Artificial Intelligence for Text Analytics

Natural Language Processing with Transformers

Min-Yuh Day, Ph.D, Associate Professor

Institute of Information Management, National Taipei University

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

1102AITA04
MBA, IM, NTPU (M5026) (Spring 2022)
Tue 2, 3, 4 (9:10-12:00) (B8F40)

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

2022-03-15
<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Subject/Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2022/02/22</td>
<td>Introduction to Artificial Intelligence for Text Analytics</td>
</tr>
<tr>
<td>2</td>
<td>2022/03/01</td>
<td>Foundations of Text Analytics: Natural Language Processing (NLP)</td>
</tr>
<tr>
<td>3</td>
<td>2022/03/08</td>
<td>Python for Natural Language Processing</td>
</tr>
<tr>
<td>4</td>
<td>2022/03/15</td>
<td>Natural Language Processing with Transformers</td>
</tr>
<tr>
<td>5</td>
<td>2022/03/22</td>
<td>Case Study on Artificial Intelligence for Text Analytics I</td>
</tr>
<tr>
<td>6</td>
<td>2022/03/29</td>
<td>Text Classification and Sentiment Analysis</td>
</tr>
<tr>
<td>Week</td>
<td>Date</td>
<td>Subject/Topics</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>7</td>
<td>2022/04/05</td>
<td>Tomb-Sweeping Day (Holiday, No Classes)</td>
</tr>
<tr>
<td>8</td>
<td>2022/04/12</td>
<td>Midterm Project Report</td>
</tr>
<tr>
<td>9</td>
<td>2022/04/19</td>
<td>Multilingual Named Entity Recognition (NER), Text Similarity and Clustering</td>
</tr>
<tr>
<td>10</td>
<td>2022/04/26</td>
<td>Text Summarization and Topic Models</td>
</tr>
<tr>
<td>11</td>
<td>2022/05/03</td>
<td>Text Generation</td>
</tr>
<tr>
<td>12</td>
<td>2022/05/10</td>
<td>Case Study on Artificial Intelligence for Text Analytics II</td>
</tr>
</tbody>
</table>
## Syllabus

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Subject/Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>2022/05/17</td>
<td>Question Answering and Dialogue Systems</td>
</tr>
<tr>
<td>14</td>
<td>2022/05/24</td>
<td>Deep Learning, Transfer Learning, Zero-Shot, and Few-Shot Learning for Text Analytics</td>
</tr>
<tr>
<td>15</td>
<td>2022/05/31</td>
<td>Final Project Report I</td>
</tr>
<tr>
<td>16</td>
<td>2022/06/07</td>
<td>Final Project Report II</td>
</tr>
<tr>
<td>17</td>
<td>2022/06/14</td>
<td>Self-learning</td>
</tr>
<tr>
<td>18</td>
<td>2022/06/21</td>
<td>Self-learning</td>
</tr>
</tbody>
</table>
Natural Language Processing with Transformers
Outline

• Natural Language Processing with Transformers
  • Transformer (Attention is All You Need)
  • Encoder-Decoder
  • Attention Mechanisms
  • Transfer Learning in NLP
  • BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Source: [https://www.amazon.com/Natural-Language-Processing-Transformers-Applications/dp/1098103246](https://www.amazon.com/Natural-Language-Processing-Transformers-Applications/dp/1098103246)
The Transformers Timeline

2017 2018 2019 2020 2021 2022

Transformer ULMFit BERT RoBERTa XLM-R DeBERTa GPT-Neo GPT-J

GPT GPT-2 DistilBERT GPT-3 T5

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
Transformer (Attention is All You Need)  
(Vaswani et al., 2017)
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT (Bidirectional Encoder Representations from Transformers)

Overall pre-training and fine-tuning procedures for BERT

Pre-training

Fine-Tuning

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 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)

**BERT input representation**

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

Fine-tuning BERT on NLP Tasks

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

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

(c) Question Answering Tasks: SQuAD v1.1

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

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

Sentiment Analysis: Single Sentence Classification

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

Transformer Models

Transformer

Encoder -> Decoder

DistilBERT

BERT

RoBERTa

XLM-R

XLM

ALBERT

ELECTRA

DeBERTa

T5

BART

M2M-100

BigBird

GPT

GPT-2

CTRL

GPT-3

GPT-Neo

GPT-J

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

Source: https://github.com/thunlp/PLMpapers
Transformers Pre-trained Language Model

90+ Models

Pre-trained Models (PTM)

Pre-trained Models (PTM)

