人工智能文本分析
(AI for Text Analytics)

文本分類
(Text Classification)

Min-Yuh Day
戴敏育
Associate Professor
副教授

Institute of Information Management, National Taipei University

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

2020-11-05
<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Subject/Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2020/09/17</td>
<td>Course Orientation on Artificial Intelligence for Text Analytics</td>
</tr>
<tr>
<td>2</td>
<td>2020/09/24</td>
<td>Foundations of Text Analytics: Natural Language Processing; NLP</td>
</tr>
<tr>
<td>3</td>
<td>2020/10/01</td>
<td>Mid-Autumn Festival</td>
</tr>
<tr>
<td>4</td>
<td>2020/10/08</td>
<td>Python for Natural Language Processing</td>
</tr>
<tr>
<td>5</td>
<td>2020/10/15</td>
<td>Processing and Understanding Text</td>
</tr>
<tr>
<td>6</td>
<td>2020/10/22</td>
<td>Feature Engineering for Text Representation</td>
</tr>
</tbody>
</table>
# 課程大綱 (Syllabus)

週次 (Week) 日期 (Date) 內容 (Subject/Topics)

7 2020/10/29 人工智慧文本分析個案研究 I  
(Case Study on Artificial Intelligence for Text Analytics I)

8 2020/11/05 文本分類  
(Text Classification)

9 2020/11/12 文本摘要和主題模型  
(Text Summarization and Topic Models)

10 2020/11/19 期中報告 (Midterm Project Report)

11 2020/11/26 文本相似度和分群  
(Text Similarity and Clustering)

12 2020/12/03 語意分析和命名實體識別  
(Semantic Analysis and Named Entity Recognition; NER)
<table>
<thead>
<tr>
<th>週次 (Week)</th>
<th>日期 (Date)</th>
<th>內容 (Subject/Topics)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>2020/12/10</td>
<td>情感分析 (Sentiment Analysis)</td>
</tr>
<tr>
<td>14</td>
<td>2020/12/17</td>
<td>人工智慧文本分析個案研究 II (Case Study on Artificial Intelligence for Text Analytics II)</td>
</tr>
<tr>
<td>15</td>
<td>2020/12/24</td>
<td>深度學習和通用句子嵌入模型 (Deep Learning and Universal Sentence-Embedding Models)</td>
</tr>
<tr>
<td>16</td>
<td>2020/12/31</td>
<td>問答系統與對話系統 (Question Answering and Dialogue Systems)</td>
</tr>
<tr>
<td>17</td>
<td>2021/01/07</td>
<td>期末報告 I (Final Project Presentation I)</td>
</tr>
<tr>
<td>18</td>
<td>2021/01/14</td>
<td>期末報告 II (Final Project Presentation II)</td>
</tr>
</tbody>
</table>
Outline

• Traditional Feature Engineering for Text Data
  • Bag of Words Model
  • Bag of N-Grams Model
  • TF-IDF Model
• Advanced Word Embeddings with Deep Learning
  • Word2Vec Model
  • Robust Word2Vec Models with Gensim
  • GloVe Model
  • FastText Model
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

Documents → Language Detection → Pre-processing → Modeling → Output

English → Tokenization (English) → POS Tagging (English) → Stopword Removal (EN) → ... → Sentiment

Spanish → Tokenization (Spanish) → POS Tagging (Spanish) → Stopword Removal (ES) → ... → Classification

Arabic → Tokenization (Arabic) → POS Tagging (Arabic) → Stopword Removal (AR) → ... → Entity Extraction

Deep Learning-based NLP

Documents → Preprocessing → Dense Embeddings → Hidden Layers → Output Units → Output

Dense Embeddings obtained via word2vec, doc2vec, GloVe, etc.

Output Units

Sentiment → Classification → Entity Extraction → Translation → Topic Modelling → ...

Source: http://blog.aylien.com/leveraging-deep-learning-for-multilingual/
Modern NLP Pipeline

[Diagram showing a modern NLP pipeline with stages such as Language Detection, Tokenize, POS Tagging, Token Filtering, Build Vocabulary, Bag-of-Words & Vectorization, and Machine Learning, leading to output tasks like Classification, Sentiment Analysis, Entity Extraction, Topic Modeling, and Similarity.]
Modern NLP Pipeline
Deep Learning NLP

Documents → Preprocessing → Dense Word Embeddings → Deep Neural Network

Pre-generated Lookup
OR
Generated in 1st level of NeuralNet

Task / Output
- Classification
- Sentiment Analysis
- Entity Extraction
- Topic Modeling
- Document Similarity

