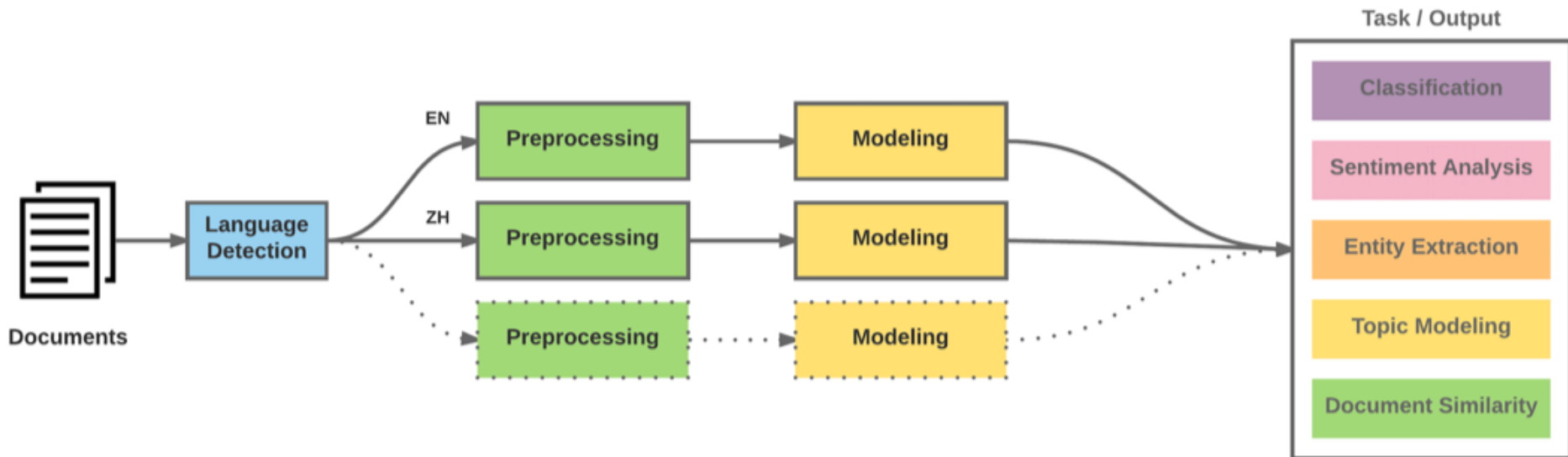
Deep Learning-based NLP



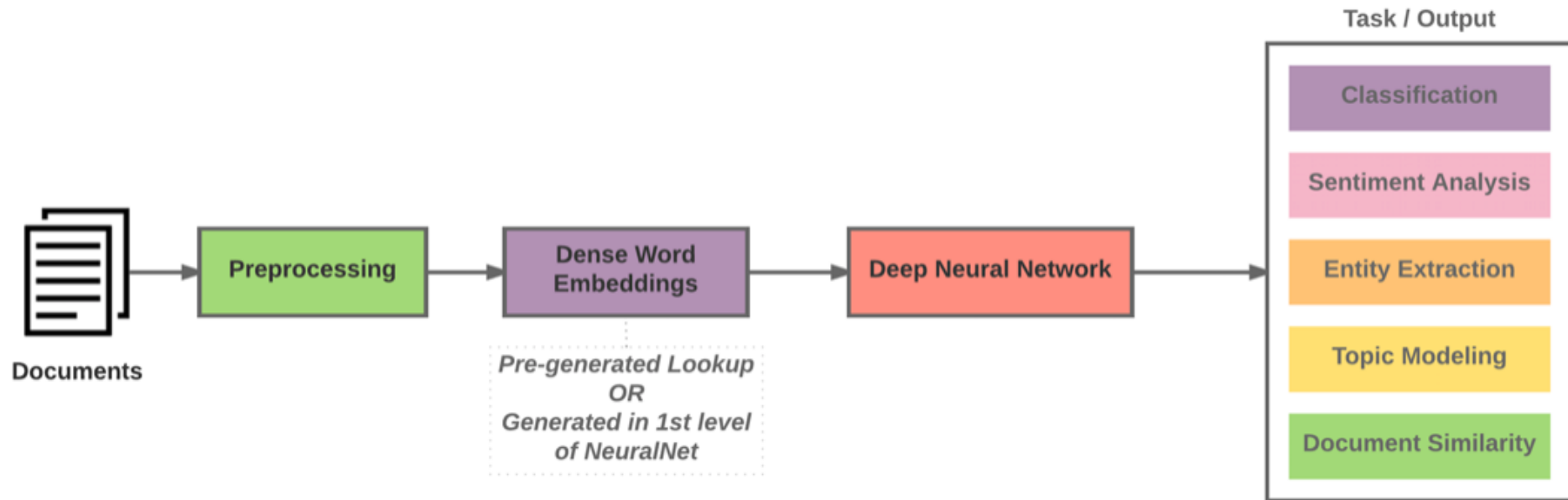
Modern NLP Pipeline



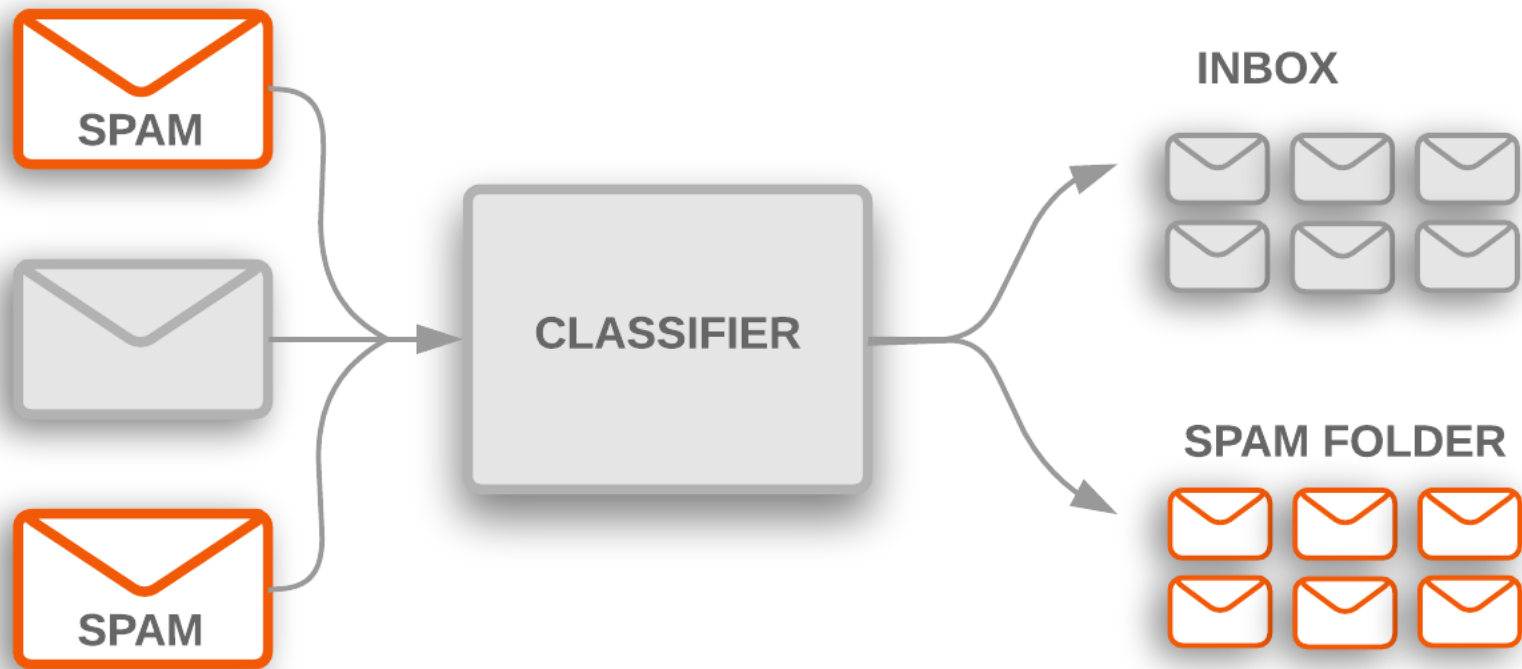
Modern NLP Pipeline



Deep Learning NLP

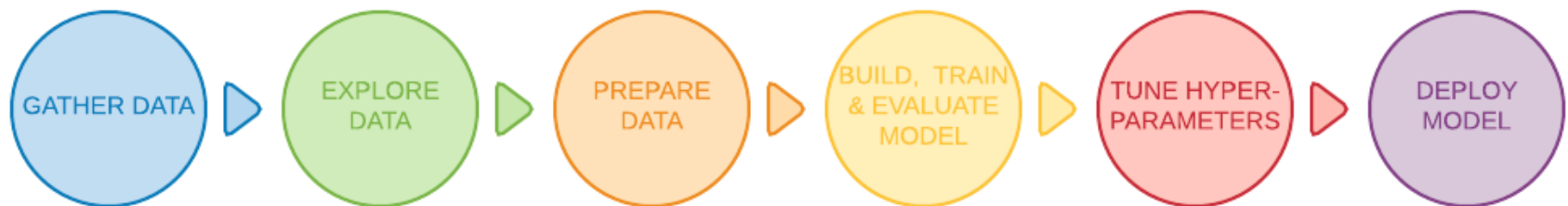


Text Classification

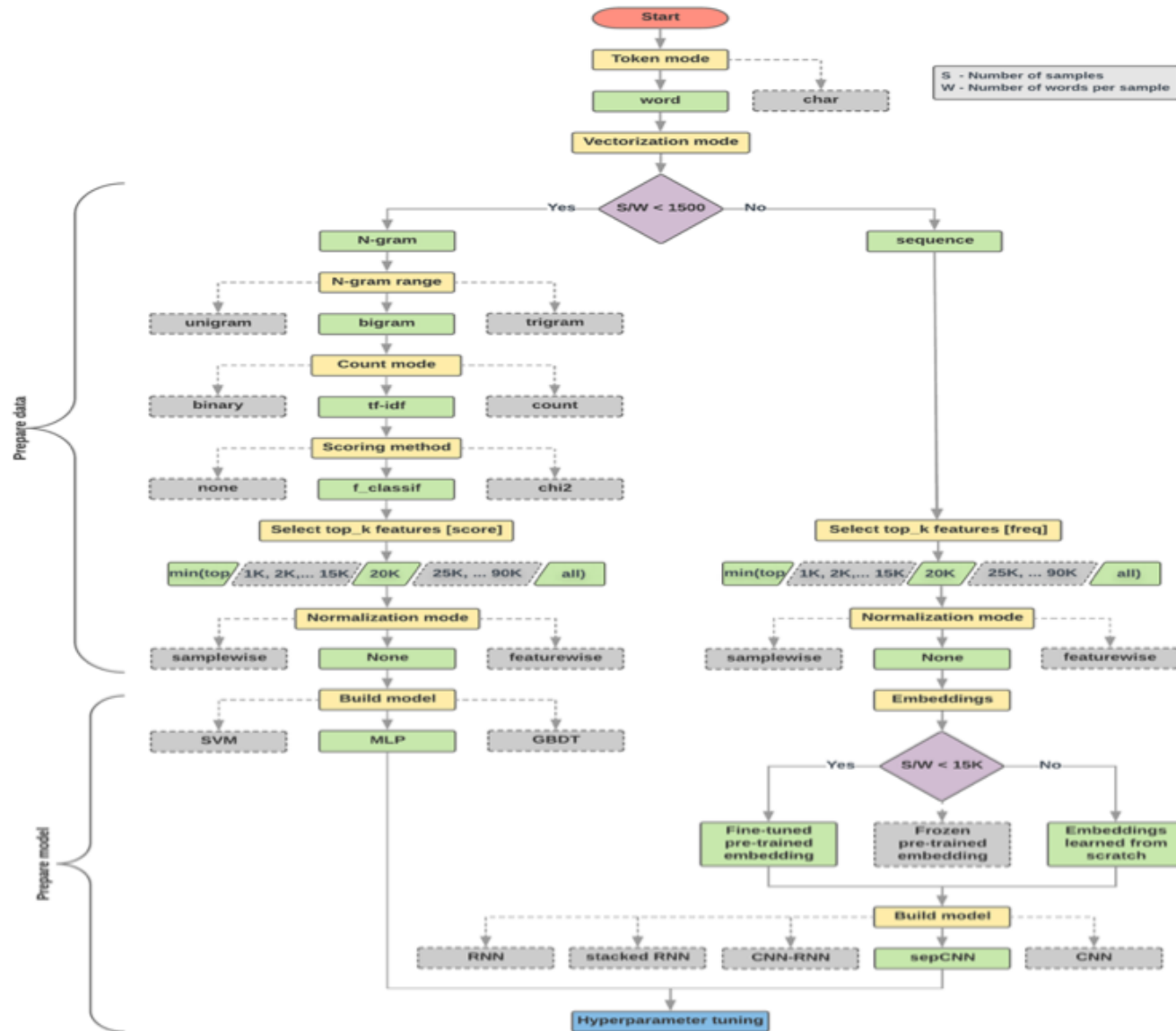


Text Classification Workflow

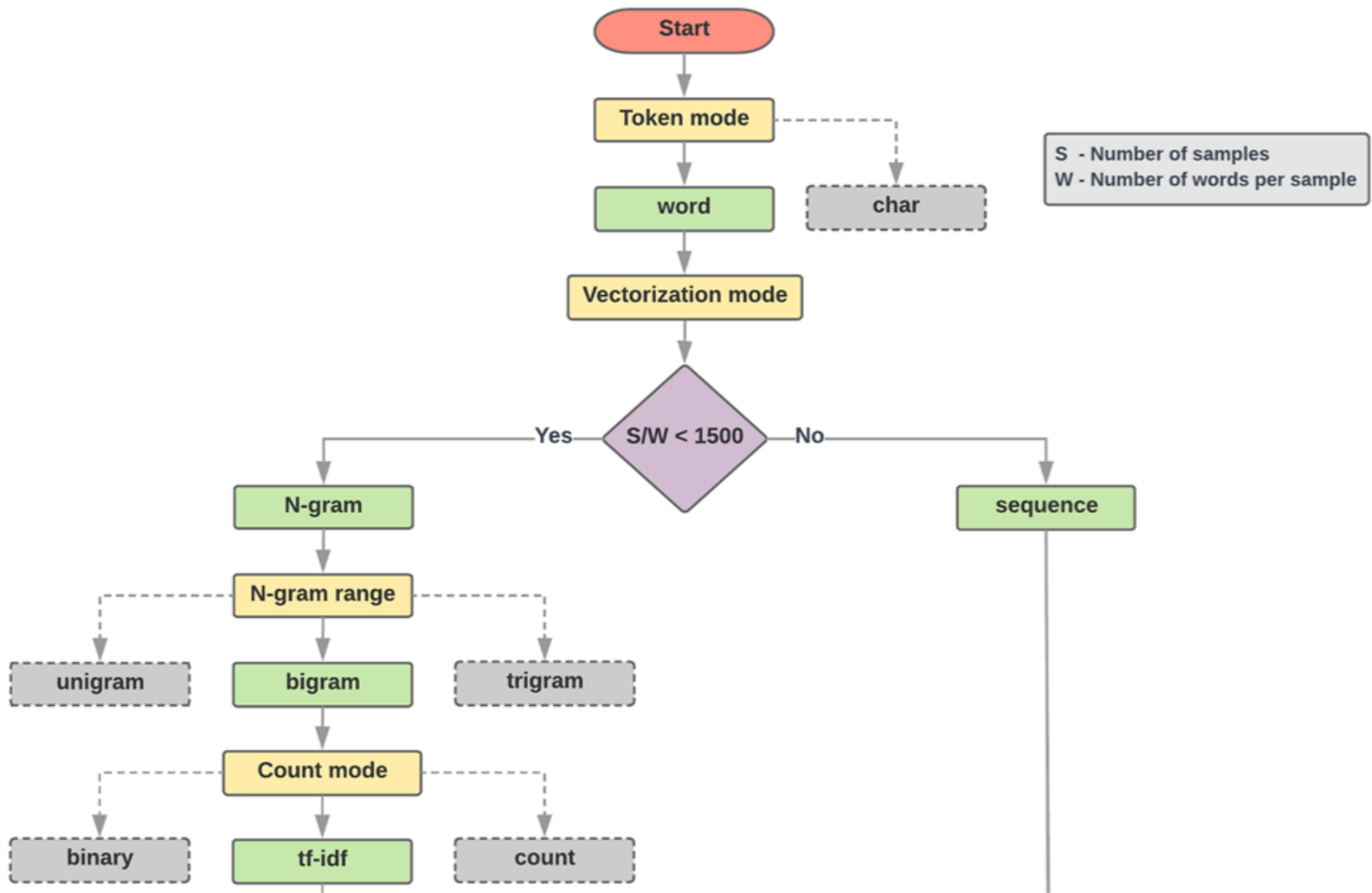
- Step 1: Gather Data
- Step 2: Explore Your Data
- Step 2.5: Choose a Model*
- Step 3: Prepare Your Data
- Step 4: Build, Train, and Evaluate Your Model
- Step 5: Tune Hyperparameters
- Step 6: Deploy Your Model



