Text Classification and Sentiment Analysis

Min-Yuh Day, Ph.D,
Associate Professor

Institute of Information Management, National Taipei University

https://web.ntpu.edu.tw/~myday
<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>
</tbody>
</table>
## Syllabus

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Subject/Topics</th>
</tr>
</thead>
<tbody>
<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>
Text Classification and Sentiment Analysis
Outline

• Text Classification and Sentiment Analysis
  • Dataset
  • Tokenizer
  • Training a Text Classifier
  • Fine-Tuning Transformers
Text Classification (TC) Tasks

- Sentiment Analysis
- News Categorization
- Product Categorization
- Topic Analysis
  - Topic Classification: “customer support” or “ease of use”
- Natural language inference (NLI)
  - recognizing textual entailment (RTE)
  - entailment, contradiction, and neutral

Deep learning models for text embedding and classification

# Text Classification Models on Sentiment Analysis

<table>
<thead>
<tr>
<th>Method</th>
<th>IMDB</th>
<th>SST-2</th>
<th>Amazon-2</th>
<th>Amazon-5</th>
<th>Yelp-2</th>
<th>Yelp-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive Bayes [43]</td>
<td>-</td>
<td>81.80</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LDA [214]</td>
<td>67.40</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BoW+SVM [31]</td>
<td>87.80</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>if:Δ idf [215]</td>
<td>88.10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Char-level CNN [50]</td>
<td>-</td>
<td>-</td>
<td>94.49</td>
<td>59.46</td>
<td>95.12</td>
<td>62.05</td>
</tr>
<tr>
<td>Deep Pyramid CNN [49]</td>
<td>-</td>
<td>84.46</td>
<td>96.68</td>
<td>65.82</td>
<td>97.36</td>
<td>69.40</td>
</tr>
<tr>
<td>ULMFiT [216]</td>
<td>95.40</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>97.84</td>
</tr>
<tr>
<td>BLSTM-2DCNN [40]</td>
<td>-</td>
<td>89.50</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Neural Semantic Encoder [95]</td>
<td>-</td>
<td>89.70</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BCN+Char+CoVe [217]</td>
<td>91.80</td>
<td>90.30</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GLUE ELMo baseline [22]</td>
<td>-</td>
<td>90.40</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BERT ELMo baseline [7]</td>
<td>-</td>
<td>90.40</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CCCCapsNet [76]</td>
<td>-</td>
<td>-</td>
<td>94.96</td>
<td>60.95</td>
<td>96.48</td>
<td>65.85</td>
</tr>
<tr>
<td>Virtual adversarial training [173]</td>
<td>94.10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Block-sparse LSTM [218]</td>
<td>94.99</td>
<td>93.20</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BERT-base [7, 154]</td>
<td>95.63</td>
<td>93.50</td>
<td>96.04</td>
<td>61.60</td>
<td>98.08</td>
<td>70.58</td>
</tr>
<tr>
<td>BERT-large [7, 154]</td>
<td>95.79</td>
<td>94.9</td>
<td>96.07</td>
<td>62.20</td>
<td>98.19</td>
<td>71.38</td>
</tr>
<tr>
<td>ALBERT [147]</td>
<td>-</td>
<td>95.20</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Multi-Task DNN [23]</td>
<td>83.20</td>
<td>95.60</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Snorkel MeTaL [219]</td>
<td>-</td>
<td>96.20</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BERT Finetune + UDA [220]</td>
<td>95.80</td>
<td>96.50</td>
<td>62.88</td>
<td>97.95</td>
<td>62.92</td>
<td></td>
</tr>
<tr>
<td>RoBERTa (+additional data) [146]</td>
<td>-</td>
<td>96.40</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XLNet-Large (ensemble) [156]</td>
<td>96.21</td>
<td>96.80</td>
<td>97.60</td>
<td>67.74</td>
<td>98.45</td>
<td>72.20</td>
</tr>
</tbody>
</table>