Source: http://mattfortier.me/2017/01/31/nlp-intro-pt-1-overview/
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

Start

Token mode

word

Vectorization mode

Yes

N-gram

N-gram range

unigram

bigram

trigram

Count mode

binary

tf-idf

No

S/W < 1500

sequence

S - Number of samples
W - Number of words per sample

Source: Google Developers (2020), Machine Learning Guides: Text Classification,
https://developers.google.com/machine-learning/guides/text-classification
Text Classification S/W<1500: N-gram

Prepare data

- N-gram
  - N-gram range
    - unigram
    - bigram
    - trigram
  - Count mode
    - binary
    - tf-idf
    - count
  - Scoring method
    - none
    - f_classif
    - chi2
  - Select top_k features [score]
    - min(top: 1K, 2K, ..., 15K, 20K, 25K, ..., 90K, all)
  - Normalization mode
    - samplewise
    - None
    - featurewise
  - Build model
    - SVM
    - MLP
    - GBDT

Text Classification S/W\geq1500: Sequence

Select top_k features [freq]

min(top_{1K, 2K,... 15K}, 20K, 25K,... 90K, all)

Normalization mode

samplewise

None

featurewise

Embeddings

S/W < 15K

Yes

Fine-tuned pre-trained embedding

No

Frozen pre-trained embedding

Embeddings learned from scratch

Build model

RNN

stacked RNN

CNN-RNN

sepCNN

CNN

Hyperparameter tuning

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]  
T1: 'The mouse ran down' =  
   [1, 2, 3, 6]
One-hot encoding

'The mouse ran up the clock' =

<table>
<thead>
<tr>
<th>Word</th>
<th>Index</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>The</td>
<td>1</td>
<td>[0, 1, 0, 0, 0, 0, 0, 0]</td>
</tr>
<tr>
<td>mouse</td>
<td>2</td>
<td>[0, 0, 1, 0, 0, 0, 0, 0]</td>
</tr>
<tr>
<td>ran</td>
<td>3</td>
<td>[0, 0, 0, 1, 0, 0, 0, 0]</td>
</tr>
<tr>
<td>up</td>
<td>4</td>
<td>[0, 0, 0, 0, 1, 0, 0, 0]</td>
</tr>
<tr>
<td>the</td>
<td>1</td>
<td>[0, 1, 0, 0, 0, 0, 0, 0]</td>
</tr>
<tr>
<td>clock</td>
<td>5</td>
<td>[0, 0, 0, 0, 0, 0, 1, 0]</td>
</tr>
</tbody>
</table>

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

Male-Female
Verb Tense
Country-Capital

Source: https://developers.google.com/machine-learning/guides/text-classification/step-3
Word embeddings

The mouse ran up the clock

The mouse ran down

Source: https://developers.google.com/machine-learning/guides/text-classification/step-3
Text Classification

Top K Features (20K) vs Accuracy

Text Classification

Linear stack of layers

Text Classification

Last layer

Output layer
Binary classification

Output layer
Multi-class classification

Sequence to Sequence (Seq2Seq)

Encoder

Knowledge → is → power → <end>

Decoder

d₀ → d₁ → d₂ → d₃

Attention

Source: https://google.github.io/seq2seq/
Transformer (Attention is All You Need) (Vaswani et al., 2017)

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

Jacob Devlin  Ming-Wei Chang  Kenton Lee  Kristina Toutanova
Google AI Language
{jacobdevlin, mingweichang, kentonl1, 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.

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

<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<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>BERT_{BASE}</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>BERT_{LARGE}</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

Transfer Learning in Natural Language Processing

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**

<table>
<thead>
<tr>
<th>Input</th>
<th>[CLS]</th>
<th>my</th>
<th>dog</th>
<th>is</th>
<th>cute</th>
<th>[SEP]</th>
<th>he</th>
<th>likes</th>
<th>play</th>
<th># # ing</th>
<th>[SEP]</th>
</tr>
</thead>
</table>

**Token Embeddings**

<table>
<thead>
<tr>
<th>Input</th>
<th>[CLS]</th>
<th>my</th>
<th>dog</th>
<th>is</th>
<th>cute</th>
<th>[SEP]</th>
<th>he</th>
<th>likes</th>
<th>play</th>
<th># # ing</th>
<th>[SEP]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Token Embeddings</td>
<td>E_{CLS}</td>
<td>E_{my}</td>
<td>E_{dog}</td>
<td>E_{is}</td>
<td>E_{cute}</td>
<td>E_{[SEP]}</td>
<td>E_{he}</td>
<td>E_{likes}</td>
<td>E_{play}</td>
<td>E_{# # ing}</td>
<td>E_{[SEP]}</td>
</tr>
</tbody>
</table>