<table>
<thead>
<tr>
<th>Knowledge-Enriched</th>
<th>ERNIE(THU) [14], KnowBERT [136], K-BERT [111]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SentiLR [83], KEPLER [195], WKLM [202]</td>
</tr>
<tr>
<td>Multilingual</td>
<td>XLU mBERT [36], Unicoder [68], XLM [27], XLM-R [28], MultiFit [42]</td>
</tr>
<tr>
<td></td>
<td>XLG MASS [160], mBART [118], XNLG [19]</td>
</tr>
<tr>
<td>Language-Specific</td>
<td>ERNIE(Baidu) [170], BERT-wwm-Chinese [29], NEZHA [198], ZEN [37]</td>
</tr>
<tr>
<td></td>
<td>BERTje [33], CamemBERT [125], FlauBERT [95], RobBERT [35]</td>
</tr>
</tbody>
</table>

**Extensions**

- **Image**
  - ViLBERT [120], LXMERT [175], VisualBERT [103], B2T2 [2], VL-BERT [163]

- **Multi-Modal**
  - Video VideoBERT [165], CBT [164]
  - Speech SpeechBERT [22]

- **Domain-Specific**
  - SentiLR [83], BioBERT [98], SciBERT [11], PatentBERT [97]

- **Model Pruning**
  - CompressingBERT [51]

- **Quantization**
  - Q-BERT [156], Q8BERT [211]

- **Parameter Sharing**
  - ALBERT [93]

- **Distillation**
  - DistilBERT [152], TinyBERT [75], MiniLM [194]

- **Module Replacing**
  - BERT-of-Theseus [203]

The Encoder-Decoder Framework

- The encoder-decoder framework
- Attention Mechanisms
- Transfer Learning in NLP
RNN

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
An encoder-decoder architecture with a pair of RNN

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

An encoder-decoder architecture with an attention mechanism

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

alignment of words in English and the generated translation in French

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
Encoder-Decoder Architecture of the Original Transformer

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

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
ULMFiT: 3 Steps
Transfer Learning in NLP

1. Pretraining
2. Domain adaptation
3. Fine-tuning

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O’Reilly Media.
An overview of the Hugging Face Ecosystem

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O’Reilly Media.
A typical pipeline for training transformer models with the Datasets, Tokenizers, and Transformers libraries

Load and process datasets  
Tokenize input texts  
Load models, train and infer  
Load metrics evaluate models

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O’Reilly Media.  
https://github.com/nlp-with-transformers/notebooks
The Illustrated Transformer
Jay Alammar (2018)

The Illustrated Transformer
Jay Alammar (2018)

Source: Jay Alammar (2018), The Illustrated Transformer,
http://jalammar.github.io/illustrated-transformer/
The Illustrated Transformer
Jay Alammar (2018)

The Illustrated Transformer

Jay Alammar (2018)

The Illustrated Transformer
Jay Alammar (2018)

Each word is embedded into a vector of size 512.
The Illustrated Transformer
Jay Alammar (2018)

Encoder

Feed Forward

Self-Attention

x₁  x₂  x₃
Je  suis  étudiant

The Illustrated Transformer
Jay Alammar (2018)

Multiplying $x_1$ by the WQ weight matrix produces $q_1$, the "query" vector associated with that word.

We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.
The Illustrated Transformer
Jay Alammar (2018)

Input

Embedding

x1

q1

k1

v1

q1 \cdot k_1 = 112

x2

q2

k2

v2

q_1 \cdot k_2 = 96

The Illustrated Transformer
Jay Alammar (2018)

Input

Embedding

Queries

Keys

Values

Score

Divide by 8 ($\sqrt{d_k}$)

Softmax

Thinking

Machines

$x_1$

$q_1$

$k_1$

$v_1$

$x_2$

$q_2$

$k_2$

$v_2$

$q_1 \cdot k_1 = 112$

$q_1 \cdot k_2 = 96$

14

12

0.88

0.12


<table>
<thead>
<tr>
<th>Input</th>
<th>Thinking</th>
<th>Machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedding</td>
<td>$x_1$</td>
<td>$x_2$</td>
</tr>
<tr>
<td>Queries</td>
<td>$q_1$</td>
<td>$q_2$</td>
</tr>
<tr>
<td>Keys</td>
<td>$k_1$</td>
<td>$k_2$</td>
</tr>
<tr>
<td>Values</td>
<td>$v_1$</td>
<td>$v_2$</td>
</tr>
<tr>
<td>Score</td>
<td>$q_1 \cdot k_1 = 112$</td>
<td>$q_1 \cdot k_2 = 96$</td>
</tr>
</tbody>
</table>