Classification Models on News Categorization, and Topic Classification

<table>
<thead>
<tr>
<th>Method</th>
<th>News Categorization</th>
<th>Topic Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AG News</td>
<td>20NEWS</td>
</tr>
<tr>
<td>Hierarchical</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Log-bilinear Model [221]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Text GCN [107]</td>
<td>67.61</td>
<td>86.34</td>
</tr>
<tr>
<td>Simplified GCN [108]</td>
<td>-</td>
<td>88.50</td>
</tr>
<tr>
<td>Char-level CNN [50]</td>
<td>90.49</td>
<td>-</td>
</tr>
<tr>
<td>CCCapsNet [76]</td>
<td>92.39</td>
<td>-</td>
</tr>
<tr>
<td>LEAM [84]</td>
<td>92.45</td>
<td>81.91</td>
</tr>
<tr>
<td>fastText [30]</td>
<td>92.50</td>
<td>-</td>
</tr>
<tr>
<td>CapsuleNet B [71]</td>
<td>92.60</td>
<td>-</td>
</tr>
<tr>
<td>Deep Pyramid CNN [49]</td>
<td>93.13</td>
<td>-</td>
</tr>
<tr>
<td>ULMFiT [216]</td>
<td>94.99</td>
<td>-</td>
</tr>
<tr>
<td>L MIXED [174]</td>
<td>95.05</td>
<td>-</td>
</tr>
<tr>
<td>BERT-large [220]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XLNet [156]</td>
<td>95.51</td>
<td>-</td>
</tr>
</tbody>
</table>

### Classification Models on Natural Language Inference (NLI)

<table>
<thead>
<tr>
<th>Method</th>
<th>SNLI</th>
<th>MultiNLI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unigrams Features [208]</td>
<td>71.6</td>
<td>—</td>
</tr>
<tr>
<td>Lexicalized [208]</td>
<td>78.2</td>
<td>—</td>
</tr>
<tr>
<td>LSTM encoders (100D) [208]</td>
<td>77.6</td>
<td>—</td>
</tr>
<tr>
<td>Tree-based CNN [61]</td>
<td>82.1</td>
<td>—</td>
</tr>
<tr>
<td>biLSTM Encoder [209]</td>
<td>81.5</td>
<td>67.5</td>
</tr>
<tr>
<td>Neural Semantic Encoders (300D) [95]</td>
<td>84.6</td>
<td>—</td>
</tr>
<tr>
<td>RNN-based Sentence Encoder [224]</td>
<td>85.5</td>
<td>73.2</td>
</tr>
<tr>
<td>DiSAN (300D) [81]</td>
<td>85.6</td>
<td>—</td>
</tr>
<tr>
<td>Decomposable Attention Model [92]</td>
<td>86.3</td>
<td>—</td>
</tr>
<tr>
<td>Reinforced Self-Attention (300D) [177]</td>
<td>86.3</td>
<td>—</td>
</tr>
<tr>
<td>Generalized Pooling (600D) [93]</td>
<td>86.6</td>
<td>73.8</td>
</tr>
<tr>
<td>Bilateral multi-perspective matching [41]</td>
<td>87.5</td>
<td>—</td>
</tr>
<tr>
<td>Multiway Attention Network [87]</td>
<td>88.3</td>
<td>78.5</td>
</tr>
<tr>
<td>ESIM + ELMo [4]</td>
<td>88.7</td>
<td>72.9</td>
</tr>
<tr>
<td>DMAN with Reinforcement Learning [225]</td>
<td>88.8</td>
<td>88.8</td>
</tr>
<tr>
<td>BiLSTM + ELMo + Attn [22]</td>
<td>—</td>
<td>74.1</td>
</tr>
<tr>
<td>Fine-Tuned LM-Pretrained Transformer [6]</td>
<td>89.9</td>
<td>82.1</td>
</tr>
<tr>
<td>Multi-Task DNN [23]</td>
<td>91.6</td>
<td>86.7</td>
</tr>
<tr>
<td>SemBERT [155]</td>
<td>91.9</td>
<td>84.4</td>
</tr>
<tr>
<td>RoBERTa [146]</td>
<td>92.6</td>
<td>90.8</td>
</tr>
<tr>
<td>XLNet [156]</td>
<td>—</td>
<td>90.2</td>
</tr>
</tbody>
</table>