**Segment Embeddings**

<table>
<thead>
<tr>
<th>Input</th>
<th>[CLS]</th>
<th>my</th>
<th>dog</th>
<th>is</th>
<th>cute</th>
<th>[SEP]</th>
<th>he</th>
<th>likes</th>
<th>play</th>
<th># # ing</th>
<th>[SEP]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segment Embeddings</td>
<td>E_{A}</td>
<td>E_{A}</td>
<td>E_{A}</td>
<td>E_{A}</td>
<td>E_{A}</td>
<td>E_{A}</td>
<td>E_{B}</td>
<td>E_{B}</td>
<td>E_{B}</td>
<td>E_{B}</td>
<td>E_{B}</td>
</tr>
</tbody>
</table>

**Position Embeddings**

<table>
<thead>
<tr>
<th>Input</th>
<th>[CLS]</th>
<th>my</th>
<th>dog</th>
<th>is</th>
<th>cute</th>
<th>[SEP]</th>
<th>he</th>
<th>likes</th>
<th>play</th>
<th># # ing</th>
<th>[SEP]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position Embeddings</td>
<td>E_{0}</td>
<td>E_{1}</td>
<td>E_{2}</td>
<td>E_{3}</td>
<td>E_{4}</td>
<td>E_{5}</td>
<td>E_{6}</td>
<td>E_{7}</td>
<td>E_{8}</td>
<td>E_{9}</td>
<td>E_{10}</td>
</tr>
</tbody>
</table>

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

Pre-trained Language Model (PLM)

Semi-supervised Sequence Learning
context2Vec
Pre-trained seq2seq

ULMFiT
ELMo
Transformer

Bidirectional LM
Larger model
More data

Multi-lingual
Cross-lingual
Multi-task

MultiFiT
Mass
UniLM

Span prediction
Remove NSP
Longer time
Remove NSP
More data

MT-DNN
XLM UDify
Knowledge distillation

MT-DNN KD
SpanBERT
RoBERTa

Span prediction
Remove NSP
More data

XLNet

ERNE (Tsinghua)
Neural entity linker

KnowBert

VideoBERT
CBT
ViLBERT
VisualBERT
B2T2
Unicoder-VL
LXMERT
VL-BERT
UNITER

Grover

Defense
Whole Word Masking

By Xiaozhi Wang & Zhengyan Zhang @THUNLP

Source: https://github.com/thunlp/PLMpapers
Turing Natural Language Generation (T-NLG)

- MegatronLM: 8.3b
- GPT-2: 1.5b
- RoBERTa: 355m
- DistilBERT: 66m
- BERT-Large: 340m