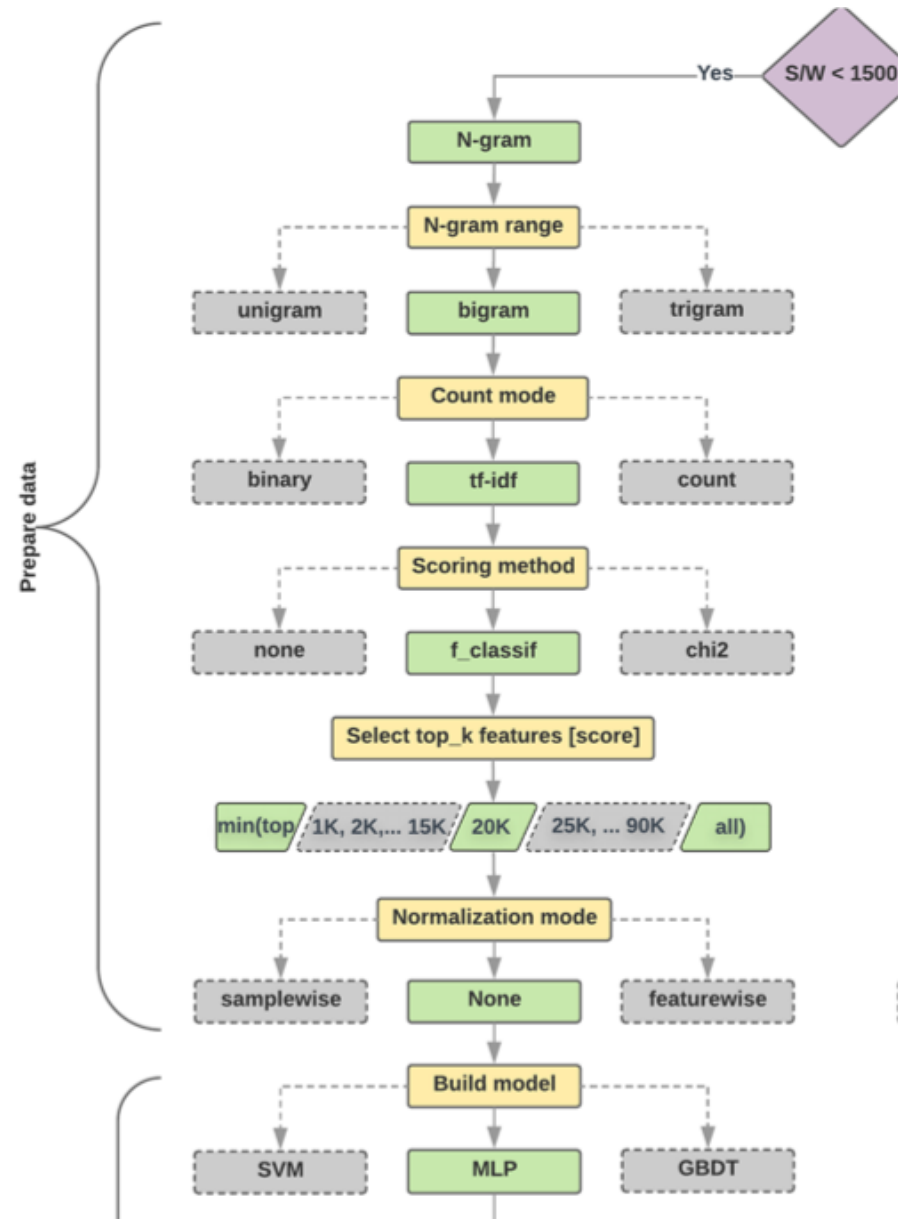
Text Classification Flowchart



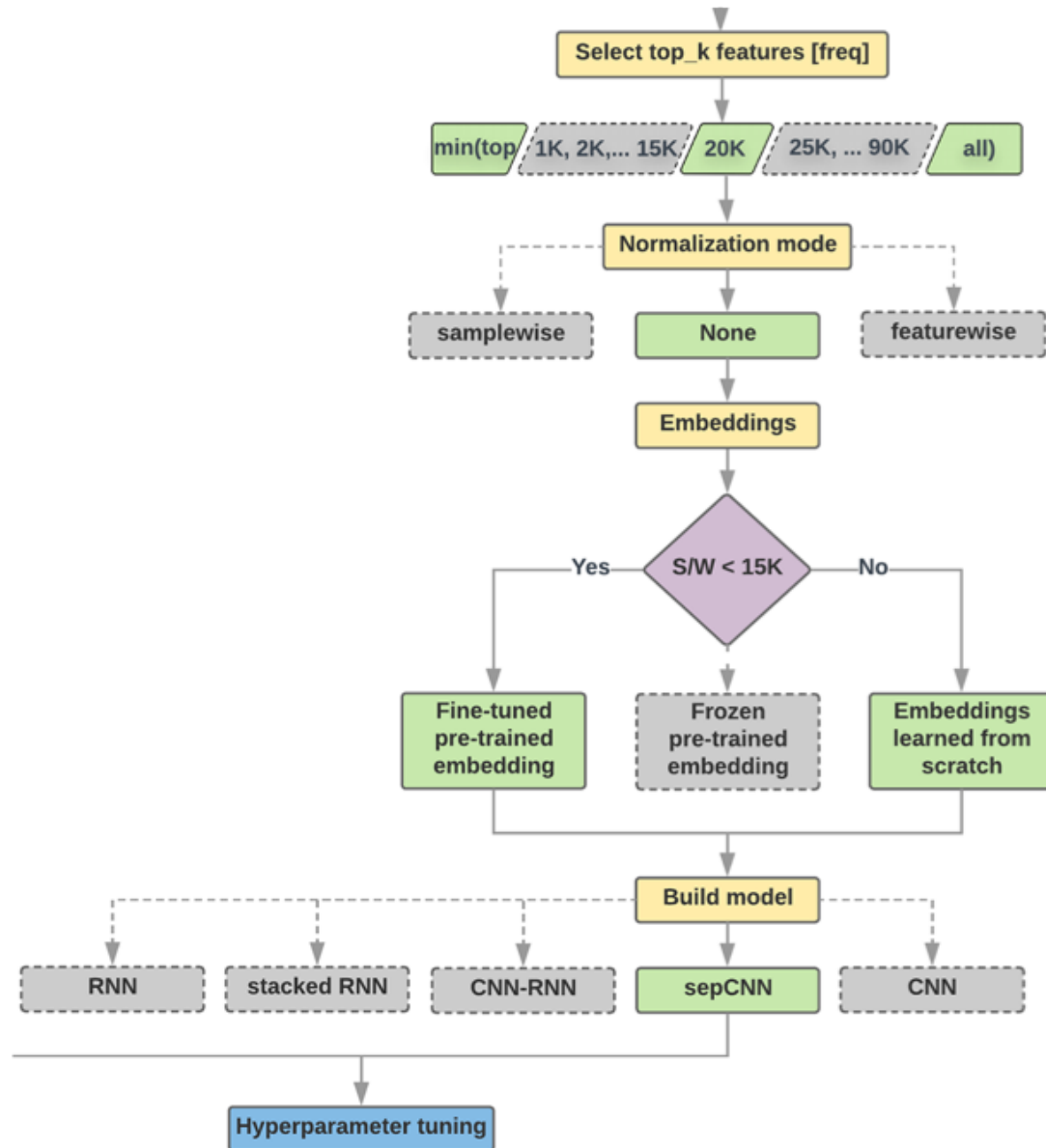
Text Classification Flowchart



Text Classification S/W<1500: N-gram



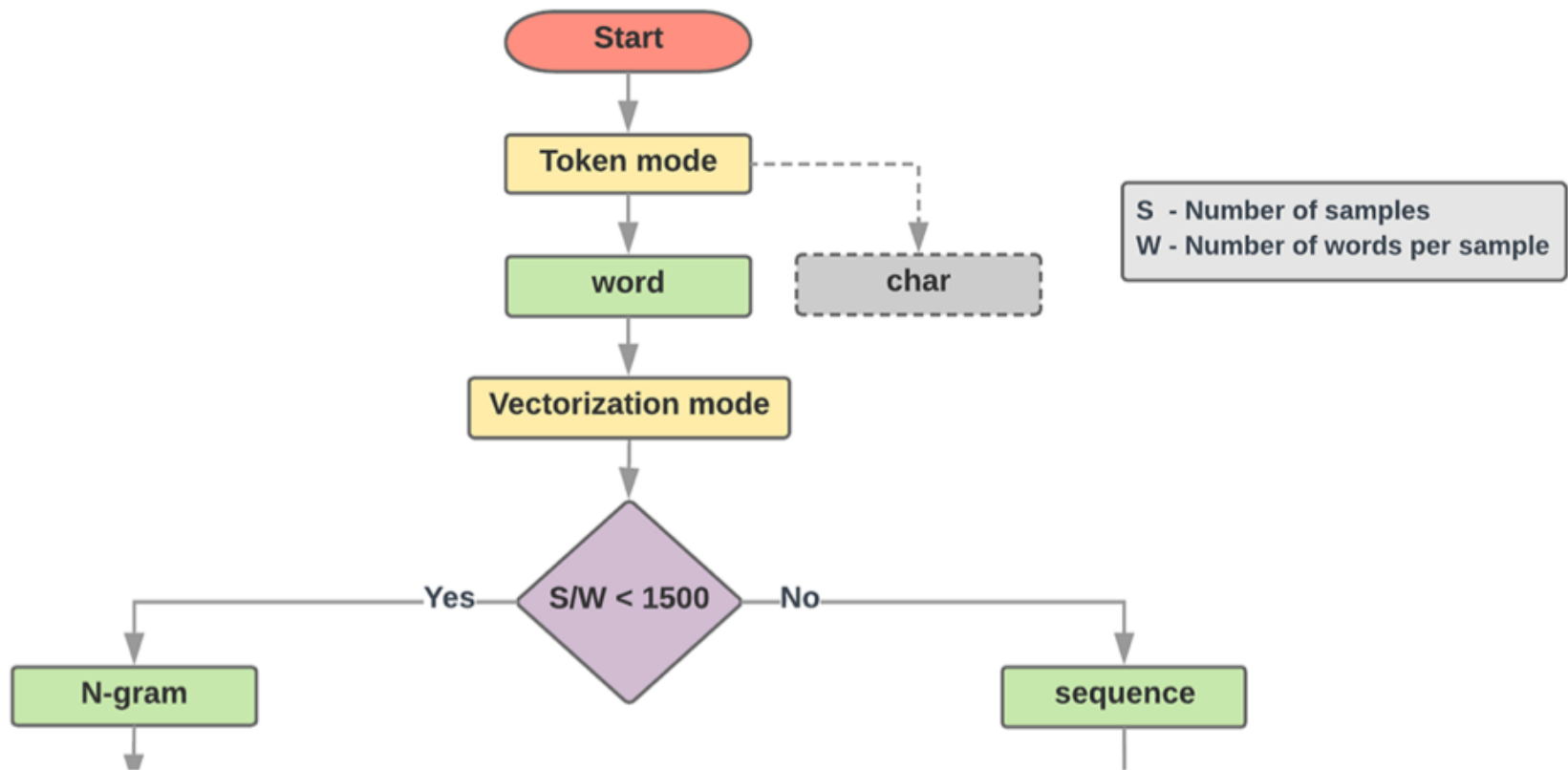
Text Classification $S/W \geq 1500$: Sequence



Step 2.5: Choose a Model

Samples/Words < 1500

$$150,000/100 = 1500$$

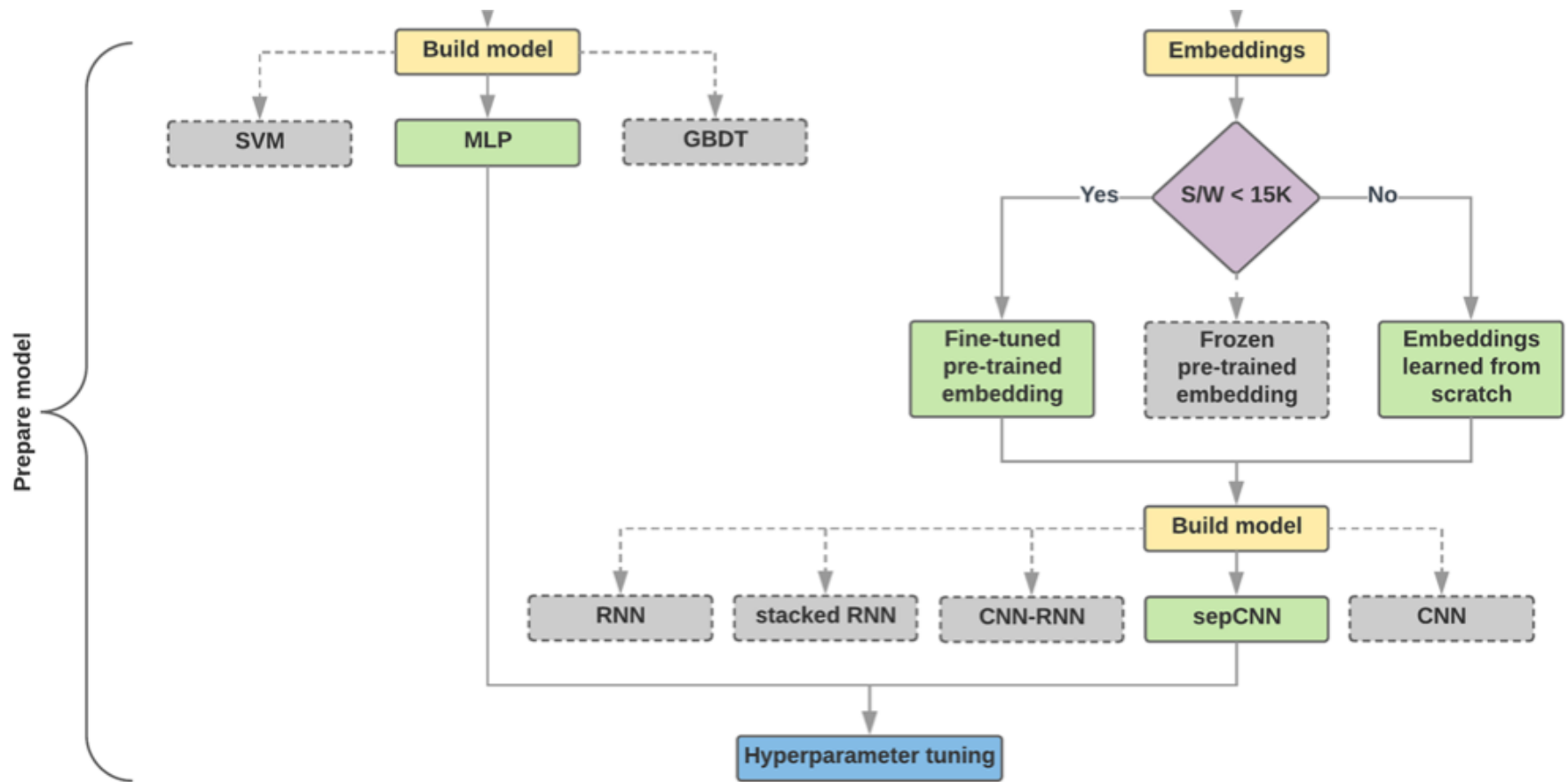


IMDb review dataset,
the samples/words-per-sample ratio is ~ 144

Step 2.5: Choose a Model

Samples/Words < 15,000

1,500,000/100 = 15,000



Step 3: Prepare Your Data

Texts:

T1: 'The mouse ran up the clock'

T2: 'The mouse ran down'

Token Index:

```
{'the': 1, 'mouse': 2, 'ran': 3, 'up': 4, 'clock': 5, 'down': 6,}.
```

NOTE: 'the' occurs most frequently,
so the index value of 1 is assigned to it.
Some libraries reserve index 0 for unknown tokens,
as is the case here.

Sequence of token indexes:

T1: 'The mouse ran up the clock' =
[1, 2, 3, 4, 1, 5]

T2: 'The mouse ran down' =
[1, 2, 3, 6]

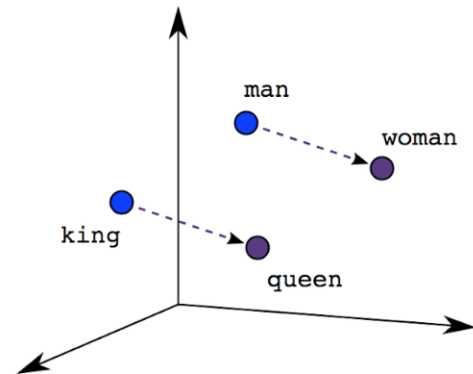
One-hot encoding

'The mouse ran up the clock' =

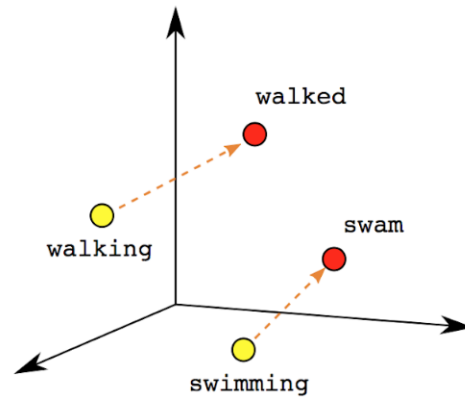
The	1	[	[0, 1, 0, 0, 0, 0, 0],
mouse	2		[0, 0, 1, 0, 0, 0, 0],
ran	3		[0, 0, 0, 1, 0, 0, 0],
up	4		[0, 0, 0, 0, 1, 0, 0],
the	1		[0, 1, 0, 0, 0, 0, 0],
clock	5		[0, 0, 0, 0, 0, 1, 0]]

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

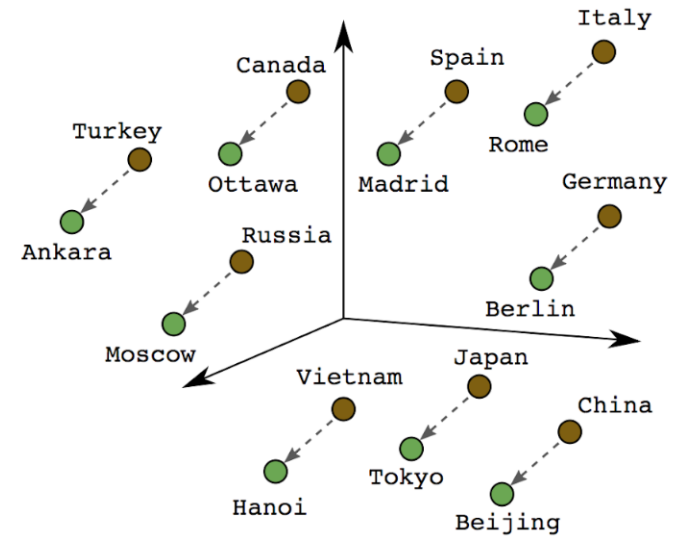
Word embeddings



Male-Female

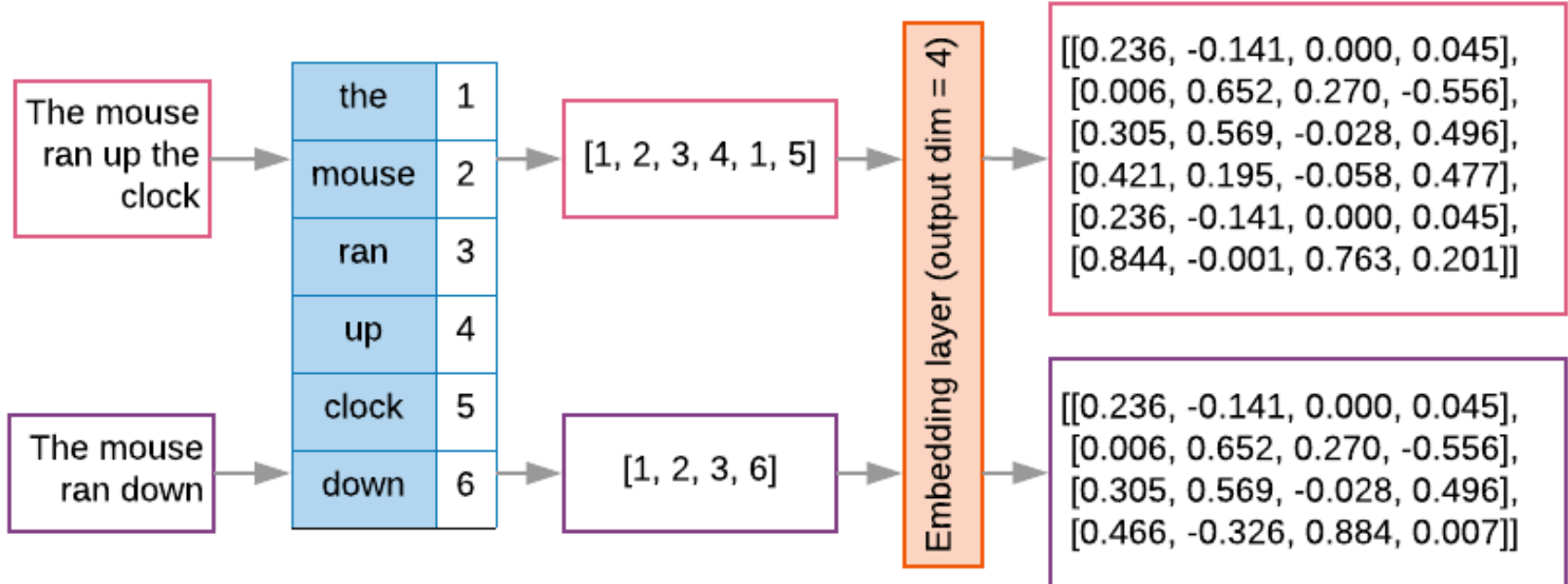


Verb Tense



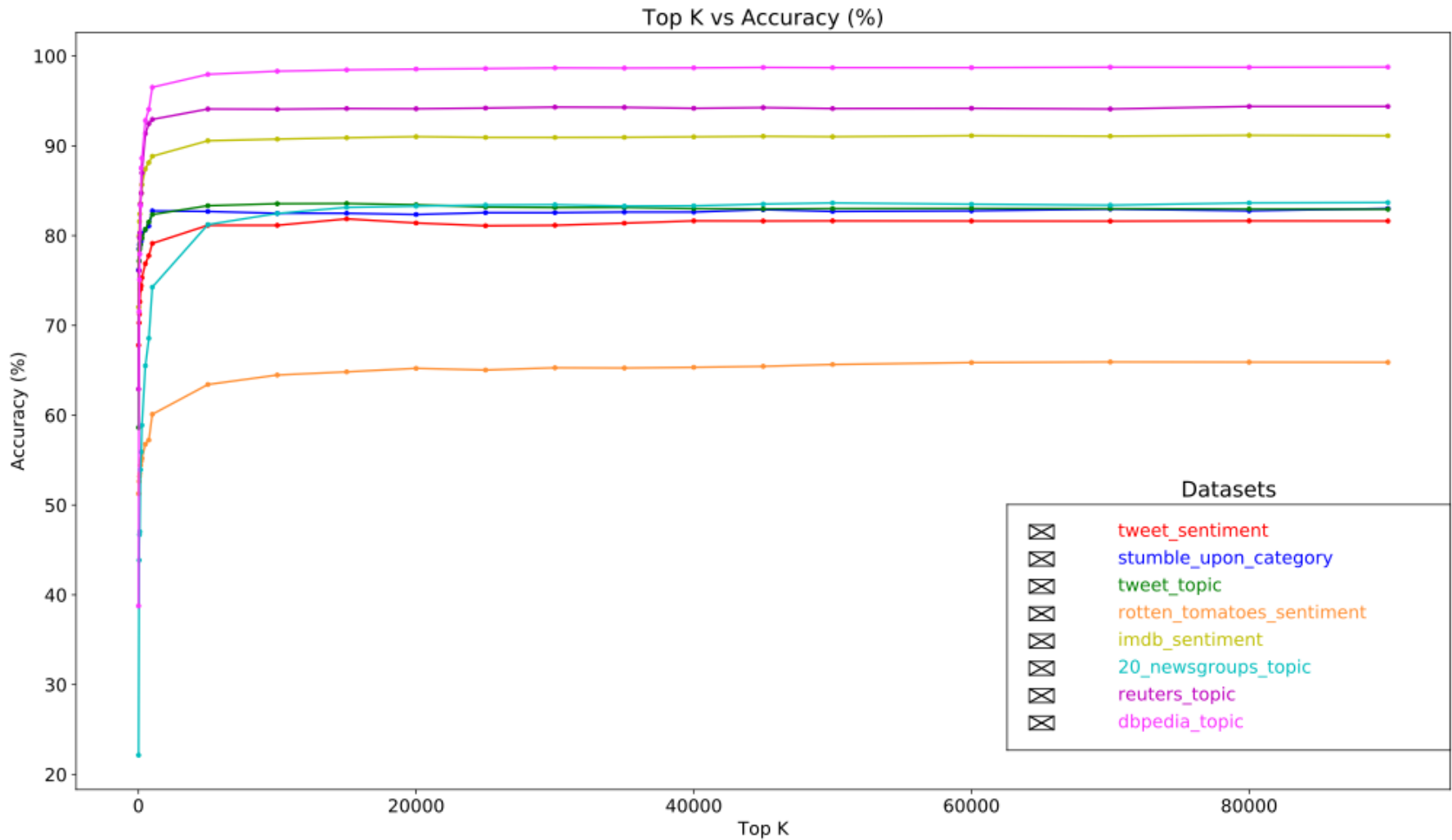
Country-Capital

Word embeddings



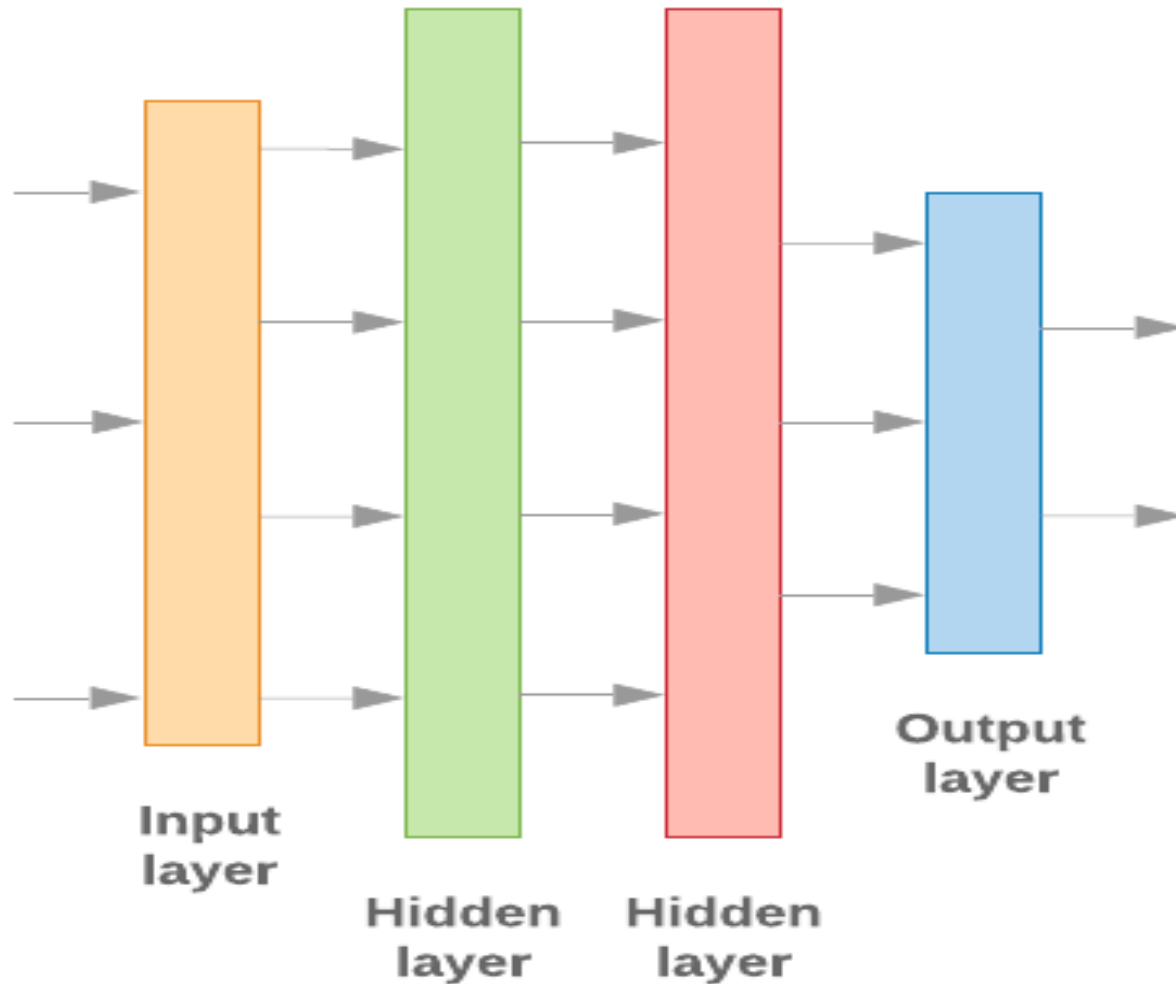
Text Classification

Top K Features (20K) vs Accuracy



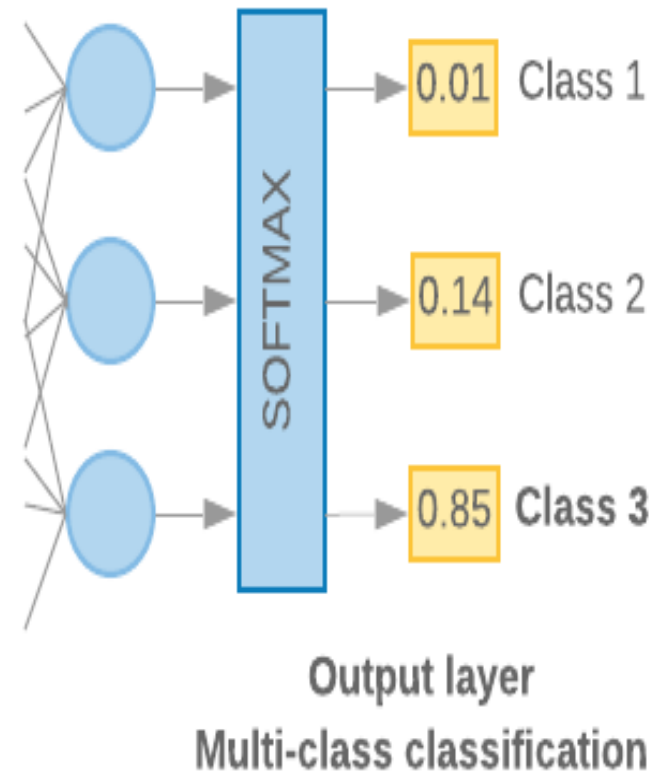
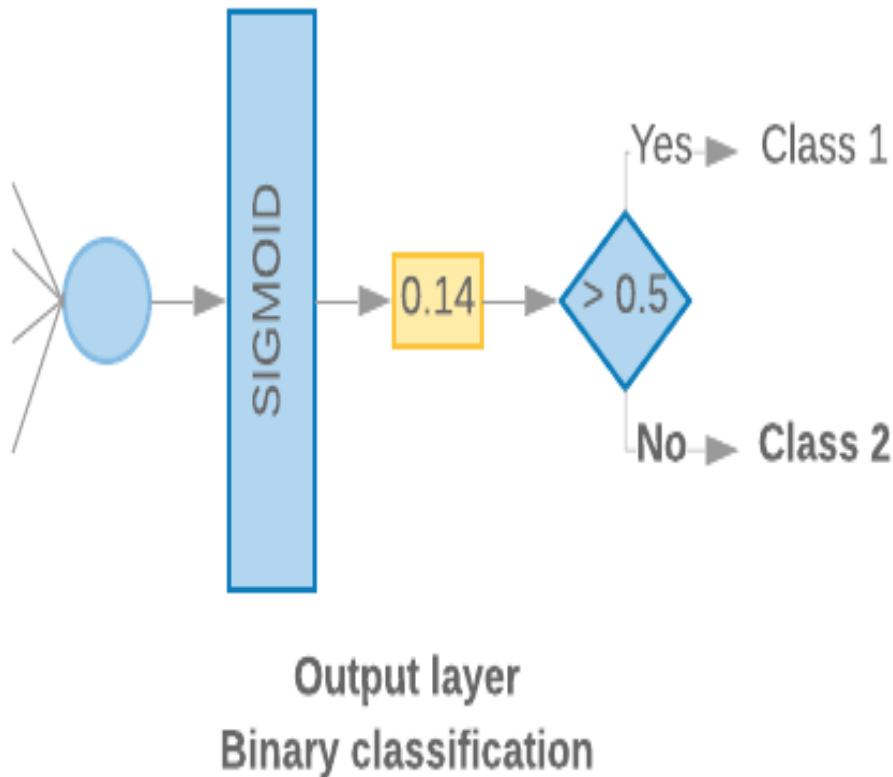
Text Classification