Divide by $8 (\sqrt{d_k})$

<table>
<thead>
<tr>
<th>14</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.88</td>
<td>0.12</td>
</tr>
</tbody>
</table>

Softmax

Softmax $X$

Value

Sum

| $z_1$ | $z_2$ |
Matrix Calculation of Self-Attention

\[ X \times W^Q = Q \]
\[ X \times W^K = K \]
\[ X \times W^V = V \]

The self-attention calculation in matrix form

\[
\text{softmax}
\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) = Z
\]

Multi-headed Attention

Multi-headed Attention

Calculating attention separately in eight different attention heads

Multi-headed Attention

1) Concatenate all the attention heads

\[ Z_0 \quad Z_1 \quad Z_2 \quad Z_3 \quad Z_4 \quad Z_5 \quad Z_6 \quad Z_7 \]

2) Multiply with a weight matrix \( W^o \) that was trained jointly with the model

\[ Z \]

3) The result would be the \( Z \) matrix that captures information from all the attention heads. We can send this forward to the FFNN

Multi-headed Attention

1) This is our input sentence*
2) We embed each word*
3) Split into 8 heads. We multiply X or R with weight matrices
4) Calculate attention using the resulting Q/K/V matrices
5) Concatenate the resulting Z matrices, then multiply with weight matrix \( W^o \) to produce the output of the layer

* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

As we encode the word "it", one attention head is focusing most on "the animal", while another is focusing on "tired" -- in a sense, the model's representation of the word "it" bakes in some of the representation of both "animal" and "tired".

Add all the attention heads

Layer: 5  
Attention: Input - Input

To give the model a sense of the order of the words, we add positional encoding vectors -- the values of which follow a specific pattern.

Positional encoding with a toy embedding size of 4

Positional Encoding

Positional encoding for 20 words (rows) with an embedding size of 512 (columns)

You can see that it appears split in half down the center. That’s because the values of the left half are generated by one function (which uses sine), and the right half is generated by another function (which uses cosine). They’re then concatenated to form each of the positional encoding vectors.

Transformers Positional Encoding

The Residuals

The Decoder Side

Decoding time step: 1 2 3 4 5 6

The Decoder Side

Decoding time step: 1 2 3 4 5 6

OUTPUT

ENCODERS

EMBEDDING
WITH TIME
SIGNAL

EMBEDDINGS

INPUT

Je  suis  étudiant

PREVIOUS
OUTPUTS

DECODERS

Linear + Softmax

$K_{encdec}$  $V_{encdec}$

The Final Linear and Softmax Layer

Which word in our vocabulary is associated with this index?

Get the index of the cell with the highest value (argmax)

The output vocabulary of our model is created in the preprocessing phase before we even begin training.

<table>
<thead>
<tr>
<th>WORD</th>
<th>a</th>
<th>am</th>
<th>I</th>
<th>thanks</th>
<th>student</th>
<th>&lt;eos&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDEX</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>
Example: one-hot encoding of output vocabulary

<table>
<thead>
<tr>
<th>WORD</th>
<th>a</th>
<th>am</th>
<th>I</th>
<th>thanks</th>
<th>student</th>
<th>&lt;eos&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDEX</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

One-hot encoding of the word “am”

0.0 1.0 0.0 0.0 0.0 0.0 0.0

The Loss Function

Untrained Model Output

Correct and desired output

### Target Model Outputs

<table>
<thead>
<tr>
<th>Output Vocabulary:</th>
<th>a</th>
<th>am</th>
<th>l</th>
<th>thanks</th>
<th>student</th>
<th>&lt;eos&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>position #1</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>position #2</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>position #3</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>position #4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>position #5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Trained Model Outputs

Output Vocabulary: a am I thanks student <eos>

position #1:
- a: 0.01
- am: 0.02
- I: 0.93
- thanks: 0.01
- student: 0.03
- <eos>: 0.01

position #2:
- a: 0.01
- am: 0.8
- I: 0.1
- thanks: 0.05
- student: 0.01
- <eos>: 0.03

position #3:
- a: 0.99
- am: 0.001
- I: 0.001
- thanks: 0.001
- student: 0.002
- <eos>: 0.001

position #4:
- a: 0.001
- am: 0.002
- I: 0.001
- thanks: 0.02
- student: 0.94
- <eos>: 0.01

position #5:
- a: 0.01
- am: 0.01
- I: 0.001
- thanks: 0.001
- student: 0.001
- <eos>: 0.98

Transformers

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

• Transformers
  • pytorch-transformers
  • pytorch-pretrained-bert

• provides state-of-the-art general-purpose architectures
  • (BERT, GPT-2, RoBERTa, XLM, DistilBert, XLNet, CTRL...)