General Language Understanding Evaluation (GLUE) benchmark

GLUE Test results

<table>
<thead>
<tr>
<th>System</th>
<th>MNLI-(m/mm)</th>
<th>QQP</th>
<th>QNLI</th>
<th>SST-2</th>
<th>CoLA</th>
<th>STS-B</th>
<th>MRPC</th>
<th>RTE</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-OpenAI SOTA</td>
<td>80.6/80.1</td>
<td>66.1</td>
<td>82.3</td>
<td>93.2</td>
<td>35.0</td>
<td>81.0</td>
<td>86.0</td>
<td>61.7</td>
<td>74.0</td>
</tr>
<tr>
<td>BiLSTM+ELMo+Attn</td>
<td>76.4/76.1</td>
<td>64.8</td>
<td>79.9</td>
<td>90.4</td>
<td>36.0</td>
<td>73.3</td>
<td>84.9</td>
<td>56.8</td>
<td>71.0</td>
</tr>
<tr>
<td>OpenAI GPT</td>
<td>82.1/81.4</td>
<td>70.3</td>
<td>88.1</td>
<td>91.3</td>
<td>45.4</td>
<td>80.0</td>
<td>82.3</td>
<td>56.0</td>
<td>75.2</td>
</tr>
<tr>
<td>BERTBASE</td>
<td>84.6/83.4</td>
<td>71.2</td>
<td>90.1</td>
<td>93.5</td>
<td>52.1</td>
<td>85.8</td>
<td>88.9</td>
<td>66.4</td>
<td>79.6</td>
</tr>
<tr>
<td>BERTLARGE</td>
<td>86.7/85.9</td>
<td>72.1</td>
<td>91.1</td>
<td>94.9</td>
<td>60.5</td>
<td>86.5</td>
<td>89.3</td>
<td>70.1</td>
<td>81.9</td>
</tr>
</tbody>
</table>

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
Emotions

- Love
- Joy
- Surprise
- Anger
- Sadness
- Fear

Example of Opinion: review segment on iPhone

“I bought an iPhone a few days ago.
It was such a nice phone.
The touch screen was really cool.
The voice quality was clear too.
However, my mother was mad with me as I did not tell her before I bought it.
She also thought the phone was too expensive, and wanted me to return it to the shop. ... ”

“(1) I bought an iPhone a few days ago.

(2) It was such a nice phone.

(3) The touch screen was really cool.

(4) The voice quality was clear too.

(5) However, my mother was mad with me as I did not tell her before I bought it.

(6) She also thought the phone was too expensive, and wanted me to return it to the shop. ... ”

Sentiment Analysis

Tasks

- Subjectivity Classification
- Sentiment Classification
- Review Usefulness Measurement
- Opinion Spam Detection
- Lexicon Creation
- Aspect Extraction
- Application

Approaches

- Machine Learning based
- Lexicon based
- Hybrid approaches
- Ontology based
- Non-Ontology based

Sentiment Classification Techniques

P–N Polarity and S–O Polarity Relationship

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

Hugging Face Hub

Models  Datasets  Metrics  Docs

Tokenizers  Transformers  Datasets

Accelerate

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

NLP with Transformers

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

from utils import *
setup_chapter()
```
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>NEGATIVE</td>
<td>0.901546</td>
</tr>
</tbody>
</table>
Fine-tuning BERT on NLP Tasks

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
Character Tokenization

text = "Tokenizing text is a core task of NLP."
tokenized_text = list(text)
print(tokenized_text)

['T', 'o', 'k', 'e', 'n', 'i', 'z', 'i', 'n', 'g', ' ', 't', 'e', 'x', 't', ' ', 'i', 's', ' ', 'a', ' ', 'c', 'o', 'r', 'e', ' ', 't', 'a', 's', 'k', ' ', 'o', 'f', ' ', 'N', 'L', 'P', '.']
token2idx = {ch: idx for idx, ch in enumerate(sorted(set(tokenized_text)))}
print(token2idx)

{' ': 0, '.': 1, 'L': 2, 'N': 3, 'P': 4, 'T': 5, 'a': 6, 'c': 7, 'e': 8, 'f': 9, 'g': 10, 'i': 11, 'k': 12, 'n': 13, 'o': 14, 'r': 15, 's': 16, 't': 17, 'x': 18, 'z': 19}

input_ids = [token2idx[token] for token in tokenized_text]
print(input_ids)

[5, 14, 12, 8, 13, 11, 19, 11, 13, 10, 0, 17, 8, 18, 17, 0, 11, 16, 0, 6, 0, 7, 14, 15, 8, 0, 17, 6, 16, 12, 0, 14, 9, 0, 3, 2, 4, 1]
Word Tokenization

text = "Tokenizing text is a core task of NLP."
tokenized_text = text.split()
print(tokenized_text)

['Tokenizing', 'text', 'is', 'a', 'core', 'task', 'of', 'NLP.']
Subword Tokenization