Linear stack of layers

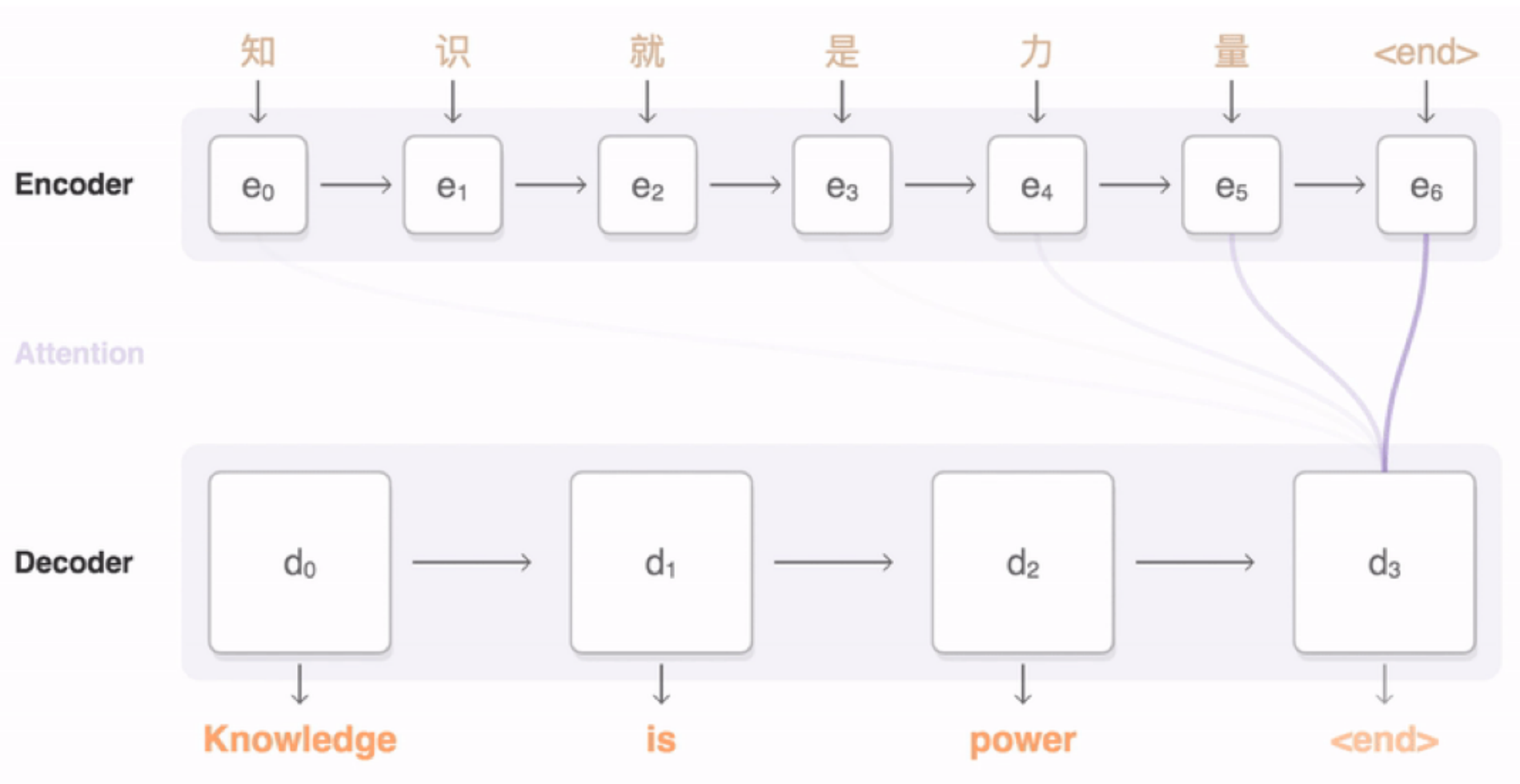


Text Classification

Last layer

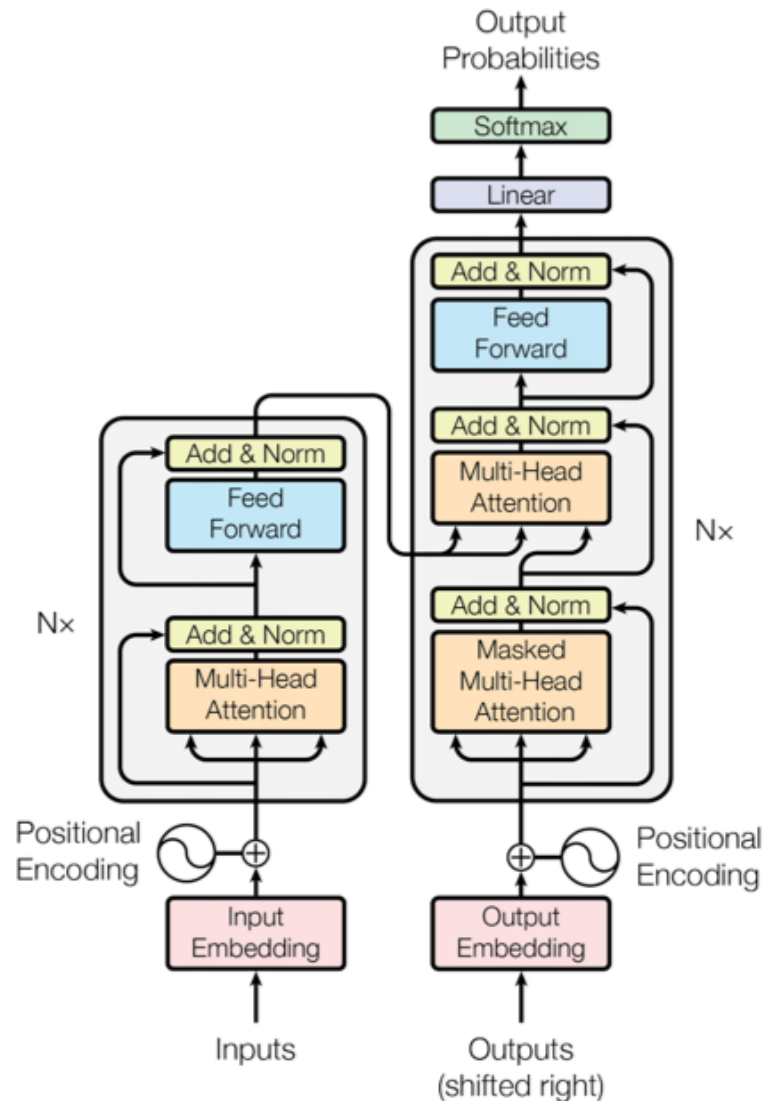


Sequence to Sequence (Seq2Seq)



Transformer (Attention is All You Need)

(Vaswani et al., 2017)



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

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

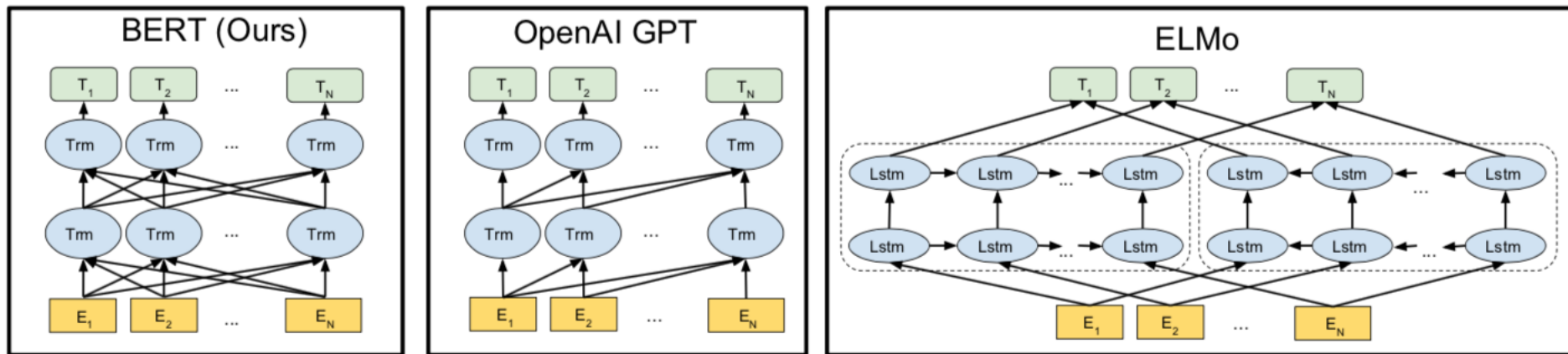
Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova

Google AI Language

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

BERT

Bidirectional Encoder Representations from Transformers



Pre-training model architectures

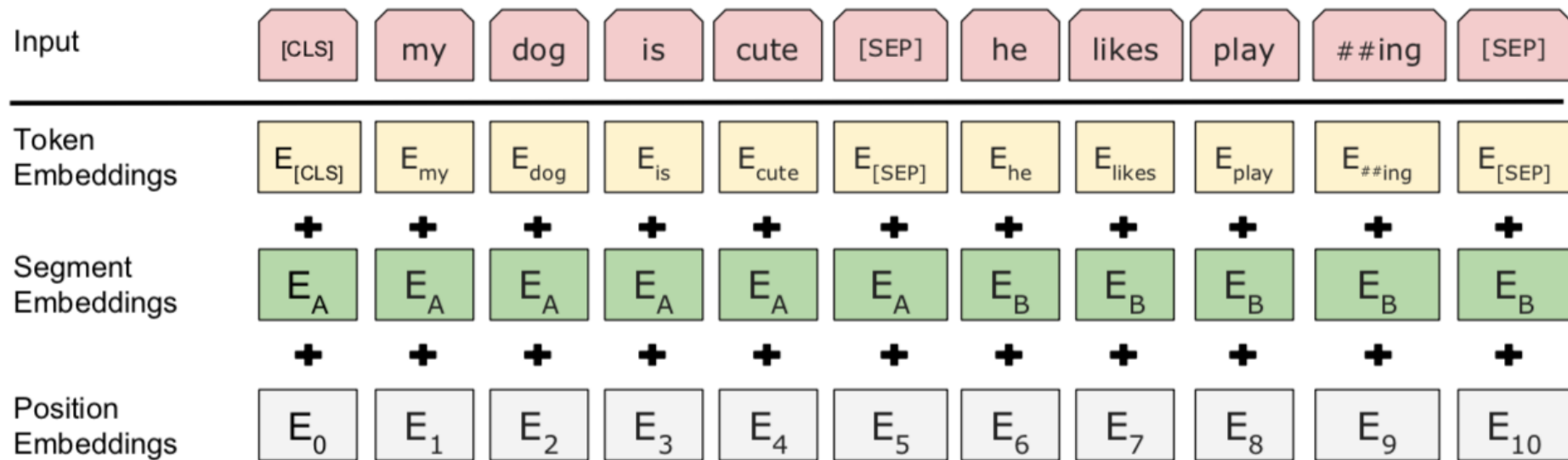
BERT uses a bidirectional Transformer.

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.

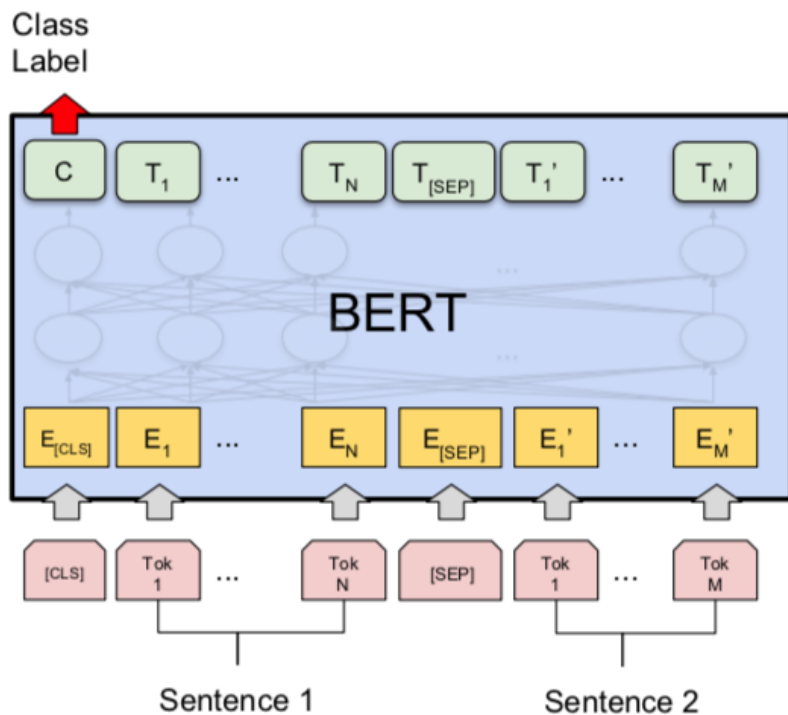
Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

BERT input representation

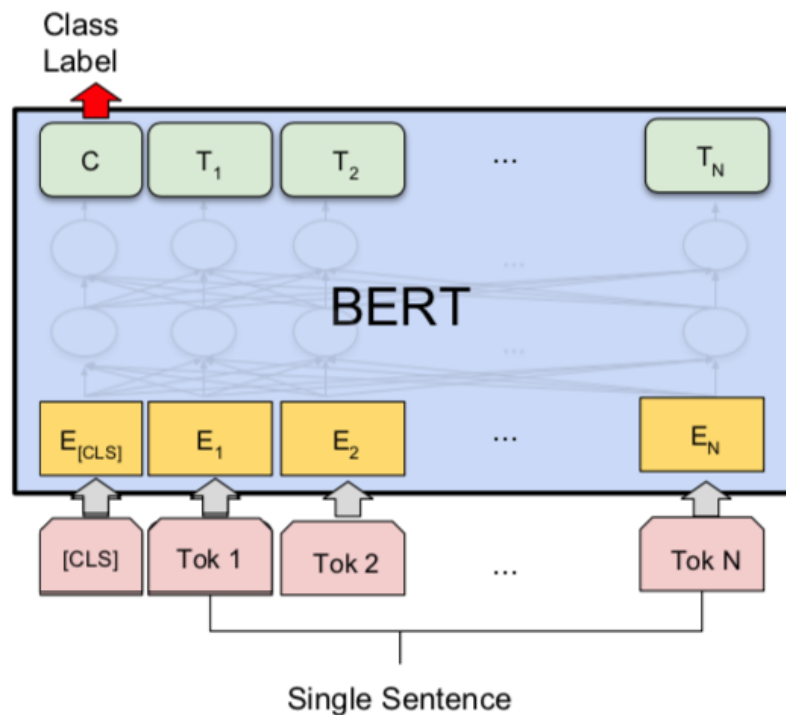


The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

BERT Sequence-level tasks

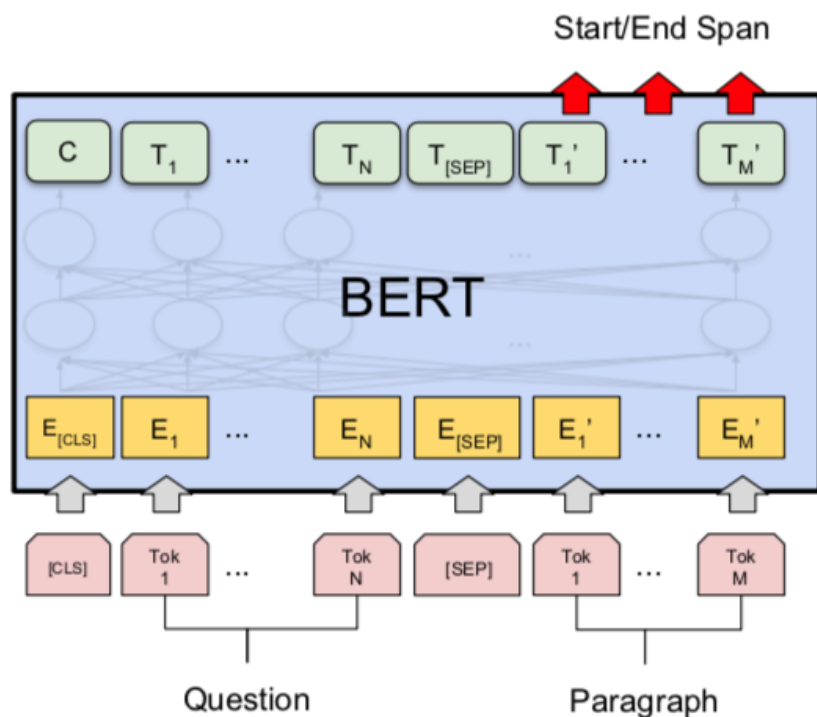


(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG

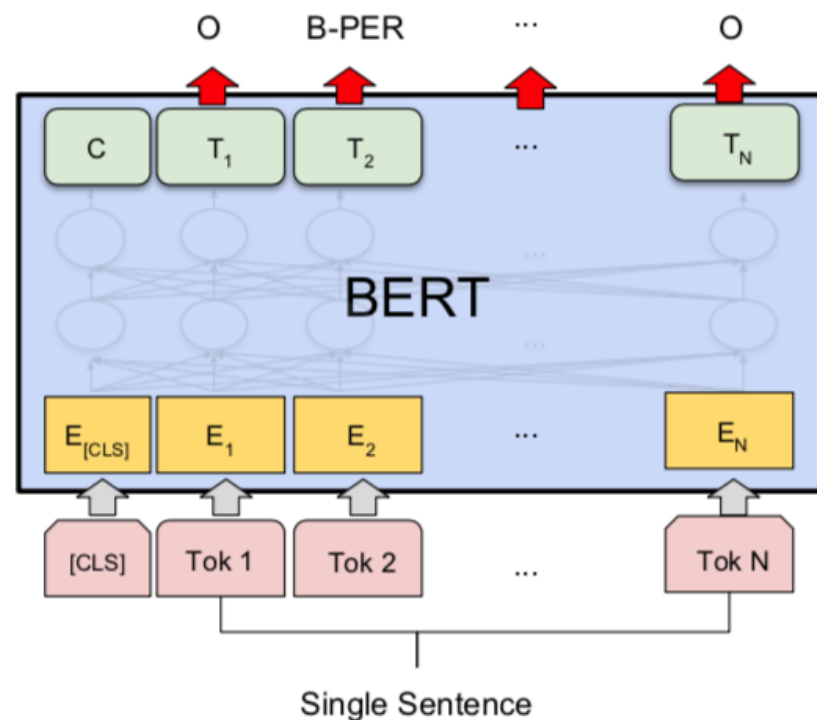


(b) Single Sentence Classification Tasks:
SST-2, CoLA

BERT Token-level tasks



(c) Question Answering Tasks:
SQuAD v1.1



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

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

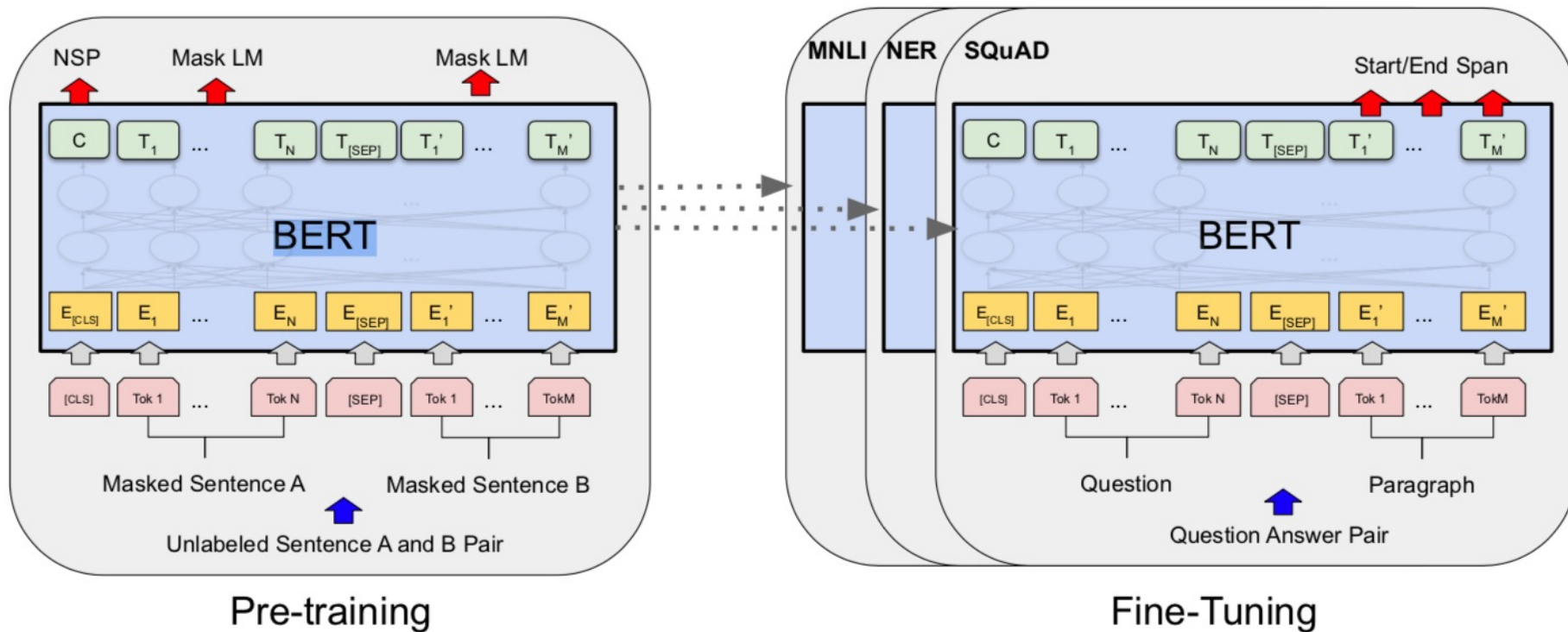
Transfer Learning in Natural Language Processing

Source: Sebastian Ruder, Matthew E. Peters, Swabha Swayamdipta, and Thomas Wolf (2019), "Transfer learning in natural language processing." In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials, pp. 15-18.