• for Natural Language Understanding (NLU) and Natural Language Generation (NLG) with over 32+ pretrained models in 100+ languages and deep interoperability between TensorFlow 2.0 and PyTorch.

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

State-of-the-art Machine Learning for Jax, Pytorch and TensorFlow

Transformers (formerly known as pytorch-transformers and pytorch-pretrained-bert) provides thousands of pretrained models to perform tasks on different modalities such as text, vision, and audio.

These models can applied on:

- Text, for tasks like text classification, information extraction, question answering, summarization, translation, text generation, in over 100 languages.
- Images, for tasks like image classification, object detection, and segmentation.
- Audio, for tasks like speech recognition and audio classification.

Transformer models can also perform tasks on several modalities combined, such as table question answering, optical character recognition, information extraction from scanned documents, video classification, and visual question answering.
Hugging Face Tasks
Natural Language Processing

- **Text Classification**: 3345 models
- **Token Classification**: 1492 models
- **Question Answering**: 1140 models
- **Translation**: 1467 models
- **Summarization**: 323 models
- **Text Generation**: 3959 models
- **Fill-Mask**: 2453 models
- **Sentence Similarity**: 352 models

[https://huggingface.co/tasks](https://huggingface.co/tasks)
# NLP with Transformers Github

## Jupyter notebooks for the Natural Language Processing with Transformers book

- [Readme](#)
- [Apache-2.0 License](#)
- [1.1k stars](#)
- [33 watching](#)
- [170 forks](#)

## Releases
- No releases published

## Packages
- [transformationbook.com](#)

---

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

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

Running on a cloud platform

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

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Colab</th>
<th>Kaggle</th>
<th>Gradient</th>
<th>Studio Lab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>Open in Colab</td>
<td>Open in Kaggle</td>
<td>Run on Gradient</td>
<td>Open Studio Lab</td>
</tr>
<tr>
<td>Text Classification</td>
<td>Open in Colab</td>
<td>Open in Kaggle</td>
<td>Run on Gradient</td>
<td>Open Studio Lab</td>
</tr>
<tr>
<td>Transformer Anatomy</td>
<td>Open in Colab</td>
<td>Open in Kaggle</td>
<td>Run on Gradient</td>
<td>Open Studio Lab</td>
</tr>
<tr>
<td>Multilingual Named Entity Recognition</td>
<td>Open in Colab</td>
<td>Open in Kaggle</td>
<td>Run on Gradient</td>
<td>Open Studio Lab</td>
</tr>
<tr>
<td>Text Generation</td>
<td>Open in Colab</td>
<td>Open in Kaggle</td>
<td>Run on Gradient</td>
<td>Open Studio Lab</td>
</tr>
<tr>
<td>Summarization</td>
<td>Open in Colab</td>
<td>Open in Kaggle</td>
<td>Run on Gradient</td>
<td>Open Studio Lab</td>
</tr>
<tr>
<td>Question Answering</td>
<td>Open in Colab</td>
<td>Open in Kaggle</td>
<td>Run on Gradient</td>
<td>Open Studio Lab</td>
</tr>
<tr>
<td>Making Transformers Efficient in Production</td>
<td>Open in Colab</td>
<td>Open in Kaggle</td>
<td>Run on Gradient</td>
<td>Open Studio Lab</td>
</tr>
<tr>
<td>Dealing with Few to No Labels</td>
<td>Open in Colab</td>
<td>Open in Kaggle</td>
<td>Run on Gradient</td>
<td>Open Studio Lab</td>
</tr>
<tr>
<td>Training Transformers from Scratch</td>
<td>Open in Colab</td>
<td>Open in Kaggle</td>
<td>Run on Gradient</td>
<td>Open Studio Lab</td>
</tr>
<tr>
<td>Future Directions</td>
<td>Open in Colab</td>
<td>Open in Kaggle</td>
<td>Run on Gradient</td>
<td>Open Studio Lab</td>
</tr>
</tbody>
</table>

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

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

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>label</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEGATIVE</td>
<td>0.901546</td>
</tr>
</tbody>
</table>
Text Classification