```python
from transformers import AutoTokenizer
model_ckpt = "distilbert-base-uncased"
tokenizer = AutoTokenizer.from_pretrained(model_ckpt)

text = "Tokenizing text is a core task of NLP."
encoded_text = tokenizer(text)
print(encoded_text)

{'input_ids': [101, 19204, 6026, 3793, 2003, 1037, 4563, 4708, 1997, 17953, 2361, 1012, 102], 'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}
tokens = tokenizer.convert_ids_to_tokens(encoded_text.input_ids)
print(tokens)

['[CLS]', 'token', '##izing', 'text', 'is', 'a', 'core', 'task', 'of', 'nl', '##p', '.', '[SEP]']
```

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
Subword Tokenization

print(tokenizer.convert_tokens_to_string(tokens))

[CLS] tokenizing text is a core task of nlp. [SEP]

tokenizer.vocab_size

30522

tokenizer.model_max_length

512
Tokenizing the Whole Dataset

```python
def tokenize(batch):
    return tokenizer(batch["text"], padding=True, truncation=True)

print(tokenize(emotions["train"][::2]))

{'input_ids': [[101, 1045, 2134, 2102, 2514, 26608, 102, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]], 'attention_mask': [[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]}
tokens2ids = list(zip(tokenizer.all_special_tokens, tokenizer.all_special_ids))
data = sorted(tokens2ids, key=lambda x : x[-1])
df = pd.DataFrame(data, columns=["Special Token", "Special Token ID"])
df.T
```
For each batch, the input sequences are padded to the maximum sequence length in the batch; the attention mask is used in the model to ignore the padded areas of the input tensors.

<table>
<thead>
<tr>
<th>Text 1</th>
<th>Input IDs</th>
<th>Attention masks</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>23 74 2 67 102</td>
<td>1 1 1 1 1 1 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>Text 2</td>
<td>101 14 66 53 7 87 14 37 31 17 9 21 102</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>Text 3</td>
<td>101 91 20 15 98 36 81 85 23 102</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 0 0 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input batch</th>
<th>Stack vectors to input matrices</th>
</tr>
</thead>
<tbody>
<tr>
<td>101 23 74 2 67 102</td>
<td>1 1 1 1 1 1 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>101 14 66 53 7 87 14 37 31 17 9 21 102</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>101 91 20 15 98 36 81 85 23 102</td>
<td>1 1 1 1 1 1 1 1 1 1 1 0 0 0</td>
</tr>
</tbody>
</table>


https://github.com/nlp-with-transformers/notebooks
Training a Text Classifier

The architecture used for sequence classification with an encoder-based transformer; it consists of the model’s pretrained body combined with a custom classification head.

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
Transformers as Feature Extractors

In the feature-based approach, the DistilBERT model is frozen and just provides features for a classifier.

https://github.com/nlp-with-transformers/notebooks
Training a Simple Classifier


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

When using the fine-tuning approach the whole DistilBERT model is trained along with the classification head.

Fine-Tuning Transformers
Loading a pretrained model

```python
from transformers import AutoModelForSequenceClassification

num_labels = 6
model = (AutoModelForSequenceClassification
           .from_pretrained(model_ckpt, num_labels=num_labels)
           .to(device))
```
from sklearn.metrics import accuracy_score, f1_score

def compute_metrics(pred):
    labels = pred.label_ids
    preds = pred.predictions.argmax(-1)
    f1 = f1_score(labels, preds, average="weighted")
    acc = accuracy_score(labels, preds)
    return {"accuracy": acc, "f1": f1}
Train the model

```python
from huggingface_hub import notebook_login

notebook_login()
```
Train the model

```python
from transformers import Trainer, TrainingArguments

batch_size = 64
logging_steps = len(emotions_encoded["train"])//batch_size
model_name = f"{model_ckpt}-finetuned-emotion"
training_args = TrainingArguments(output_dir=model_name,
                                   num_train_epochs=2,
                                   learning_rate=2e-5,
                                   per_device_train_batch_size=batch_size,
                                   per_device_eval_batch_size=batch_size,
                                   weight_decay=0.01,
                                   evaluation_strategy="epoch",
                                   disable_tqdm=False,
                                   logging_steps=logging_steps,
                                   push_to_hub=True,
                                   log_level="error")
```
Train the model