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

BERT (Bidirectional Encoder Representations from Transformers)

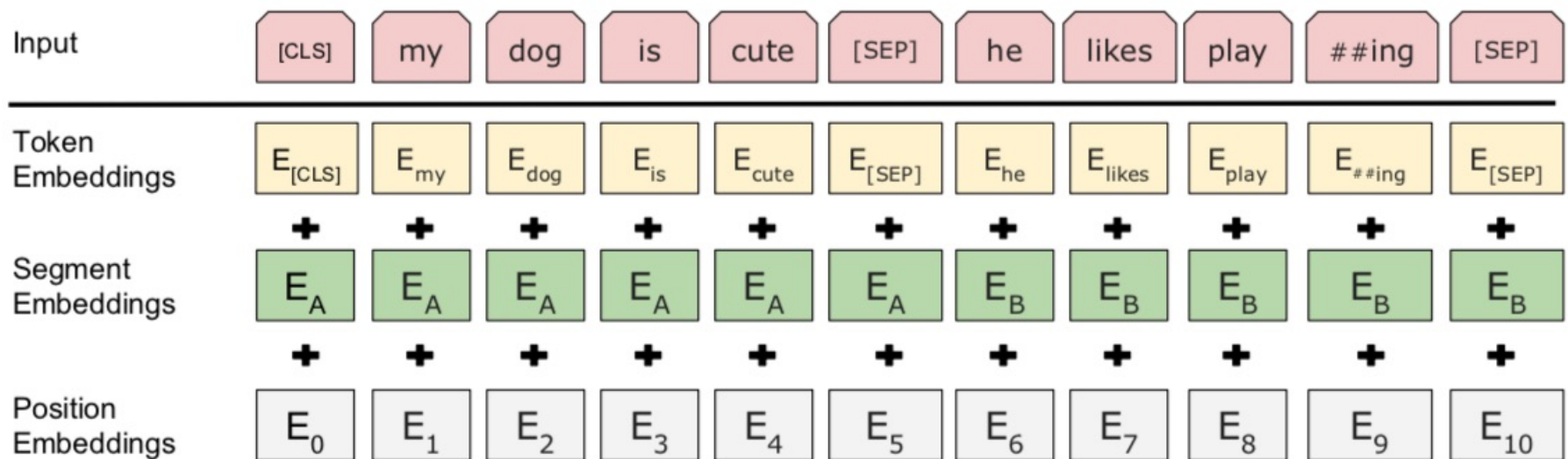
Overall pre-training and fine-tuning procedures for BERT



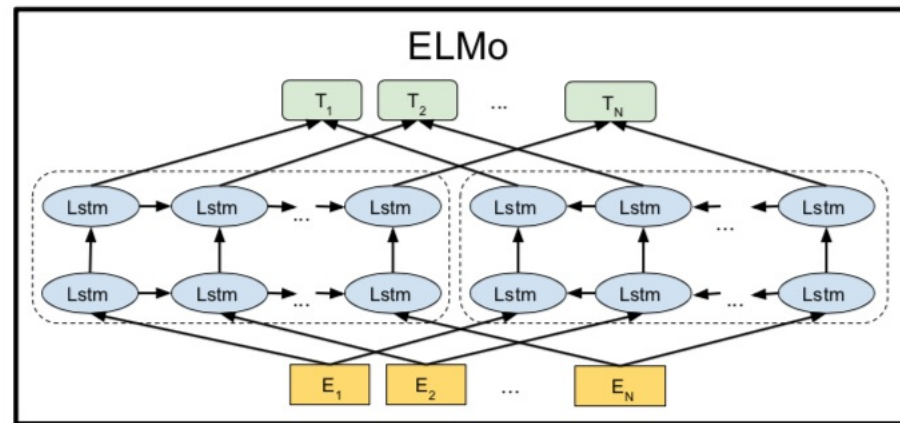
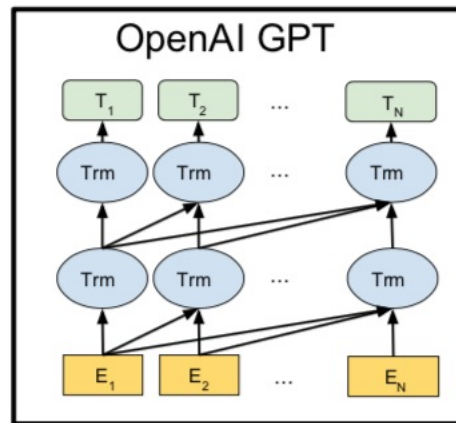
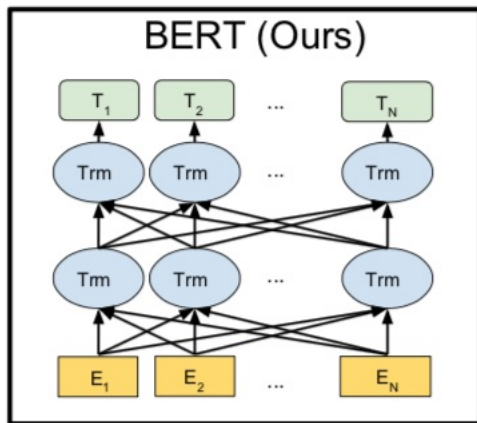
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT (Bidirectional Encoder Representations from Transformers)

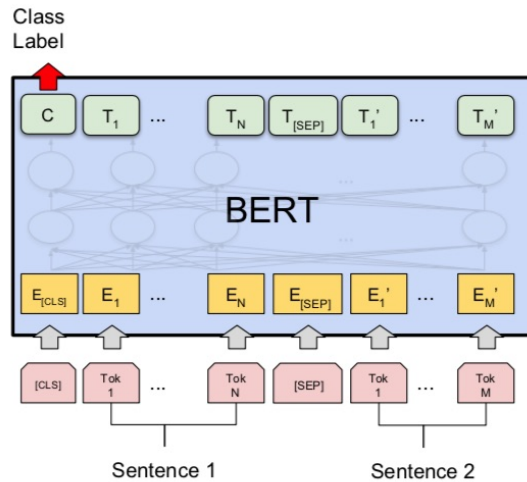
BERT input representation



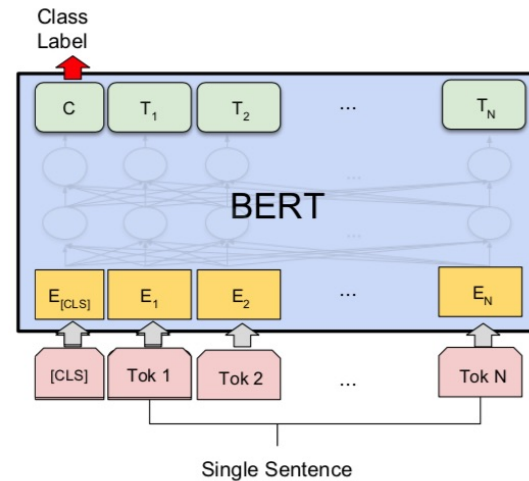
BERT, OpenAI GPT, ELMo



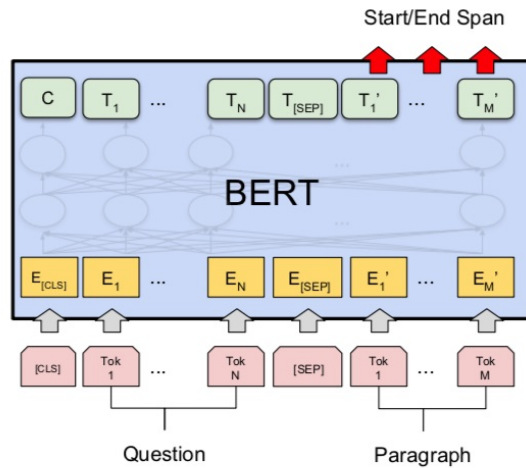
Fine-tuning BERT on Different Tasks



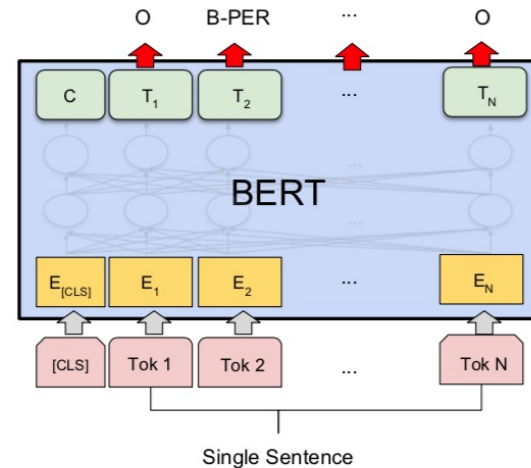
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



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



(c) Question Answering Tasks:
SQuAD v1.1

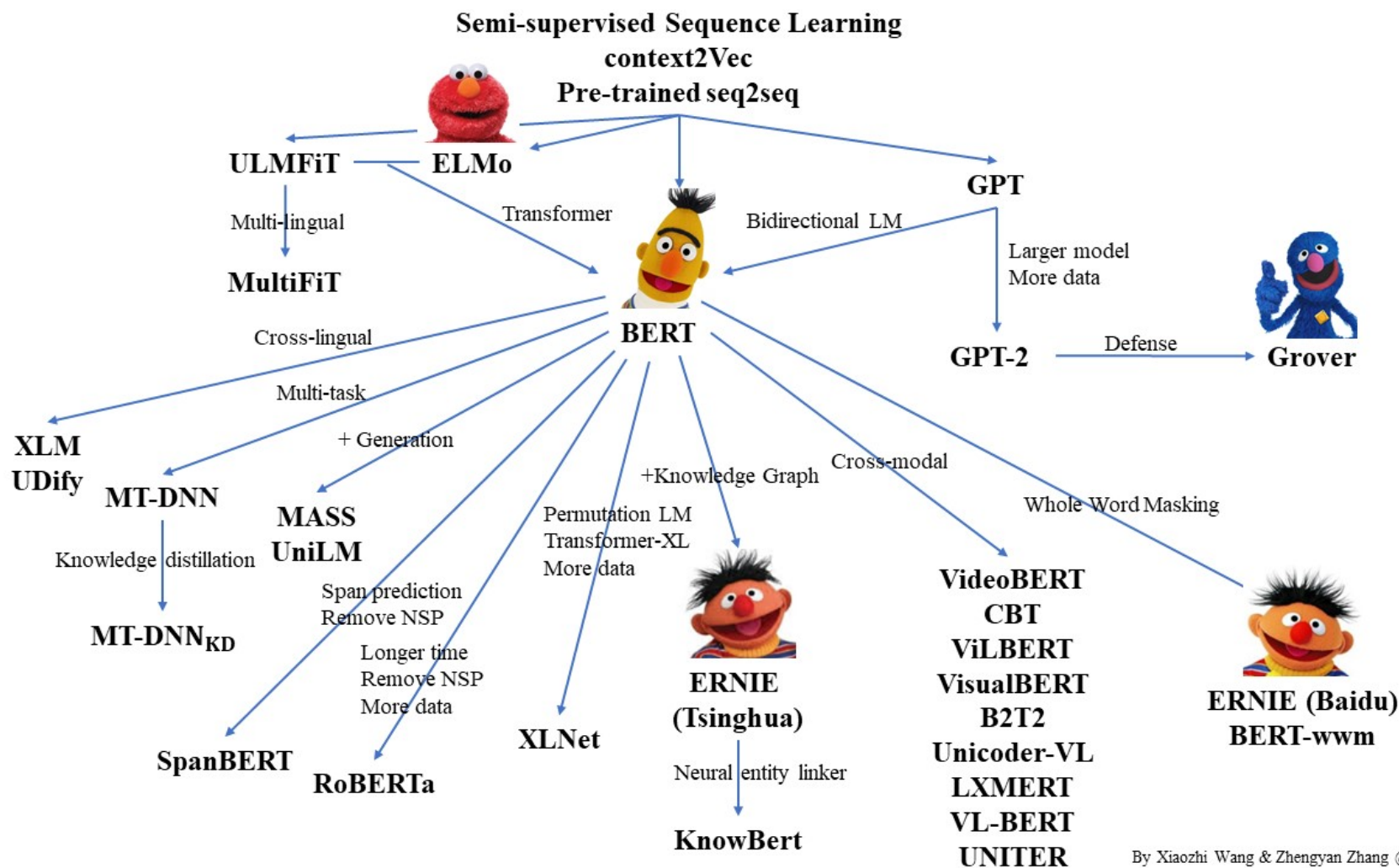


(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

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

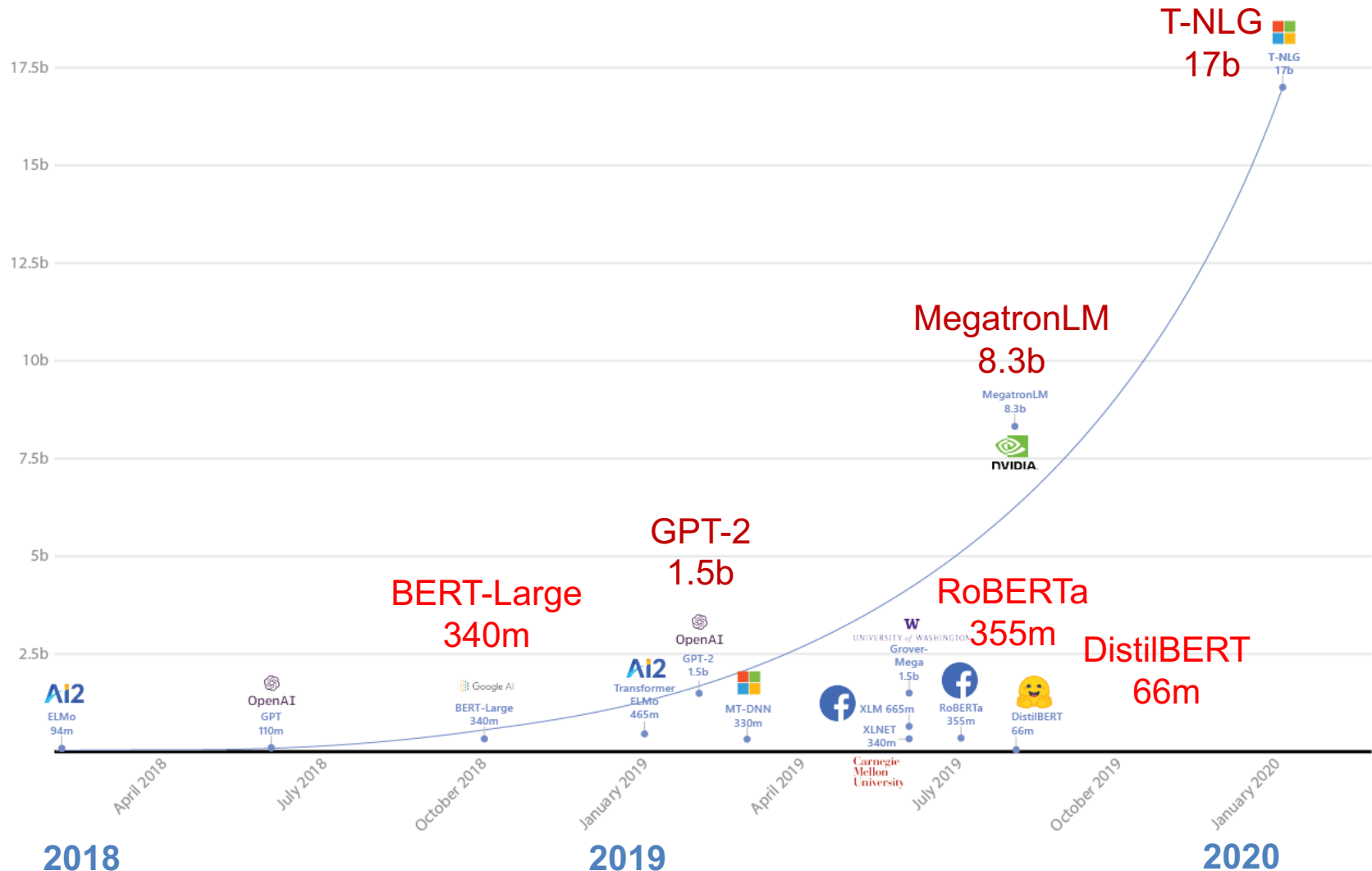
"Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

Pre-trained Language Model (PLM)



By Xiaozhi Wang & Zhengyan Zhang @THUNLP

Turing Natural Language Generation (T-NLG)



Source: <https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/>

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/csml_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

A Visual Guide to Using BERT for the First Time

(Jay Alammar, 2019)

“a visually stunning
rumination on love”

Reviewer #1

That’s a **positive** thing to say



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

Reviewer #2

That’s **negative**

Sentiment Classification: SST2

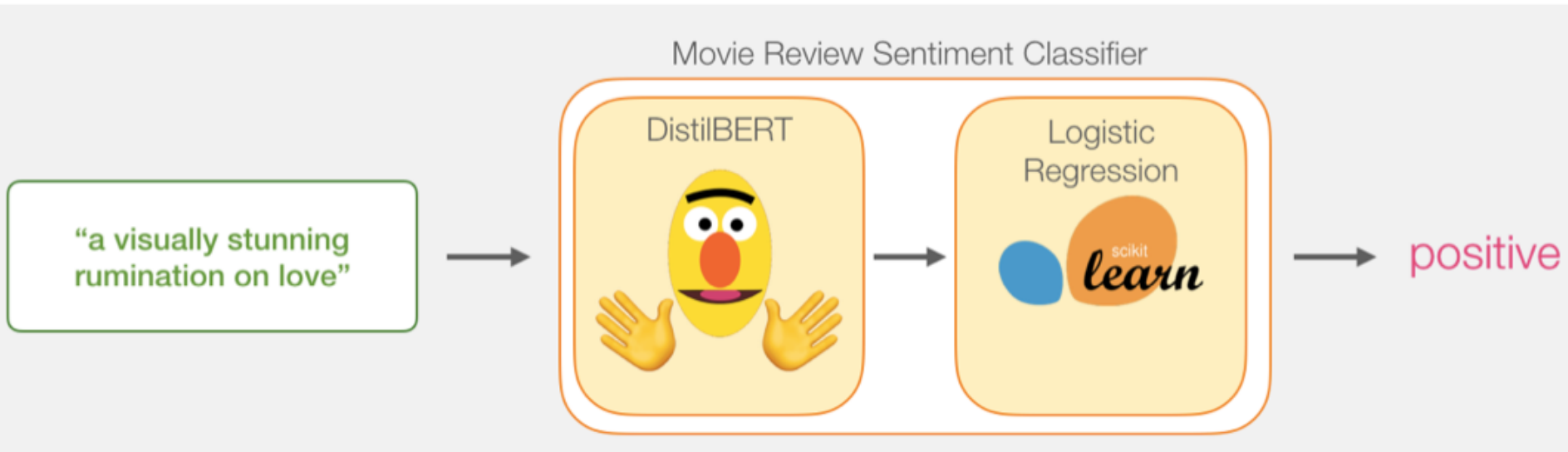
Sentences from movie reviews

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

Movie Review Sentiment Classifier



Movie Review Sentiment Classifier



Movie Review Sentiment Classifier

Model Training

Movie Review Sentiment Classifier

DistilBERT

Already (pre-)trained



Logistic
Regression

We will train in this tutorial

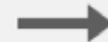
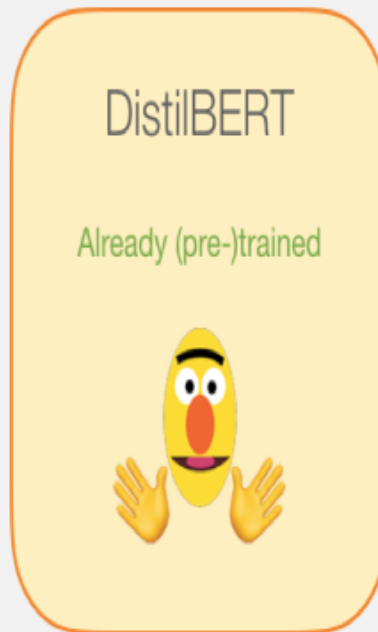
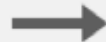


Step # 1 Use distilBERT to Generate Sentence Embeddings

Step #1: Use DistilBERT to embed all the sentences

Sentence label

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

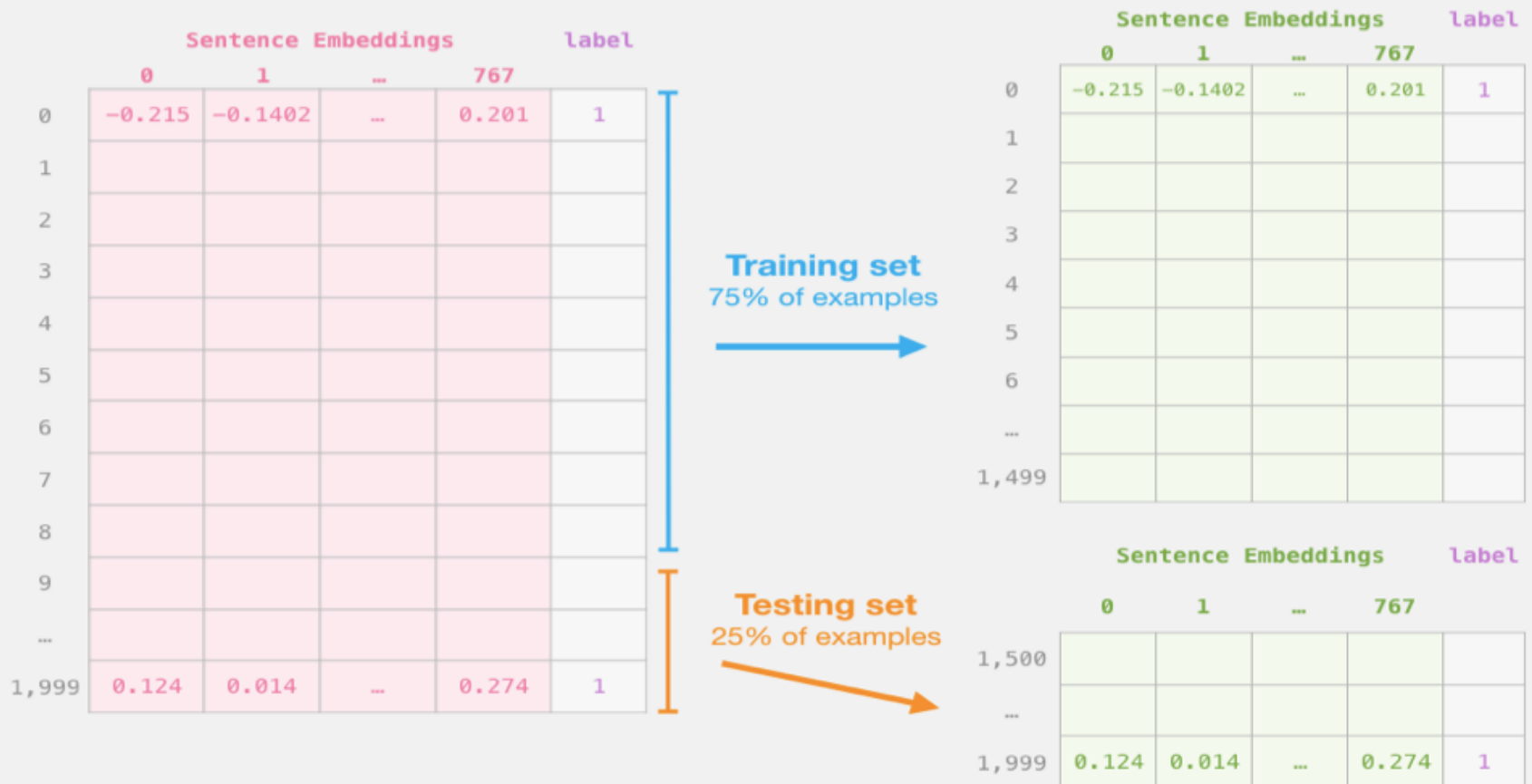


Sentence Embeddings label

	0	1	...	767	
0	-0.215	-0.1402	...	0.201	1
1	-0.172	-0.144	...	0.371	0
...	...	...	...	...	...
1,999	0.124	0.014	...	0.274	1