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

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

<table>
<thead>
<tr>
<th>label</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NEGATIVE</td>
</tr>
</tbody>
</table>
```

Named Entity Recognition

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

<table>
<thead>
<tr>
<th>entity_group</th>
<th>score</th>
<th>word</th>
<th>start</th>
<th>end</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORG</td>
<td>0.879010</td>
<td>Amazon</td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>MISC</td>
<td>0.990859</td>
<td>Optimus Prime</td>
<td>36</td>
<td>49</td>
</tr>
<tr>
<td>LOC</td>
<td>0.999755</td>
<td>Germany</td>
<td>90</td>
<td>97</td>
</tr>
<tr>
<td>MISC</td>
<td>0.556570</td>
<td>Mega</td>
<td>208</td>
<td>212</td>
</tr>
<tr>
<td>PER</td>
<td>0.590256</td>
<td>##tron</td>
<td>212</td>
<td>216</td>
</tr>
<tr>
<td>ORG</td>
<td>0.669692</td>
<td>Decept</td>
<td>253</td>
<td>259</td>
</tr>
<tr>
<td>MISC</td>
<td>0.498349</td>
<td>##icons</td>
<td>259</td>
<td>264</td>
</tr>
<tr>
<td>MISC</td>
<td>0.775362</td>
<td>Megatron</td>
<td>350</td>
<td>358</td>
</tr>
<tr>
<td>MISC</td>
<td>0.987854</td>
<td>Optimus Prime</td>
<td>367</td>
<td>380</td>
</tr>
<tr>
<td>PER</td>
<td>0.812096</td>
<td>Bumblebee</td>
<td>502</td>
<td>511</td>
</tr>
</tbody>
</table>

Question Answering

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

<table>
<thead>
<tr>
<th>score</th>
<th>start</th>
<th>end</th>
<th>answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.631292</td>
<td>335</td>
<td>358</td>
<td>an exchange of Megatron</td>
</tr>
</tbody>
</table>

Bumblebee ordered an Optimus Prime action figure from your online store in Germany. Unfortunately, when I opened the package, I discovered to my horror that I had been sent an action figure of Megatron instead.
from transformers import set_seed
set_seed(42)  # Set the seed to get reproducible results

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

Customer service response:
Dear Bumblebee, I am sorry to hear that your order was mixed up. The order was completely mislabeled, which is very common in our online store, but I can appreciate it because it was my understanding from this site and our customer service of the previous day that your order was not made correct in our mind and that we are in a process of resolving this matter. We can assure you that your order
Dear Amazon, last week I ordered an Optimus Prime action figure from your online store in Germany. Unfortunately, when I opened the package, I discovered to my horror that I had been sent an action figure of Megatron instead! As a lifelong enemy of the Decepticons, I hope you can understand my dilemma. To resolve the issue, I demand an exchange of Megatron for the Optimus Prime figure I ordered. Enclosed are copies of my records concerning this purchase. I expect to hear from you soon. Sincerely, Bumblebee.

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

Python in Google Colab (Python101)

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

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

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

```python
1 from utils import *
2 setup_chapter()
```

```python
1 text = """Dear Amazon, last week I ordered an Optimus Prime action figure \n2 from your online store in Germany. Unfortunately, when I opened the package, \n3 I discovered to my horror that I had been sent an action figure of Megatron \n4 instead! As a lifelong enemy of the Decepticons, I hope you can understand my \n5 dilemma. To resolve the issue, I demand an exchange of Megatron for the \n6 Optimus Prime figure I ordered. Enclosed are copies of my records concerning \n7 this purchase. I expect to hear from you soon. Sincerely, Rumblebee.""
```

Text Classification

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

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

https://tinyurl.com/aintpuppython101
Summary

• Natural Language Processing with Transformers
  • Transformer (Attention is All You Need)
  • Encoder-Decoder
  • Attention Mechanisms
  • Transfer Learning in NLP
  • BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
References

- Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Denis Rothman (2021), Transformers for Natural Language Processing: Build innovative deep neural network architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and more, Packt Publishing.
- Savaş Yıldırım and Meysam Asgari-Chenaghlu (2021), Mastering Transformers: Build state-of-the-art models from scratch with advanced natural language processing techniques, Packt Publishing.
- Charu C. Aggarwal (2018), Machine Learning for Text, Springer.
- Gabe Ignatow and Rada F. Mihalcea (2017), An Introduction to Text Mining: Research Design, Data Collection, and Analysis, SAGE Publications.
- Rajesh Arumugam (2018), Hands-On Natural Language Processing with Python: A practical guide to applying deep learning architectures to your NLP applications, Packt.