```python
from transformers import Trainer

trainer = Trainer(model=model, args=training_args,
                  compute_metrics=compute_metrics,
                  train_dataset=emotions_encoded['train'],
                  eval_dataset=emotions_encoded['validation'],
                  tokenizer=tokenizer)

trainer.train();
```
Train the model

def test_main():
    preds_output = trainer.predict(emotions_encoded["validation"])
    preds_output.metrics

    y_preds = np.argmax(preds_output.predictions, axis=1)
    plot_confusion_matrix(y_preds, y_valid, labels)
A Visual Guide to Using BERT for the First Time
(Jay Alammar, 2019)

### Sentiment Classification: SST2

**Sentences from movie reviews**

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

Movie Review Sentiment Classifier

“a visually stunning rumination on love” → Movie Review Sentiment Classifier → positive

Movie Review Sentiment Classifier

“a visually stunning rumination on love”

Movie Review Sentiment Classifier
Model Training

Source: Jay Alammar (2019), A Visual Guide to Using BERT for the First Time,
http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/
Step # 1 Use distilBERT to Generate Sentence Embeddings

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

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

Tokenization

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

Tokenization

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

Tokenization for BERT Model

Flowing Through DistilBERT (768 features)

Model #1 Output Class vector as Model #2 Input

Fine-tuning BERT on Single Sentence Classification Tasks

Model #1 Output Class vector as Model #2 Input

Logistic Regression Model to classify Class vector

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

df.head()

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a stirring, funny and finally transporting re...</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>apparently reassembled from the cutting room f...</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>they presume their audience won't sit still f...</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>this is a visually stunning rumination on love...</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>jonathan parker's bartleby should have been t...</td>
<td>1</td>
</tr>
</tbody>
</table>
Tokenization

tokenized = df[0].apply((lambda x: tokenizer.encode(x, add_special_tokens=True)))
### BERT/DistilBERT Input Tensor

<table>
<thead>
<tr>
<th>Input sequences (reviews)</th>
<th>Tokens in each sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>101</td>
</tr>
<tr>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1,999</td>
<td>101</td>
</tr>
</tbody>
</table>

Processing with DistilBERT

\[
\text{input_ids} = \text{torch.tensor(np.array(padded))}
\]
\[
\text{last_hidden_states} = \text{model(input_ids)}
\]
Unpacking the BERT output tensor

last_hidden_states[0]

BERT Output Tensor/predictions

66 Tokens in each sequence

2,000 Output rows (one per sequence)

768 Number of hidden units

Sentence to last_hidden_state[0]

BERT’s output for the [CLS] tokens

# Slice the output for the first position for all the sequences, take all hidden unit outputs
features = last_hidden_states[0][:,0,:].numpy()
The tensor sliced from BERT's output

Sentence Embeddings

Dataset for Logistic Regression
(768 Features)

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

<table>
<thead>
<tr>
<th>features</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>1,999</td>
<td>1</td>
</tr>
</tbody>
</table>

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

```python
# 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**

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

A Visual Notebook to Using BERT for the First Time

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

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

Running on a cloud platform

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

<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
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
  2  
  3!cd notebooks
  4!from install import *
  5!install_requirements()

[3] 1!from util import *
  2!setup_chapter()

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

Text Classification

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

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

https://tinyurl.com/aintpuppython101
Python in Google Colab (Python101)

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

Text Classification with Transformers

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

```python
[10] !nvidia-smi

# Uncomment and run this cell if you're on Colab or Kaggle
2 !git clone https://github.com/nlp-with-transformers/notebooks.git
3 cd notebooks
4 from install import *
5 install_requirements()

[12] # hide
2 from utils import *
3 setup_chapter()
```

The Dataset

```python
[13] 1 from datasets import list_datasets
2 all_datasets = list_datasets()
3 print(f"There are {len(all_datasets)} datasets currently available on the Hub")
4 print(f"The first 10 are: {all_datasets[:10]}")
```

There are 3783 datasets currently available on the Hub
The first 10 are: ['acronym_identification', 'ade_corpus_v2', 'adversarial_qa',

https://tinyurl.com/aintpupython101
Summary

• Text Classification and Sentiment Analysis
  • Dataset
  • Tokenizer
  • Training a Text Classifier
  • Fine-Tuning Transformers
• 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.


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


• 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.


• The Super Duper NLP Repo, https://notebooks.quantumstat.com/


• NLP with Transformer, https://github.com/nlp-with-transformers/notebooks

• Min-Yuh Day (2022), Python 101, https://tinyurl.com/aintpupython101