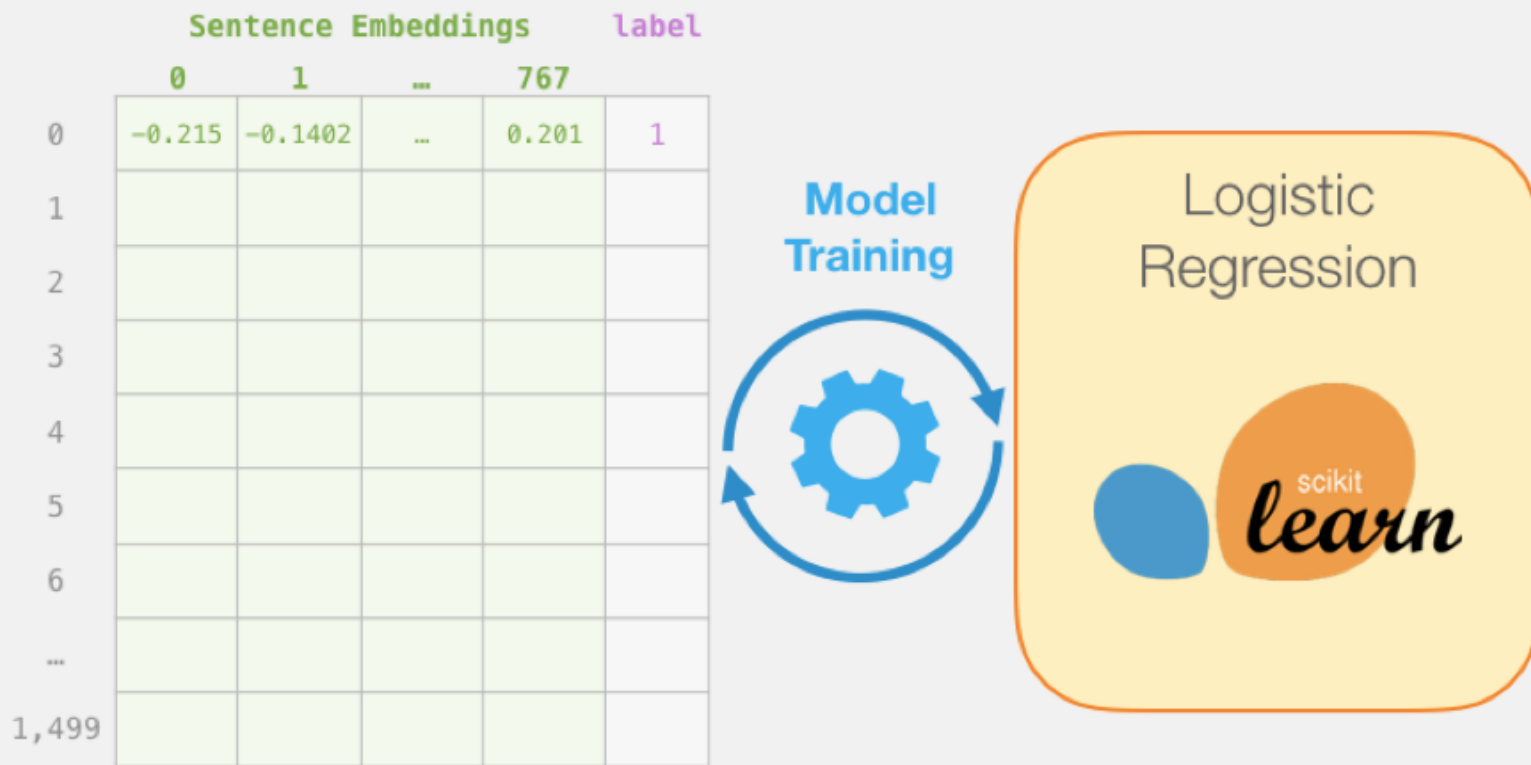
Step #2: Test/Train Split for Model #2, Logistic Regression

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



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

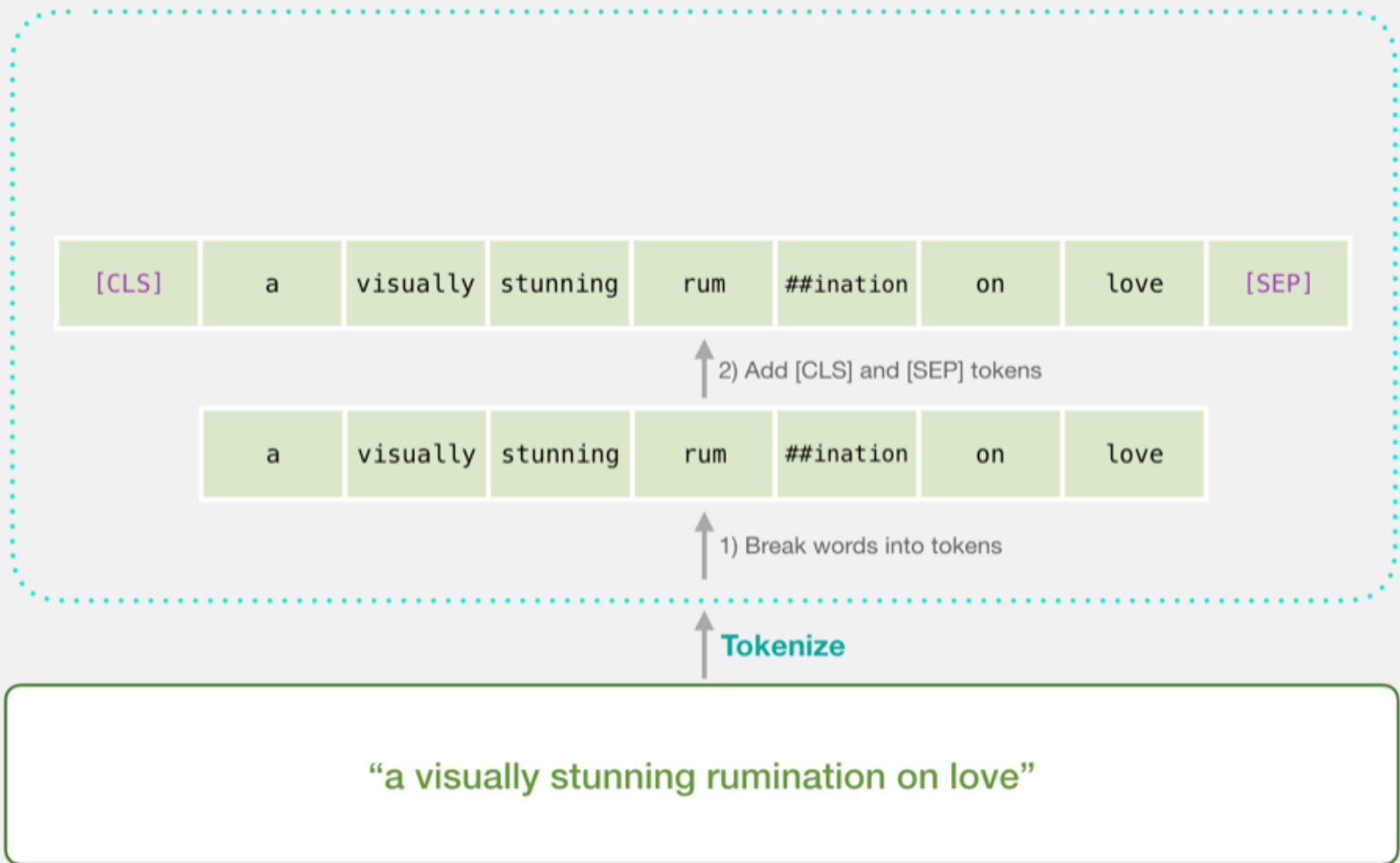
Step #3: Train the logistic regression model using the training set



Tokenization

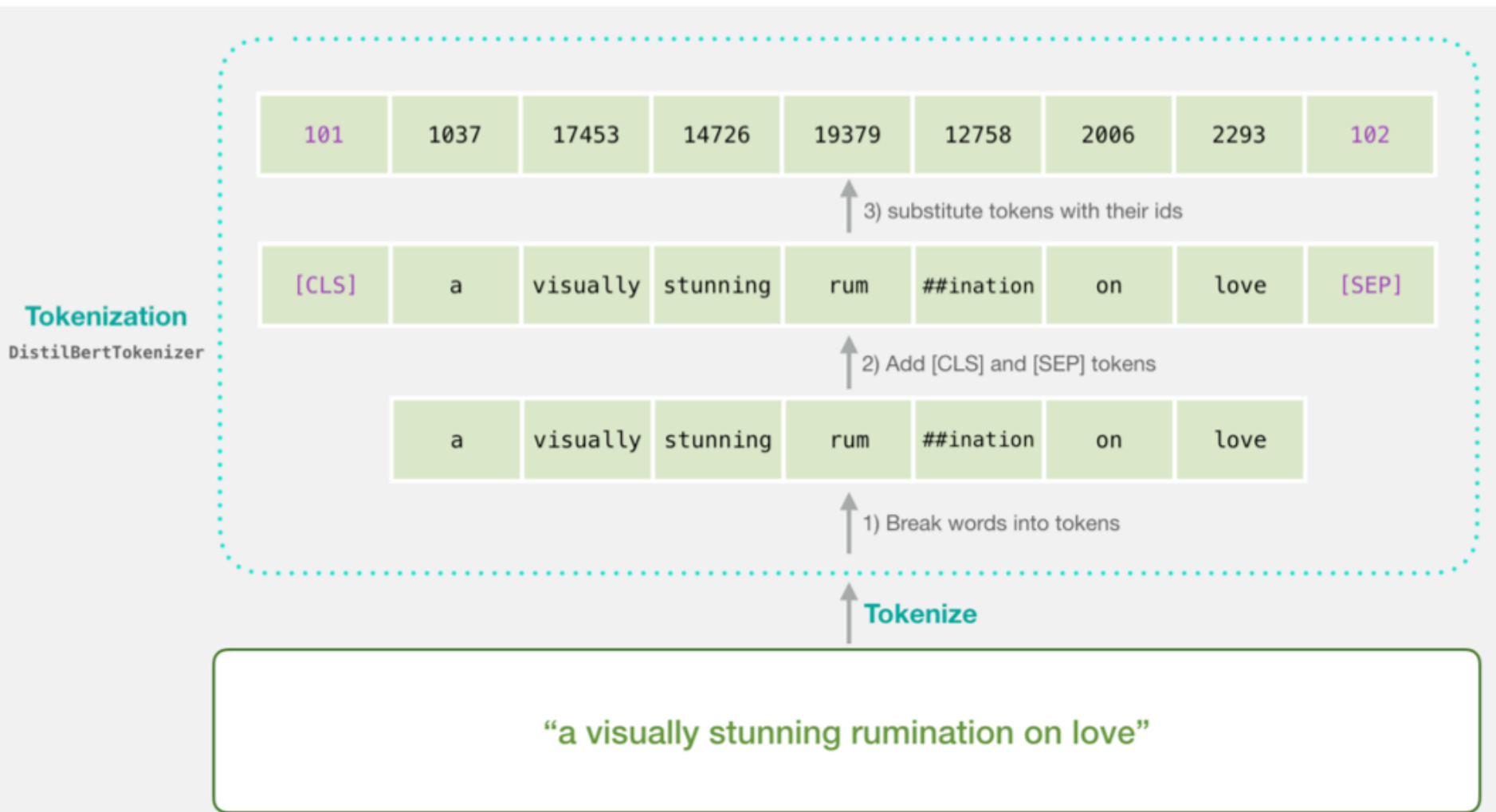
[CLS] a visually stunning rum ##ination on love [SEP]
a visually stunning rumination on love

Tokenization
DistilBertTokenizer

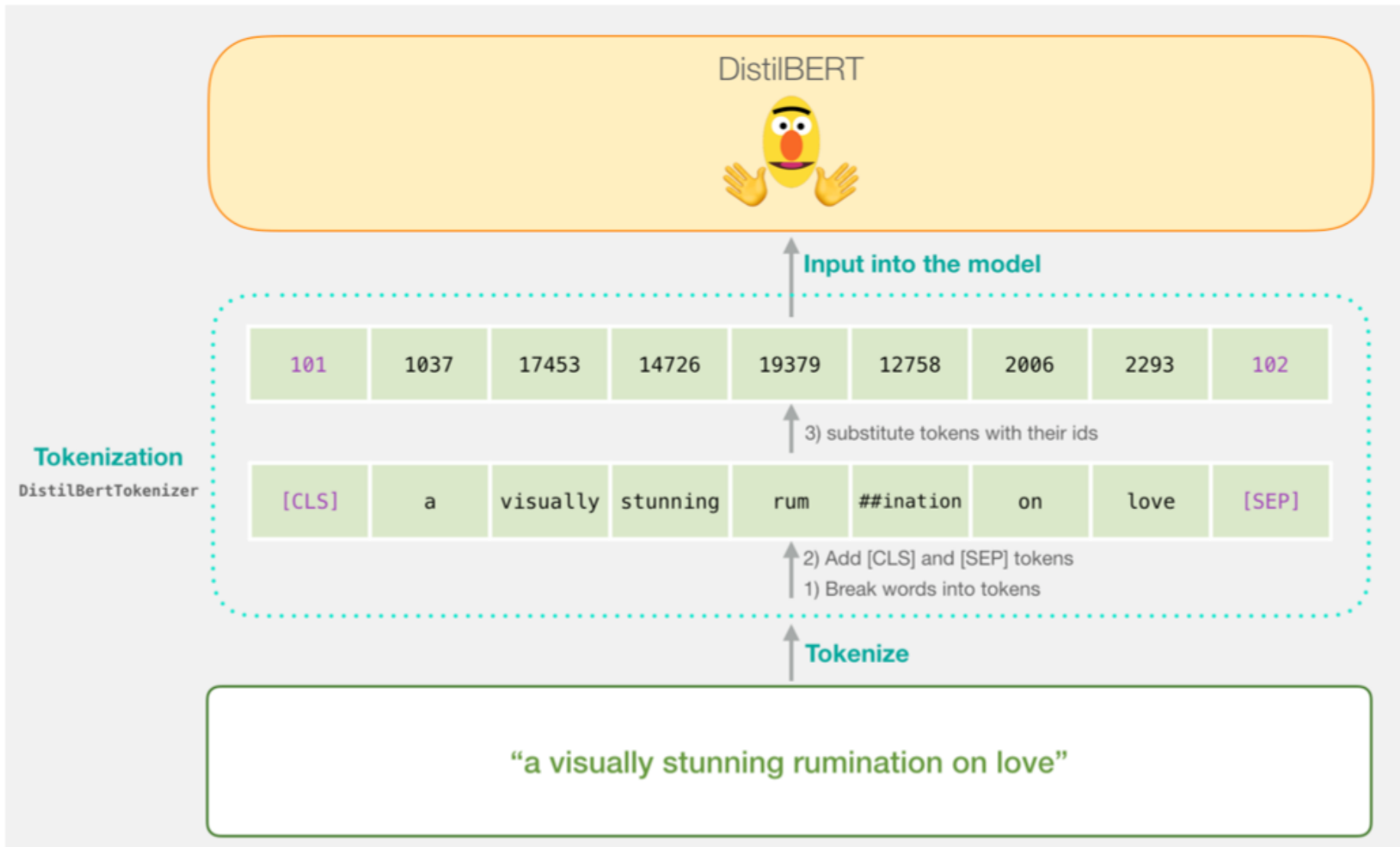


Tokenization

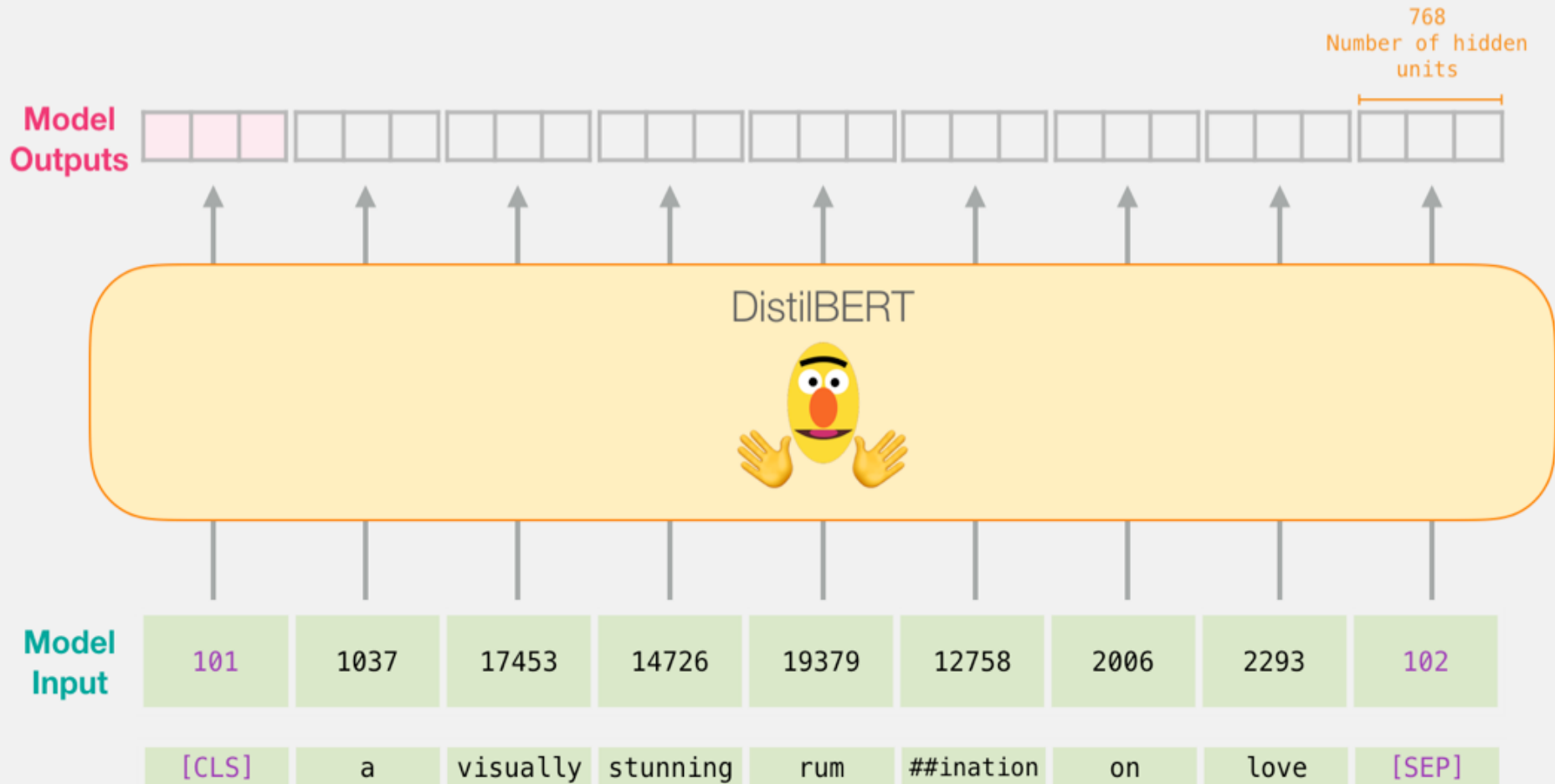
```
tokenizer.encode("a visually stunning rumination on love",  
                add_special_tokens=True)
```



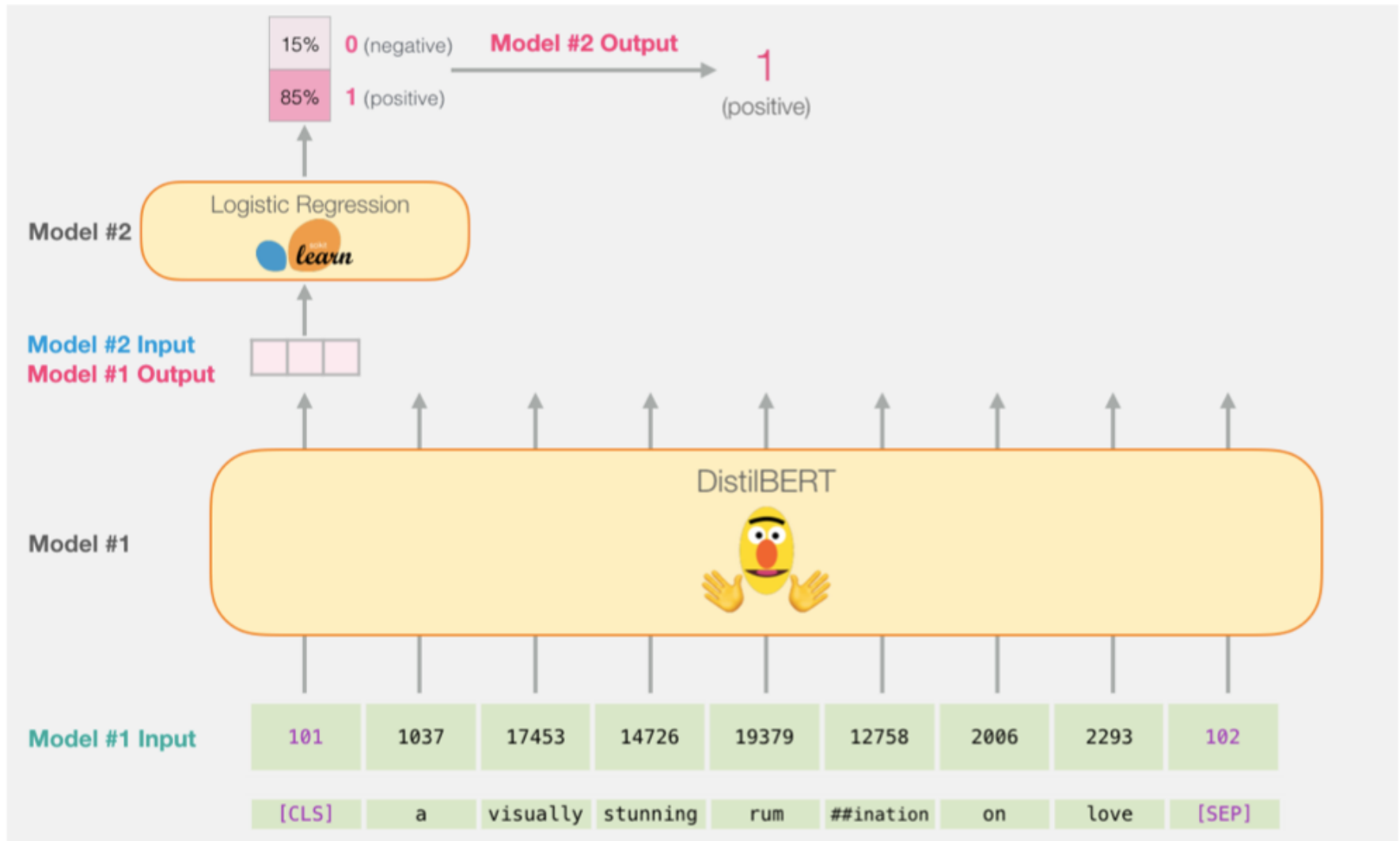
Tokenization for BERT Model



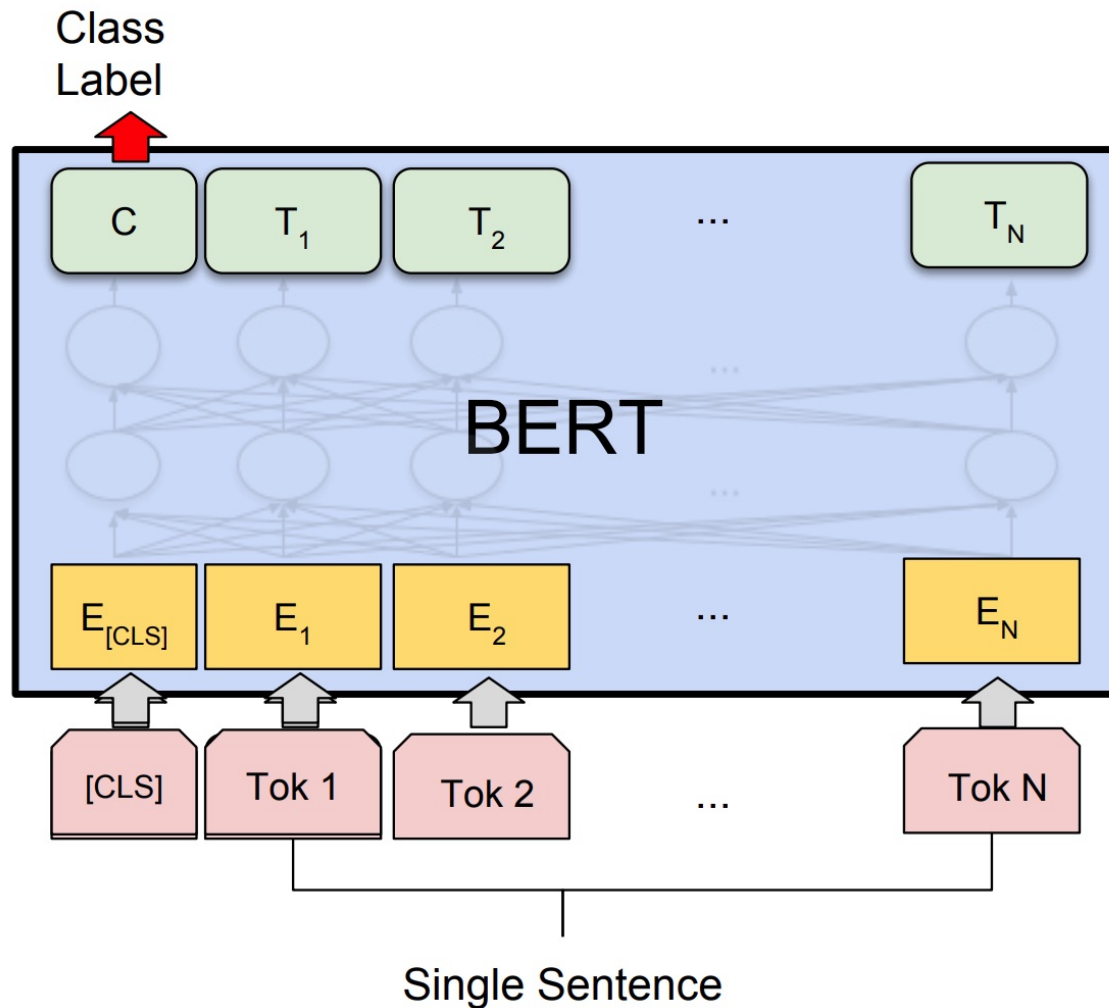
Flowing Through DistilBERT (768 features)



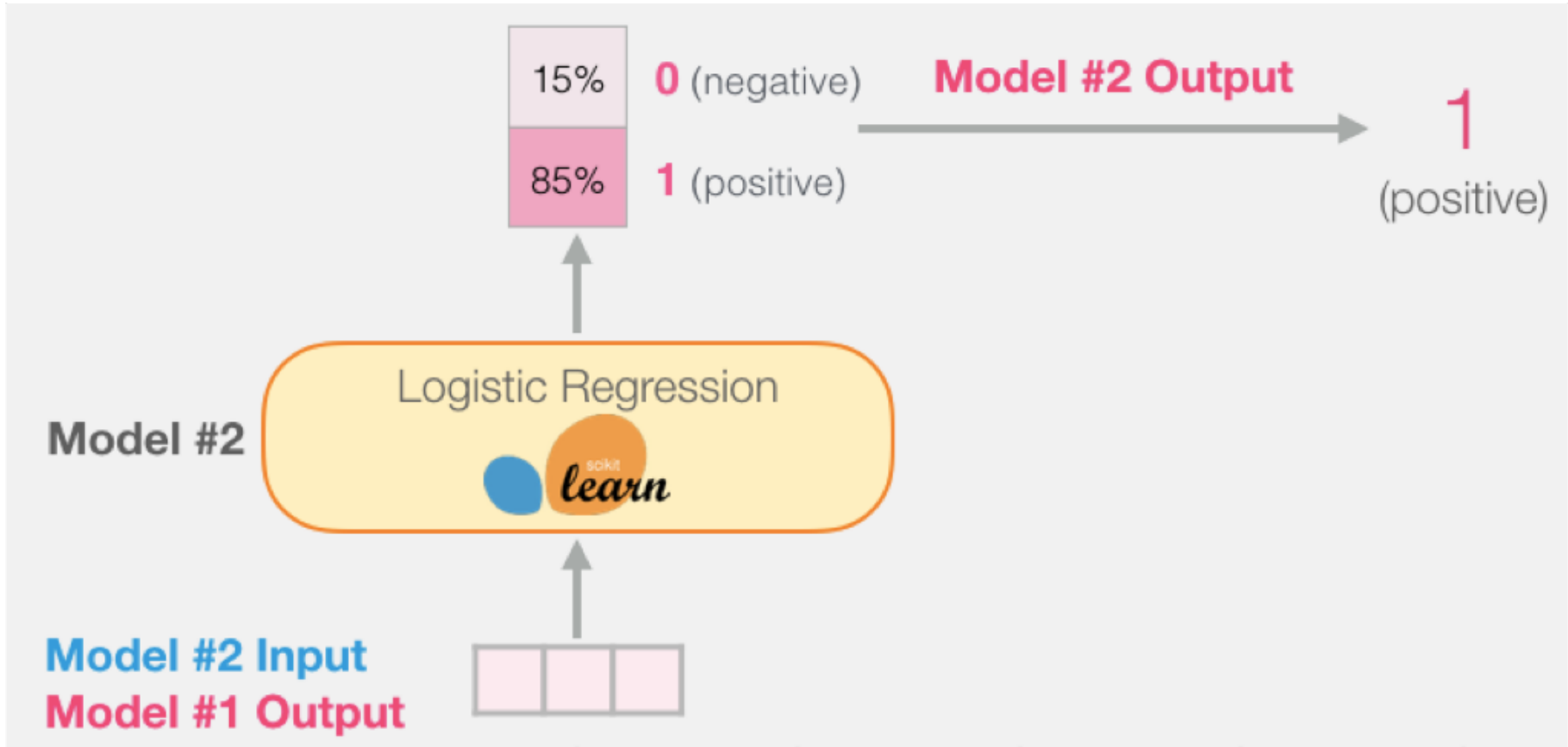
Model #1 Output Class vector as Model #2 Input



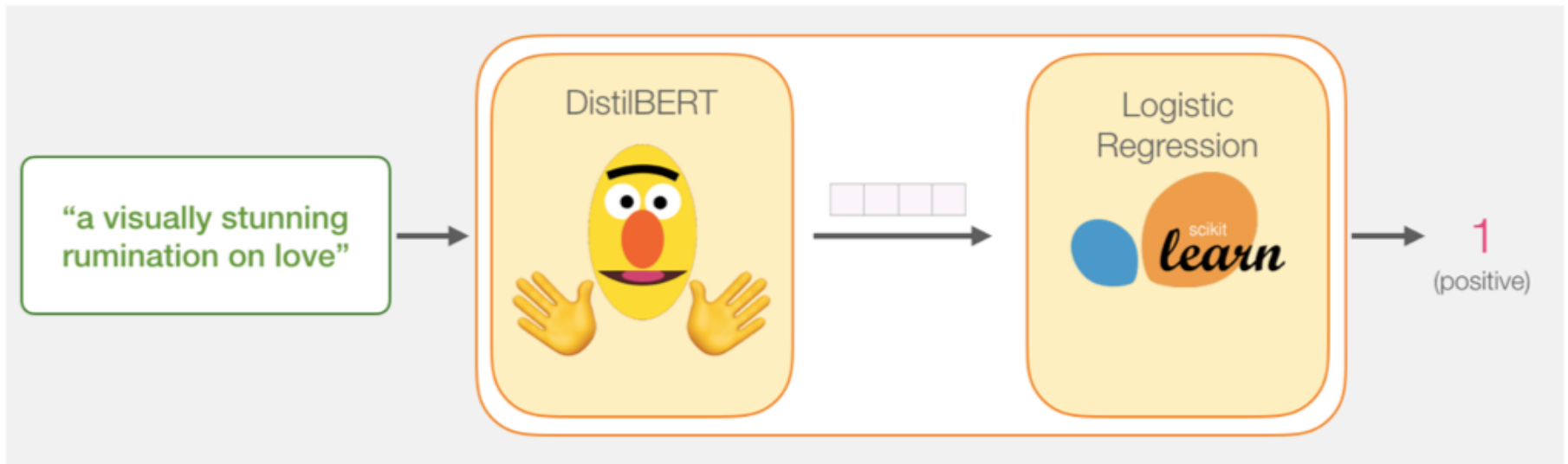
Fine-tuning BERT on Single Sentence Classification Tasks



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)))
```

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

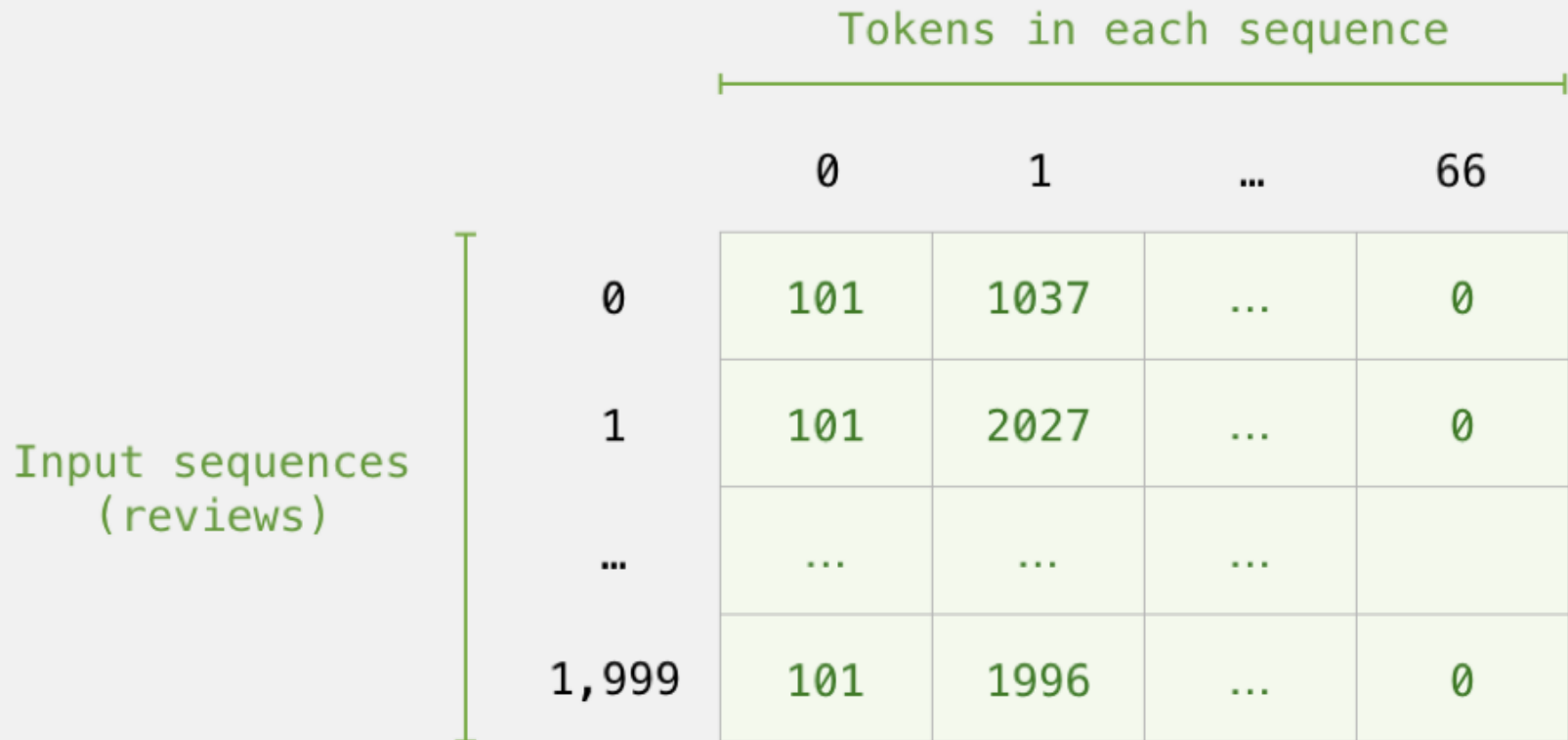


Sequences of Token IDs

```
[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

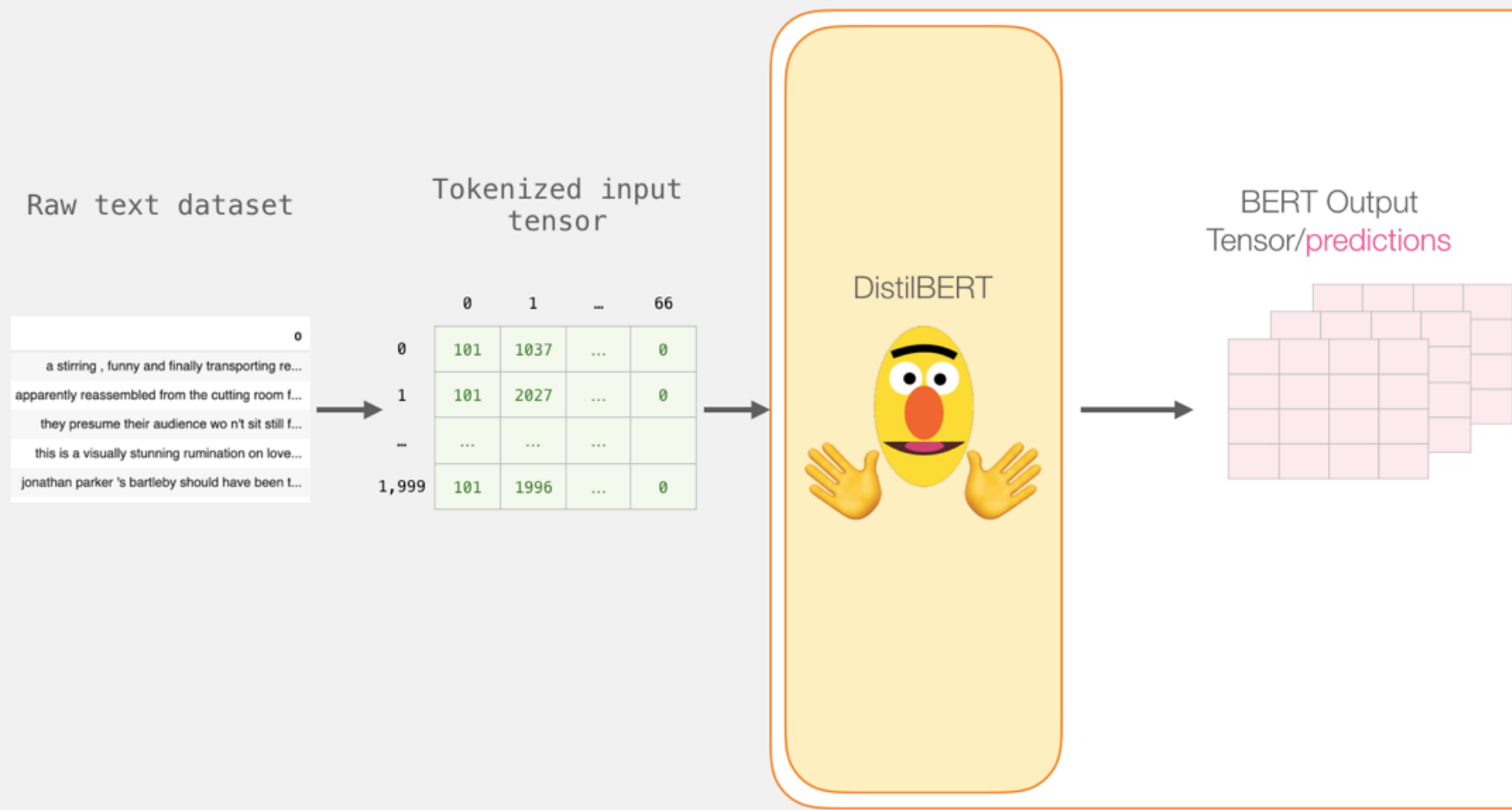


The diagram illustrates the structure of the BERT/DistilBERT input tensor. It features a grid of tokens with green text and a light green background. A vertical green bracket on the left is labeled 'Input sequences (reviews)', and a horizontal green bracket at the top is labeled 'Tokens in each sequence'. The grid has 2000 rows (indices 0 to 1,999) and 67 columns (indices 0 to 66). The first column (index 0) contains the token '101' for all rows. The second column (index 1) contains unique tokens for each row, such as '1037', '2027', and '1996'. The last column (index 66) contains the token '0' for all rows. Ellipses (...) are used to represent intermediate columns and rows.

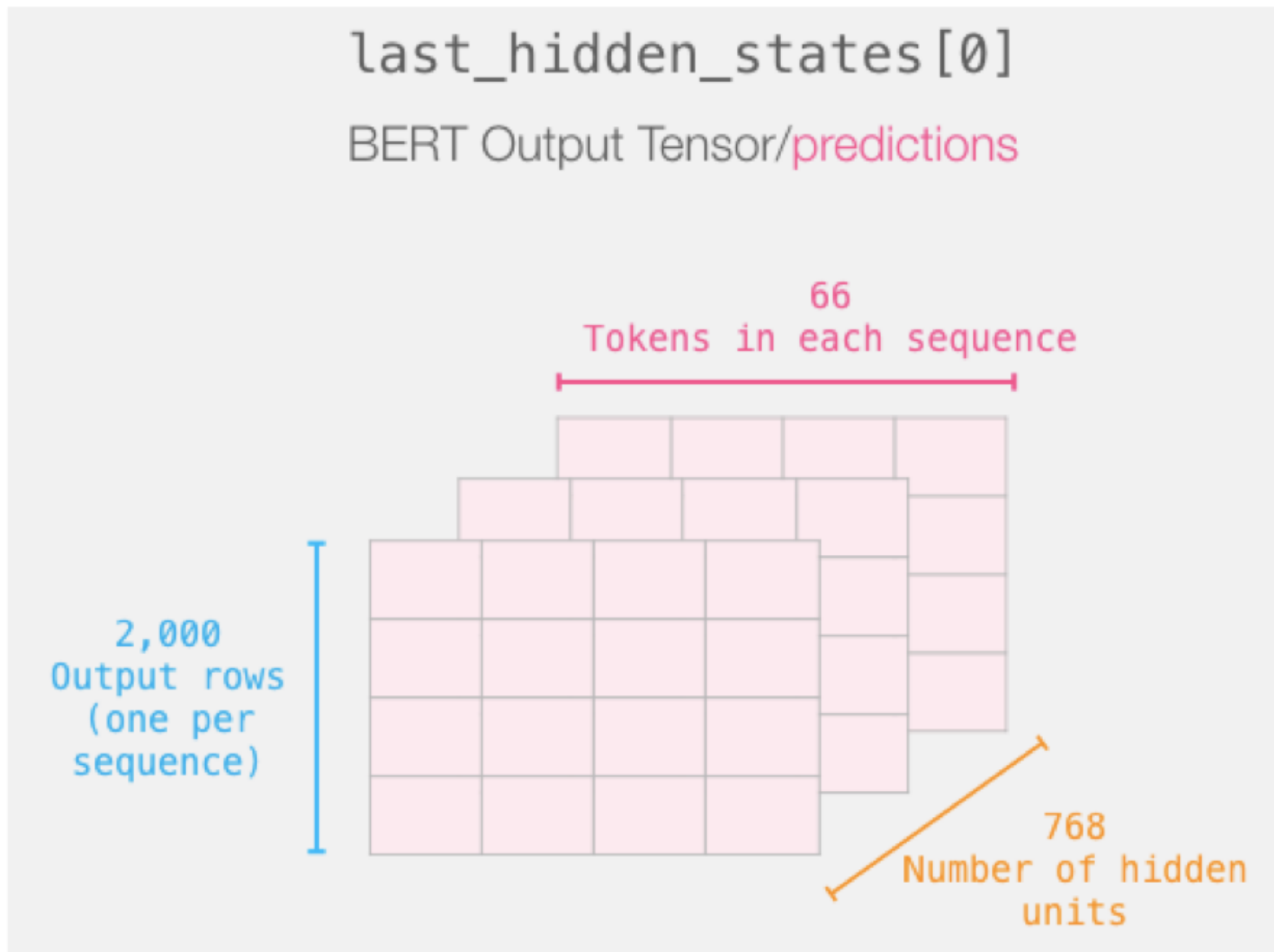
		Tokens in each sequence			
		0	1	...	66
Input sequences (reviews)	0	101	1037	...	0
	1	101	2027	...	0
	...	...	...	...	
	1,999	101	1996	...	0

Processing with DistilBERT

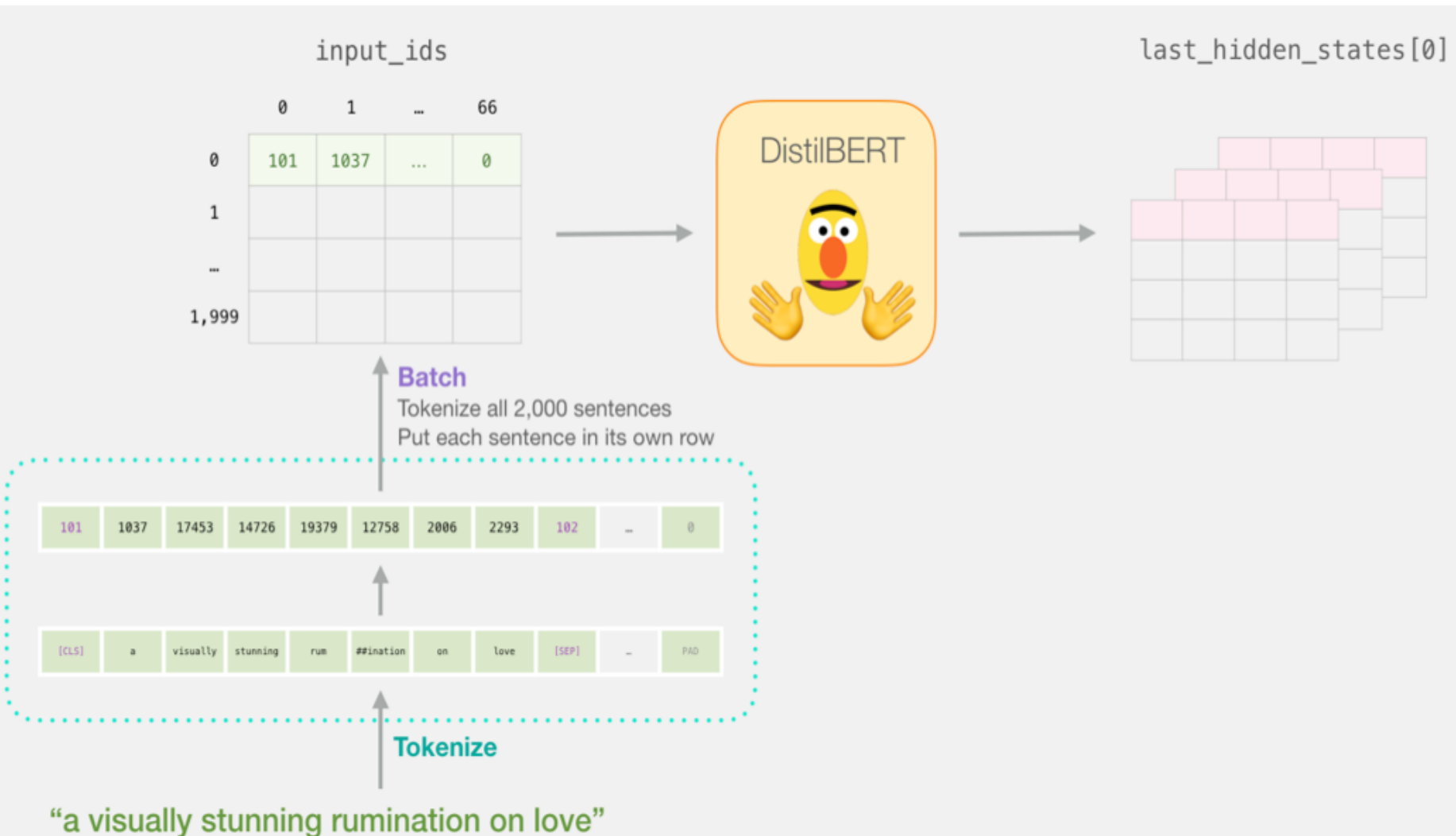
```
input_ids = torch.tensor(np.array(padded))  
last_hidden_states = model(input_ids)
```



Unpacking the BERT output tensor



Sentence to last_hidden_state[0]

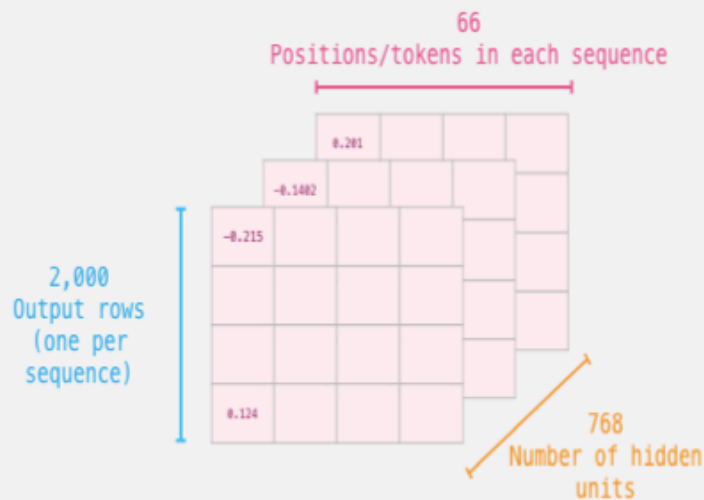


BERT's output for the [CLS] tokens

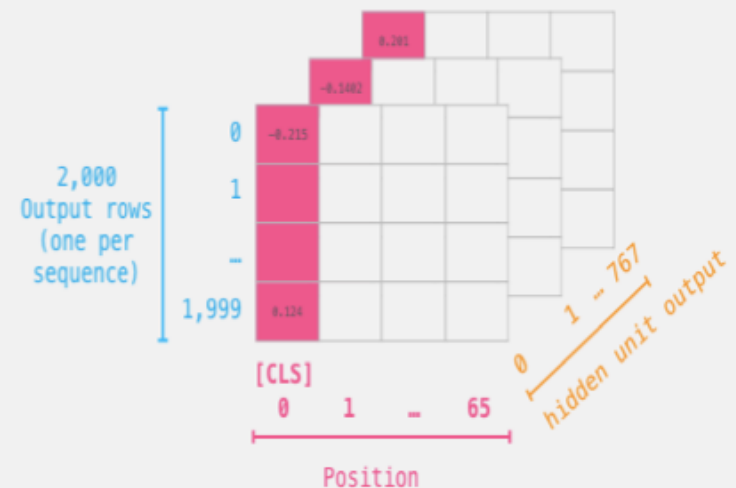
Slice the output for the first position for all the sequences, take all hidden unit outputs

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

`last_hidden_states[0]`
BERT Output Tensor/*predictions*

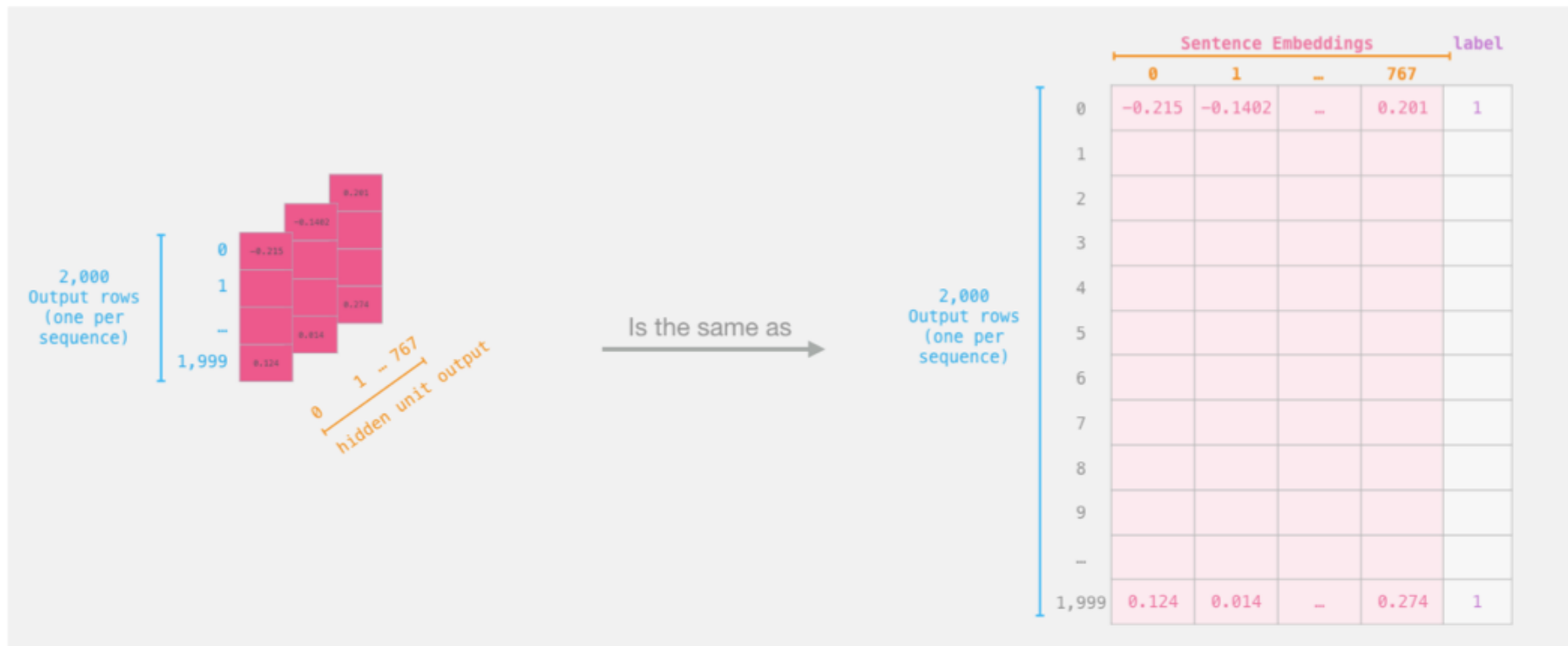


only the first position: [CLS]
`last_hidden_states[0][:,0,:]`
all sentences all hidden unit outputs



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)

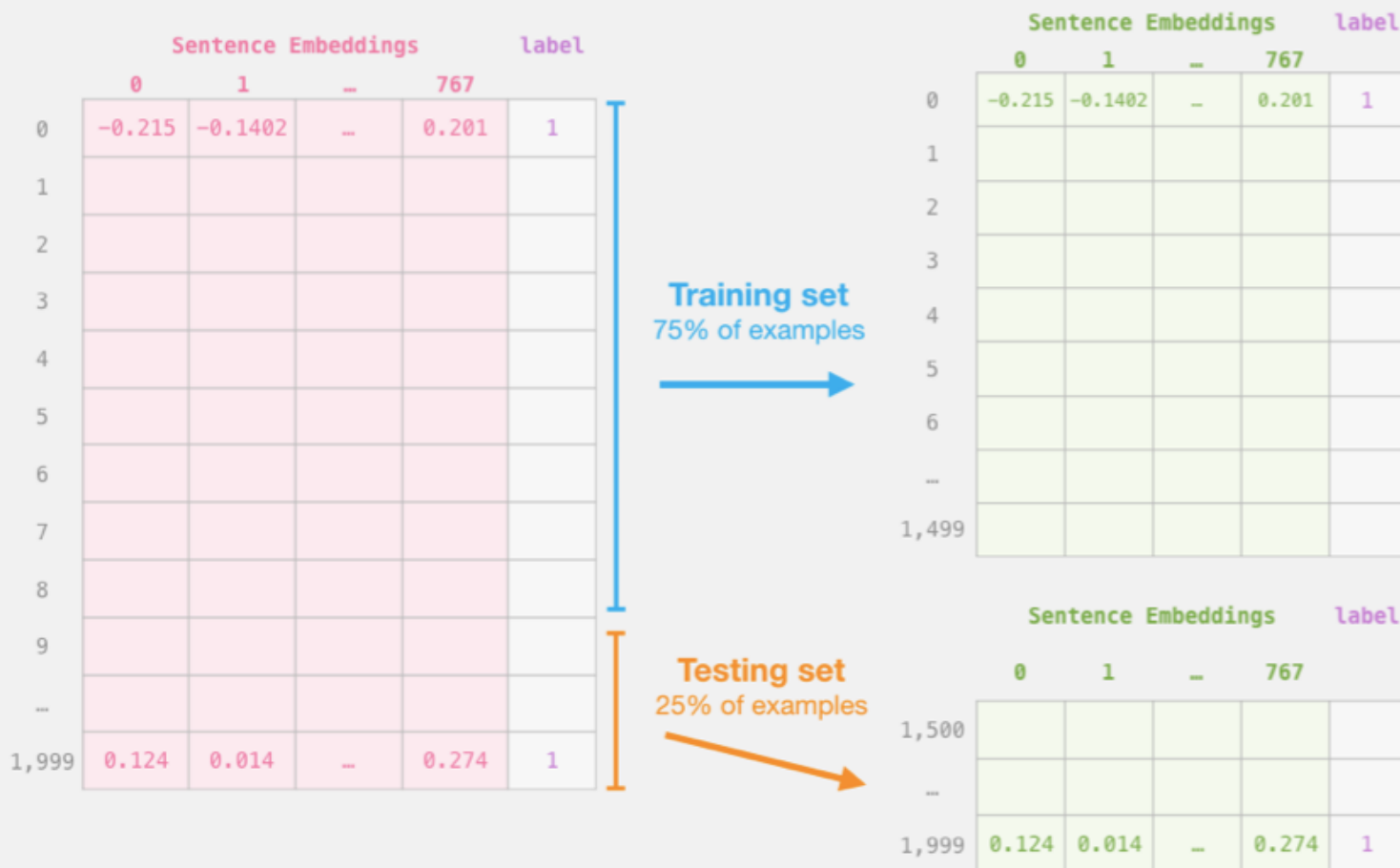
features				label
	0	1	...	
0				1
1				0
...				
1,999				1

```

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

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

Share

+ Code + Text Copy to Drive

Connect Editing

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

“a visually stunning
rumination on love”

Reviewer #1

That’s a **positive** thing to say

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

Reviewer #2

That’s **negative**

https://colab.research.google.com/github/jalammar/jalammar.github.io/blob/master/notebooks/bert/A_Visual_Notebook_to_Using_BERT_for_the_First_Time.ipynb

Text classification with preprocessed text: Movie reviews

 TensorFlow

Install

Learn ▾

API ▾

Resources ▾

More ▾

 Search

English ▾

GitHub

Sign in

TensorFlow Core

Overview

Tutorials

Guide

TF 1

TensorFlow tutorials

Quickstart for beginners

Quickstart for experts

BEGINNER

ML basics with Keras ▴

Basic image classification

Text classification with TF Hub

Text classification with preprocessed text

Regression

Overfit and underfit

Save and load

Load and preprocess data ▾

Estimator ▾

ADVANCED

Customization ▾

Distributed training ▾

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](#).

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

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

Leveraging gensim for building a FastText model

Text Classification

Text Classification: IMDB Movie Reviews

Download the IMDB dataset

Explore the data

Prepare the data for training

Build the model

Train the model

Evaluate the model

Create a graph of accuracy and loss over time

Text Classification: BBC News Articles

Python Programming

OS, IO, files, and Google Drive

Python Numpy

Python Pandas

Section

+ Code + Text

Text Classification

- Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, <http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/>
- François Chollet (2017), Text classification with preprocessed text: Movie reviews, https://www.tensorflow.org/tutorials/keras/text_classification
- Avishek Nag (2019), Text Classification by XGBoost & Others: A Case Study Using BBC News Articles, <https://medium.com/towards-artificial-intelligence/text-classification-by-xgboost-others-a-case-study-using-bbc-news-articles-5d88e94a9f8>

Text Classification: IMDB Movie Reviews

Source: François Chollet (2017), Text classification with preprocessed text: Movie reviews, https://www.tensorflow.org/tutorials/keras/text_classification

```
[25] 1 !pip install tf-nightly
      2 import tensorflow as tf
      3 print(tf.__version__)
```

```
Collecting tf-nightly
  Downloading https://files.pythonhosted.org/packages/2a/a0/7381cd278a8e1a9235f032ea811af07bbe31ed45ac9781f2/
  517.6MB 24kB/s
Collecting tf-estimator-nightly
  Downloading https://files.pythonhosted.org/packages/0f/fb/984408ab3aee0bddfc02e1136a4fd76c4e58fd128c458e20/
  460kB 40.2MB/s
Requirement already satisfied: google-pasta>=0.1.8 in /usr/local/lib/python3.6/dist-packages (from tf-nightly)
```

<https://tinyurl.com/imtkupython101>

Yelp Dataset Download

4.5GB



Dataset

Dataset

Documentation

Download The Data

The links to download the data will be valid for **30 seconds**.

JSON

Download JSON

4.5 gigabytes compressed
9.8 gigabytes uncompressed

1 .tgz file compressed
1 .pdf file and 5 .json files uncompressed

For more information on the JSON dataset, visit the [main dataset documentation](#) page.

Photos

Download photos

7.0 gigabytes compressed
7.2 gigabytes uncompressed

1 .tar file compressed
1 .json file, 1 text file, 1 .pdf and 1 folder containing 200,000 photos

Evaluation

(Accuracy of Classification Model)

Assessment Methods for Classification

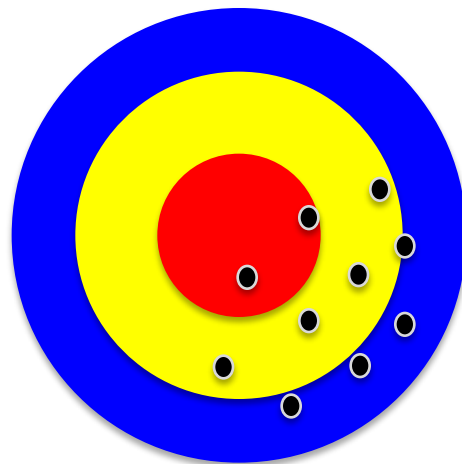
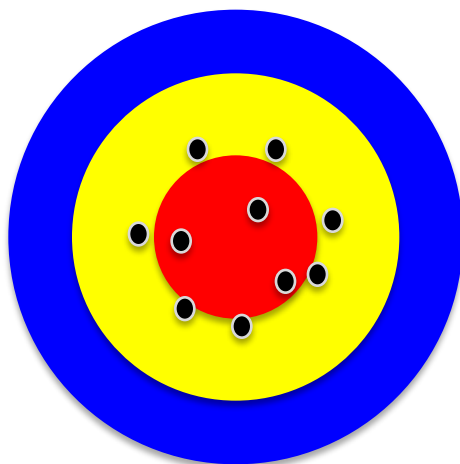
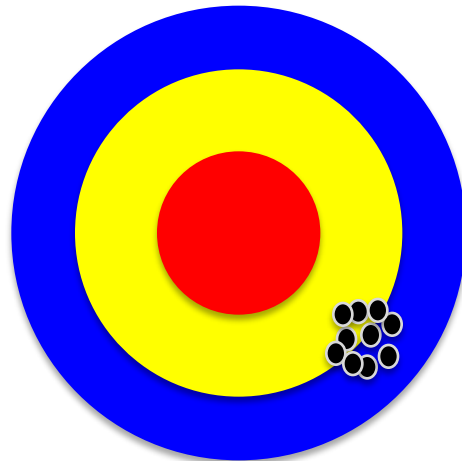
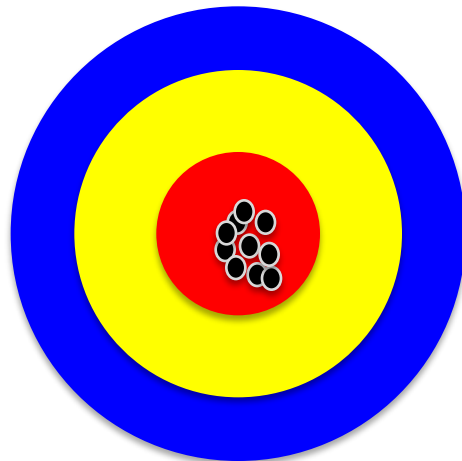
- Predictive accuracy
 - Hit rate
- Speed
 - Model building; predicting
- Robustness
- Scalability
- Interpretability
 - Transparency, explainability

Accuracy

Validity

Precision

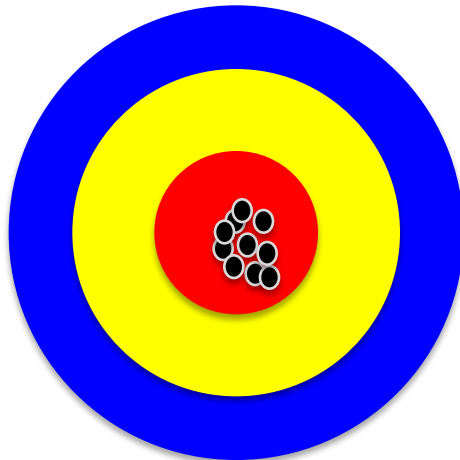
Reliability



Accuracy vs. Precision

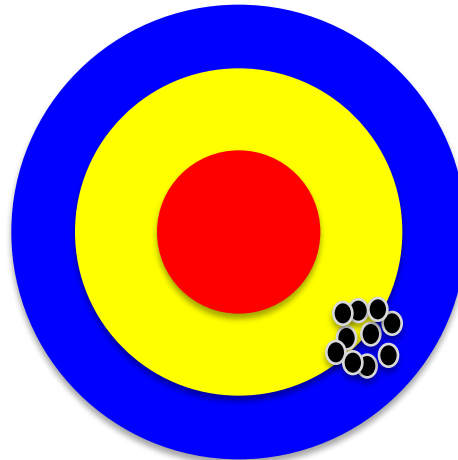
A

**High Accuracy
High Precision**



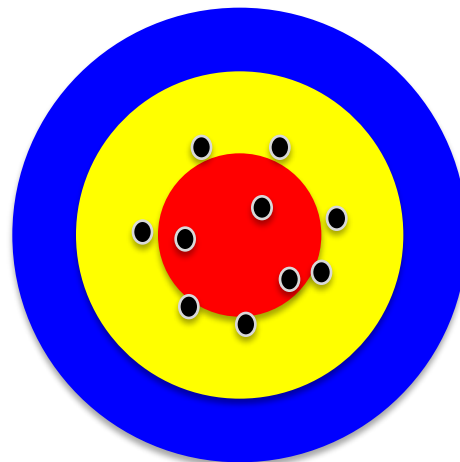
B

**Low Accuracy
High Precision**



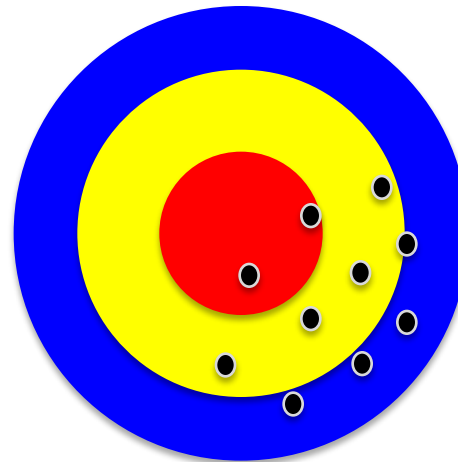
C

**High Accuracy
Low Precision**



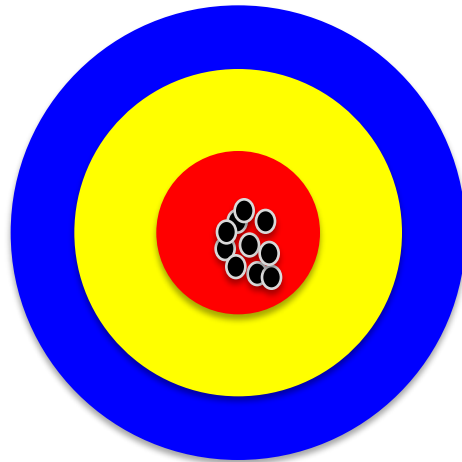
D

**Low Accuracy
Low Precision**



Accuracy vs. Precision

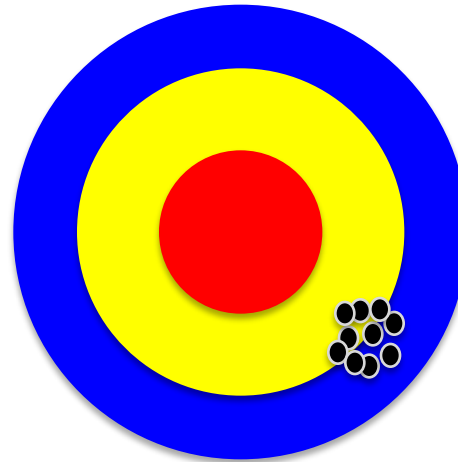
A



**High Accuracy
High Precision**

**High Validity
High Reliability**

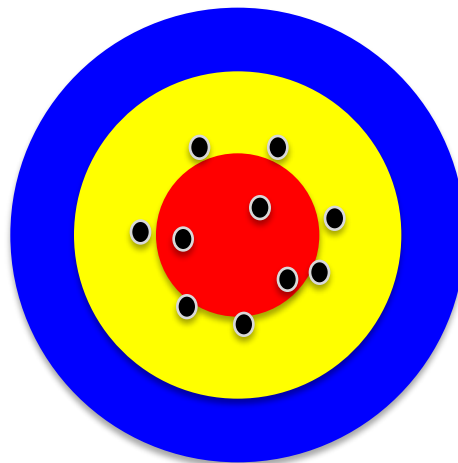
B



**Low Accuracy
High Precision**

**Low Validity
High Reliability**

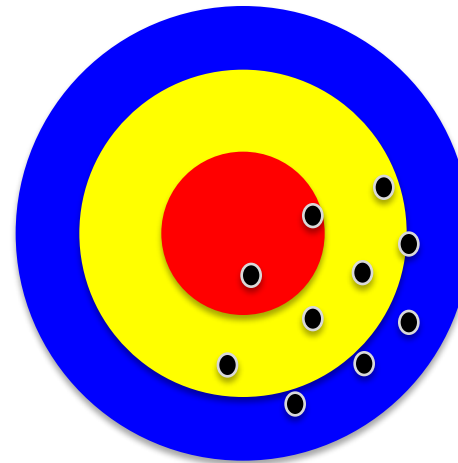
C



**High Accuracy
Low Precision**

**High Validity
Low Reliability**

D

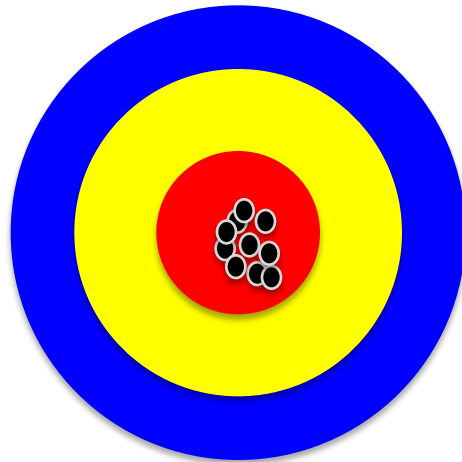


**Low Accuracy
Low Precision**

**Low Validity
Low Reliability**

Accuracy vs. Precision

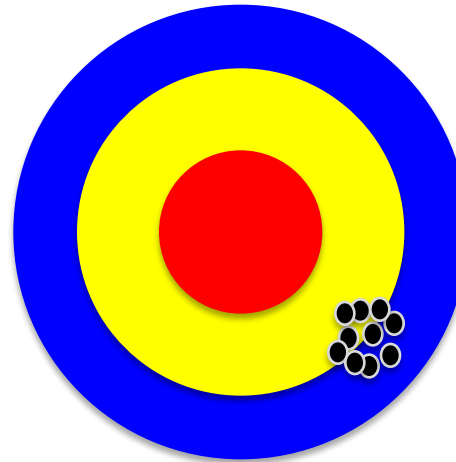
A



High Accuracy
High Precision

High Validity
High Reliability

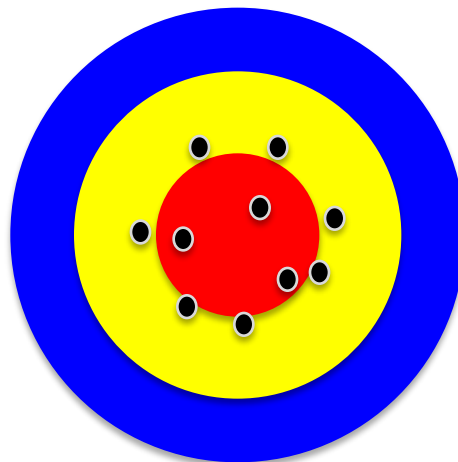
B



Low Accuracy
High Precision

Low Validity
High Reliability

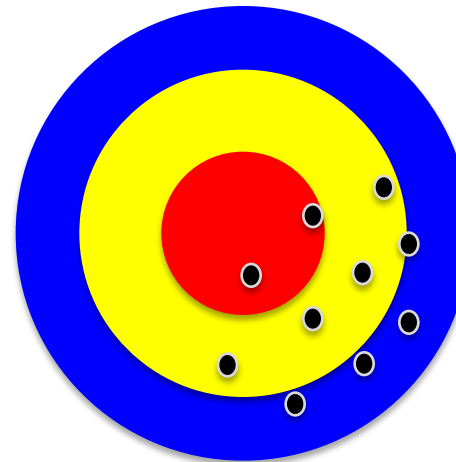
C



High Accuracy
Low Precision

High Validity
Low Reliability

D



Low Accuracy
Low Precision

Low Validity
Low Reliability

Accuracy of Classification Models

- In classification problems, the primary source for accuracy estimation is the **confusion matrix**

		True Class	
		Positive	Negative
Predicted Class	Positive	True Positive Count (TP)	False Positive Count (FP)
	Negative	False Negative Count (FN)	True Negative Count (TN)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$True\ Positive\ Rate = \frac{TP}{TP + FN}$$

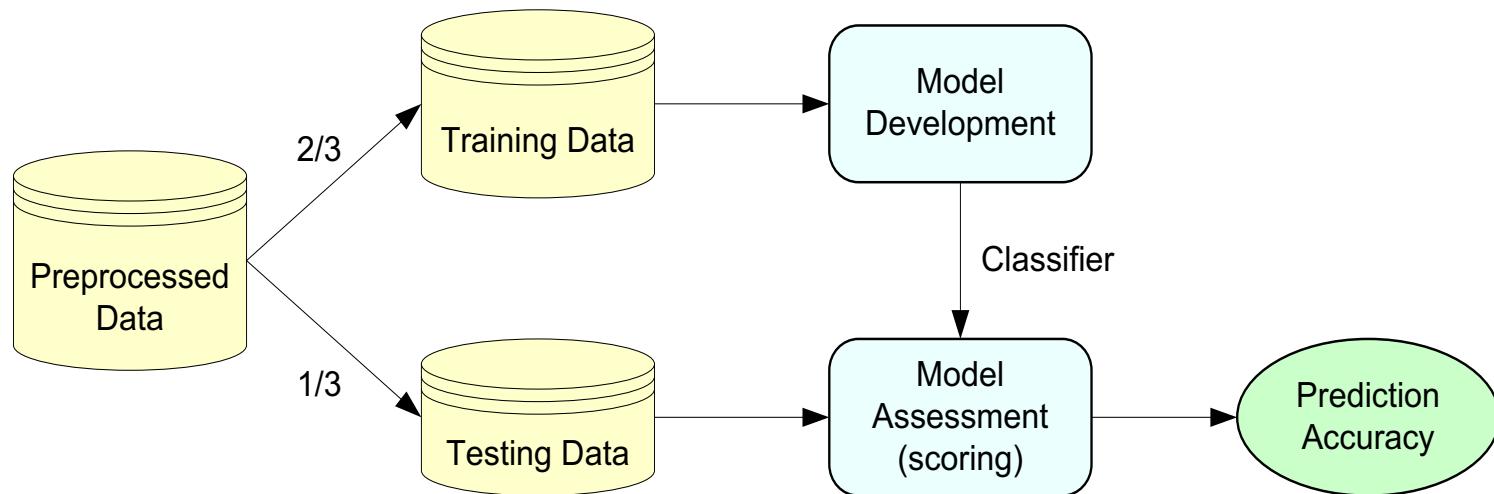
$$True\ Negative\ Rate = \frac{TN}{TN + FP}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Estimation Methodologies for Classification

- **Simple split** (or holdout or test sample estimation)
 - Split the data into 2 mutually exclusive sets training (~70%) and testing (30%)

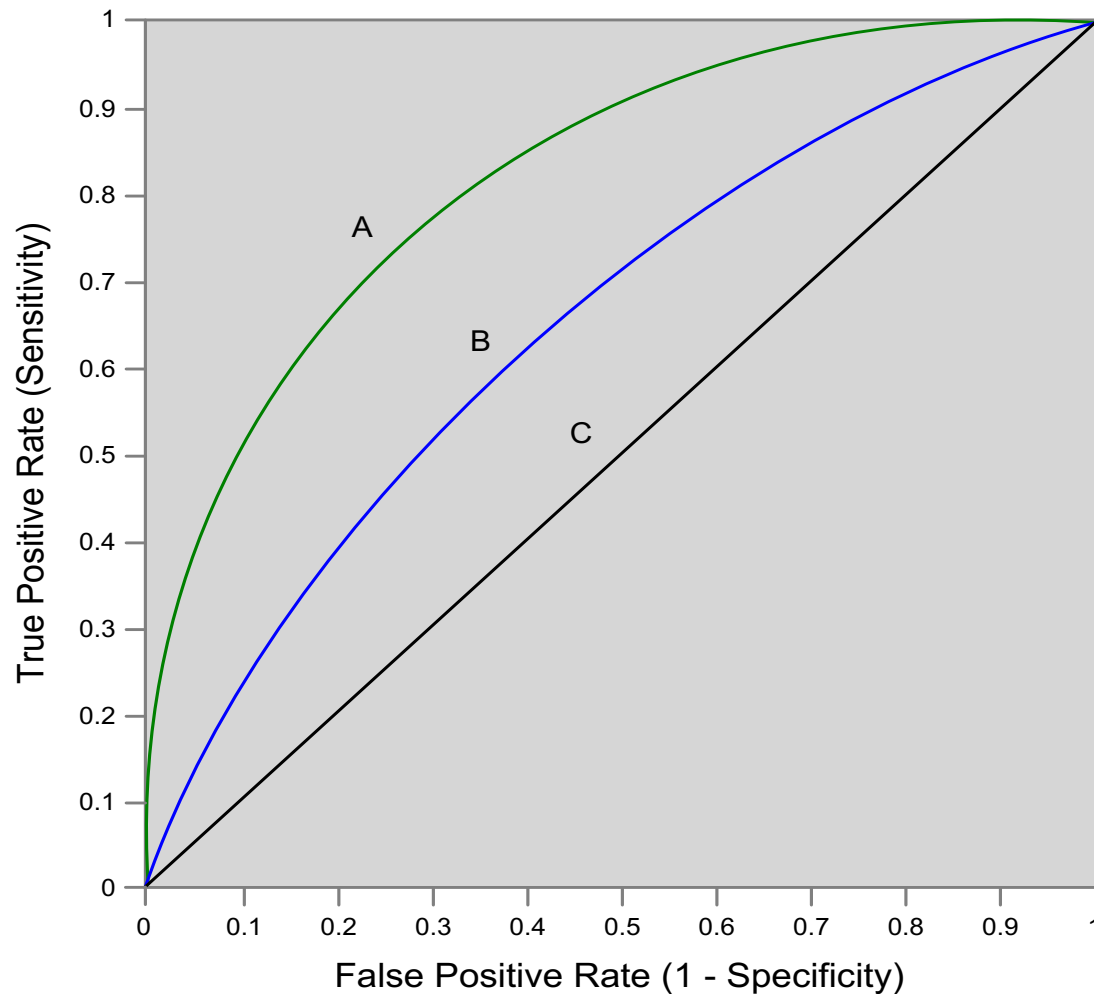


- For ANN, the data is split into three sub-sets (training [~60%], validation [~20%], testing [~20%])

Estimation Methodologies for Classification

- ***k*-Fold Cross Validation** (rotation estimation)
 - Split the data into k mutually exclusive subsets
 - Use each subset as testing while using the rest of the subsets as training
 - Repeat the experimentation for k times
 - Aggregate the test results for true estimation of prediction accuracy training
- Other estimation methodologies
 - ***Leave-one-out*, *bootstrapping*, *jackknifing***
 - ***Area under the ROC curve***

Estimation Methodologies for Classification – ROC Curve



Sensitivity = True Positive Rate

Specificity = True Negative Rate

		True Class (actual value)		total
		Positive	Negative	
Predictive Class (prediction outcome)	Positive	True Positive (TP)	False Positive (FP)	P'
	Negative	False Negative (FN)	True Negative (TN)	N'
total		P	N	

$$\text{True Positive Rate (Sensitivity)} = \frac{TP}{TP + FN}$$

$$\text{True Negative Rate (Specificity)} = \frac{TN}{TN + FP}$$

$$\text{False Positive Rate} = \frac{FP}{FP + TN}$$

$$\text{False Positive Rate (1-Specificity)} = \frac{FP}{FP + TN}$$

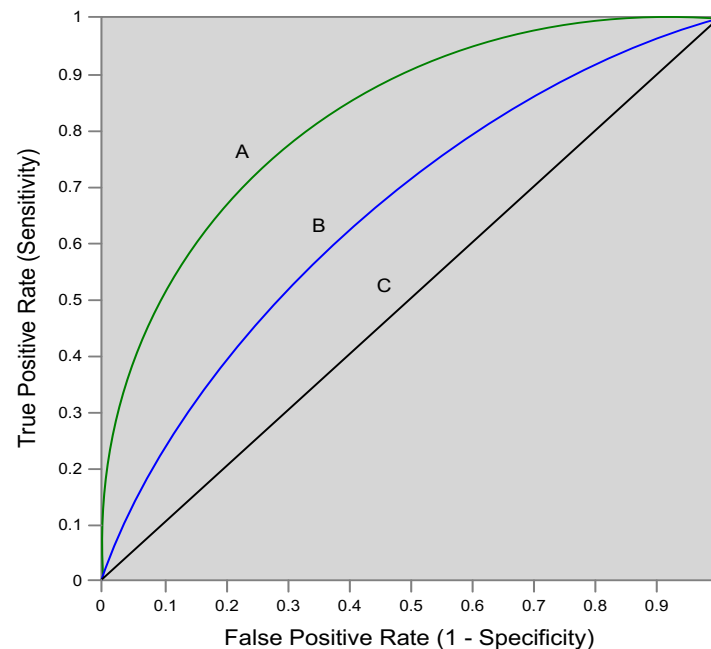
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{True Positive Rate} = \frac{TP}{TP + FN}$$

$$\text{True Negative Rate} = \frac{TN}{TN + FP}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$



		True Class (actual value)		total
		Positive	Negative	
Predictive Class (prediction outcome)	Positive	True Positive (TP)	False Positive (FP)	P'
	Negative	False Negative (FN)	True Negative (TN)	N'
total		P	N	

$$\text{True Positive Rate (Sensitivity)} = \frac{TP}{TP + FN}$$

Sensitivity

= True Positive Rate

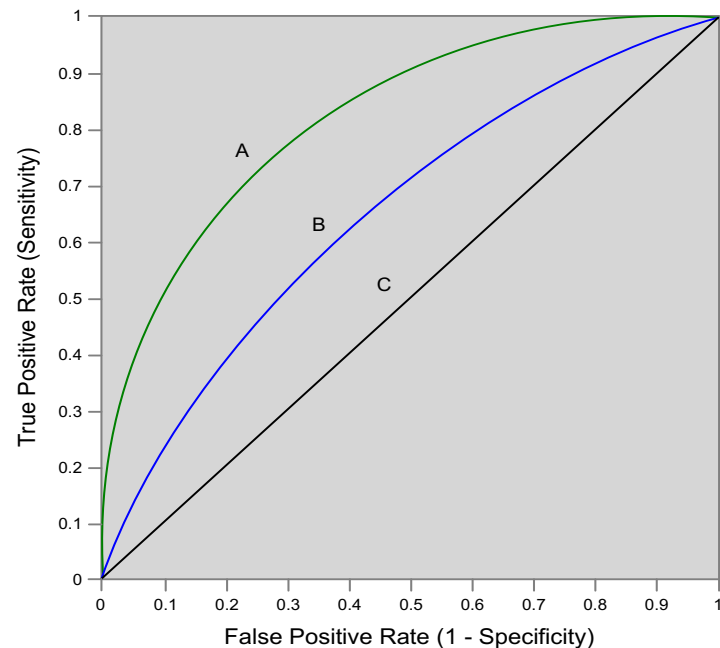
= Recall

= Hit rate

= $TP / (TP + FN)$

$$\text{True Positive Rate} = \frac{TP}{TP + FN}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$



		True Class (actual value)		total
		Positive	Negative	
Predictive Class (prediction outcome)	Positive	True Positive (TP)	False Positive (FP)	P'
	Negative	False Negative (FN)	True Negative (TN)	N'
total		P	N	

Specificity

= True Negative Rate

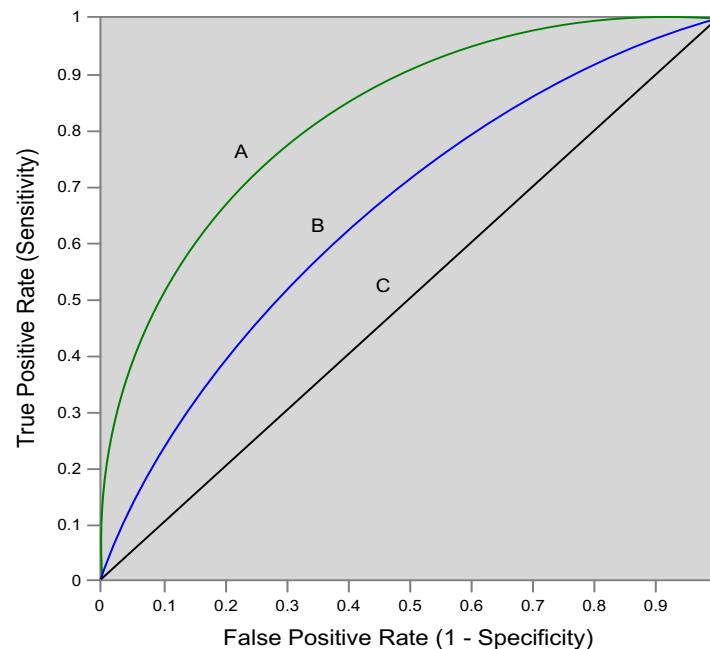
= TN / N

= $TN / (TN + FP)$

$$\text{True Negative Rate (Specificity)} = \frac{TN}{TN + FP}$$

$$\text{False Positive Rate (1 - Specificity)} = \frac{FP}{FP + TN}$$

$$\text{True Negative Rate} = \frac{TN}{TN + FP}$$



		True Class (actual value)		total
		Positive	Negative	
Predictive Class (prediction outcome)	Positive	True Positive (TP)	False Positive (FP)	P'
	Negative	False Negative (FN)	True Negative (TN)	N'
total		P	N	

Precision

= Positive Predictive Value (PPV)

$$Precision = \frac{TP}{TP + FP}$$

Recall

= True Positive Rate (TPR)

= Sensitivity

= Hit Rate

$$Recall = \frac{TP}{TP + FN}$$

F1 score (F-score)(F-measure)

is the harmonic mean of
precision and recall

$$= 2TP / (P + P')$$

$$= 2TP / (2TP + FP + FN)$$

$$F = 2 * \frac{precision * recall}{precision + recall}$$

A

63 (TP)	28 (FP)	91
37 (FN)	72 (TN)	109
100	100	200

Recall

= True Positive Rate (TPR)
 = Sensitivity
 = Hit Rate
 = $TP / (TP + FN)$

Specificity

= True Negative Rate
 = TN / N
 = $TN / (TN + FP)$

$$TPR = 0.63$$

$$Recall = \frac{TP}{TP + FN}$$

$$True\ Negative\ Rate\ (Specificity) = \frac{TN}{TN + FP}$$

$$FPR = 0.28$$

$$False\ Positive\ Rate\ (1 - Specificity) = \frac{FP}{FP + TN}$$

$$PPV = 0.69$$

$$= 63 / (63 + 28)$$

$$= 63 / 91$$

$$Precision = \frac{TP}{TP + FP}$$

Precision

= Positive Predictive Value (PPV)

$$F1 = 0.66$$

$$= 2 * (0.63 * 0.69) / (0.63 + 0.69)$$

$$= (2 * 63) / (100 + 91)$$

$$= (0.63 + 0.69) / 2 = 1.32 / 2 = 0.66$$

$$F = 2 * \frac{precision * recall}{precision + recall}$$

$$ACC = 0.68$$

$$= (63 + 72) / 200$$

$$= 135 / 200 = 67.5$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

F1 score (F-score)
(F-measure)

is the harmonic mean of precision and recall

$$= 2TP / (P + P')$$

$$= 2TP / (2TP + FP + FN)$$

A

63 (TP)	28 (FP)	91
37 (FN)	72 (TN)	109
100	100	200

$$\text{TPR} = 0.63$$

$$\text{FPR} = 0.28$$

$$\text{PPV} = 0.69$$

$$= 63 / (63 + 28)$$

$$= 63 / 91$$

$$\text{F1} = 0.66$$

$$= 2 * (0.63 * 0.69) / (0.63 + 0.69)$$

$$= (2 * 63) / (100 + 91)$$

$$= (0.63 + 0.69) / 2 = 1.32 / 2 = 0.66$$

$$\text{ACC} = 0.68$$

$$= (63 + 72) / 200$$

$$= 135 / 200 = 67.5$$

B

77 (TP)	77 (FP)	154
23 (FN)	23 (TN)	46
100	100	200

$$\text{TPR} = 0.77$$

$$\text{FPR} = 0.77$$

$$\text{PPV} = 0.50$$

$$\text{F1} = 0.61$$

$$\text{ACC} = 0.50$$

Recall

= True Positive Rate (TPR)

= Sensitivity

= Hit Rate

$$\text{Recall} = \frac{TP}{TP + FN}$$

Precision

= Positive Predictive Value (PPV)

$$\text{Precision} = \frac{TP}{TP + FP}$$

C

24 (TP)	88 (FP)	112
76 (FN)	12 (TN)	88
100	100	200

$$\text{TPR} = 0.24$$

$$\text{FPR} = 0.88$$

$$\text{PPV} = 0.21$$

$$\text{F1} = 0.22$$

$$\text{ACC} = 0.18$$

C'

76 (TP)	12 (FP)	88
24 (FN)	88 (TN)	112
100	100	200

$$\text{TPR} = 0.76$$

$$\text{FPR} = 0.12$$

$$\text{PPV} = 0.86$$

$$\text{F1} = 0.81$$

$$\text{ACC} = 0.82$$

Recall

= True Positive Rate (TPR)

= Sensitivity

= Hit Rate

$$\text{Recall} = \frac{TP}{TP + FN}$$

Precision

= Positive Predictive Value (PPV)

$$\text{Precision} = \frac{TP}{TP + FP}$$

Summary

- Text Classification
- Classification Model Evaluation
 - Confusion Matrix
 - Accuracy
 - Precision
 - Recall (TPR) (Sensitivity) (Hit Rate)
 - F1 score (F-measure) (F-score)

References

- 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/>
- Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, <http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/>
- François Chollet (2017), Text classification with preprocessed text: Movie reviews, https://www.tensorflow.org/tutorials/keras/text_classification
- Google Developers (2020), Machine Learning Guides: Text Classification, <https://developers.google.com/machine-learning/guides/text-classification>
- Avishek Nag (2019), Text Classification by XGBoost & Others: A Case Study Using BBC News Articles, <https://medium.com/towards-artificial-intelligence/text-classification-by-xgboost-others-a-case-study-using-bbc-news-articles-5d88e94a9f8>
- Min-Yuh Day (2020), Python 101, <https://tinyurl.com/imtkupython101>