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
# NLP Benchmark Datasets

<table>
<thead>
<tr>
<th>Task</th>
<th>Dataset</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WMT 2014 EN-FR</td>
<td></td>
</tr>
<tr>
<td><strong>Text Summarization</strong></td>
<td>CNN/DM</td>
<td><a href="https://cs.nyu.edu/~kcho/DMQA/">https://cs.nyu.edu/~kcho/DMQA/</a></td>
</tr>
<tr>
<td></td>
<td>Newsroom</td>
<td><a href="https://wwwsummari.es/">https://wwwsummari.es/</a></td>
</tr>
<tr>
<td></td>
<td>Gigaword</td>
<td><a href="https://catalog.ldc.upenn.edu/LDC2012T21">https://catalog.ldc.upenn.edu/LDC2012T21</a></td>
</tr>
<tr>
<td><strong>Reading Comprehension</strong></td>
<td>ARC</td>
<td><a href="http://data.allenai.org/arc/">http://data.allenai.org/arc/</a></td>
</tr>
<tr>
<td></td>
<td>CNN/DM</td>
<td><a href="https://cs.nyu.edu/~kcho/DMQA/">https://cs.nyu.edu/~kcho/DMQA/</a></td>
</tr>
<tr>
<td></td>
<td>NewsQA</td>
<td><a href="https://datasets.maluuba.com/NewsQA">https://datasets.maluuba.com/NewsQA</a></td>
</tr>
<tr>
<td></td>
<td>SQuAD</td>
<td><a href="https://rajpurkar.github.io/SQuAD-explorer/">https://rajpurkar.github.io/SQuAD-explorer/</a></td>
</tr>
<tr>
<td>Question Answering</td>
<td>NarrativeQA</td>
<td><a href="https://github.com/deepmind/narrativeqa">https://github.com/deepmind/narrativeqa</a></td>
</tr>
<tr>
<td>Question Generation</td>
<td>Quasar</td>
<td><a href="https://github.com/nyu-dl/SearchQA">https://github.com/nyu-dl/SearchQA</a></td>
</tr>
<tr>
<td></td>
<td>SearchQA</td>
<td></td>
</tr>
<tr>
<td><strong>Semantic Parsing</strong></td>
<td>AMR parsing</td>
<td><a href="https://amr.isi.edu/index.html">https://amr.isi.edu/index.html</a></td>
</tr>
<tr>
<td></td>
<td>ATIS (SQL Parsing)</td>
<td><a href="https://github.com/jkkummerfeld/text2sql-data/tree/master/data">https://github.com/jkkummerfeld/text2sql-data/tree/master/data</a></td>
</tr>
<tr>
<td></td>
<td>WikiSQL (SQL Parsing)</td>
<td><a href="https://github.com/salesforce/WikiSQL">https://github.com/salesforce/WikiSQL</a></td>
</tr>
<tr>
<td><strong>Sentiment Analysis</strong></td>
<td>IMDB Reviews</td>
<td><a href="http://ai.stanford.edu/~amaas/data/sentiment/">http://ai.stanford.edu/~amaas/data/sentiment/</a></td>
</tr>
<tr>
<td></td>
<td>SST</td>
<td><a href="https://nlp.stanford.edu/sentiment/index.html">https://nlp.stanford.edu/sentiment/index.html</a></td>
</tr>
<tr>
<td></td>
<td>Yelp Reviews</td>
<td><a href="https://www.yelp.com/dataset/challenge">https://www.yelp.com/dataset/challenge</a></td>
</tr>
<tr>
<td></td>
<td>DBpedia</td>
<td><a href="https://wiki.dbpedia.org/Datasets">https://wiki.dbpedia.org/Datasets</a></td>
</tr>
<tr>
<td></td>
<td>TREC</td>
<td><a href="https://trec.nist.gov/data.html">https://trec.nist.gov/data.html</a></td>
</tr>
<tr>
<td><strong>Natural Language Inference</strong></td>
<td>SNLI Corpus</td>
<td><a href="https://nlp.stanford.edu/projects/snli/">https://nlp.stanford.edu/projects/snli/</a></td>
</tr>
<tr>
<td></td>
<td>MultiNLI</td>
<td><a href="https://www.nyu.edu/projects/bowman/multinli/">https://www.nyu.edu/projects/bowman/multinli/</a></td>
</tr>
<tr>
<td><strong>Semantic Role Labeling</strong></td>
<td>Proposition Bank</td>
<td><a href="http://propbank.github.io/">http://propbank.github.io/</a></td>
</tr>
<tr>
<td></td>
<td>OneNotes</td>
<td><a href="https://catalog.ldc.upenn.edu/LDC2013T19">https://catalog.ldc.upenn.edu/LDC2013T19</a></td>
</tr>
</tbody>
</table>

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

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/
<table>
<thead>
<tr>
<th>sentence</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>a stirring, funny and finally transporting re imagining 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

Movie Review Sentiment Classifier

DistilBERT

Already (pre-)trained

Logistic Regression

We will train in this tutorial

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 #1: Use DistilBERT to embed all the sentences

<table>
<thead>
<tr>
<th>Sentence</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>a stirring, funny and finally transporting re imaging of beauty and the beast and 1930s</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>the movie is undone by a filmmaking methodology that 's just experimental enough</td>
<td>1</td>
</tr>
</tbody>
</table>

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

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


<table>
<thead>
<tr>
<th>Sentence Embeddings</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>-0.215</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>1,499</td>
<td></td>
</tr>
</tbody>
</table>

**Model Training**

**Logistic Regression**

![Diagram of logistic regression model training process]
Tokenization

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

a visually stunning rumination on love

Tokenization

```python
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

Model #2 Output

1 (positive)

Model #2 Input

Model #1 Output

Logistic Regression Model to classify Class vector

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

<table>
<thead>
<tr>
<th></th>
<th>text</th>
<th>sentiment</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 Input Tensor

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

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

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/
## Sentence to last_hidden_state[0]

---

**input_ids**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>101</td>
<td>1037</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,999</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**last_hidden_states[0]**

---

**Batch**

Tokenize all 2,000 sentences
Put each sentence in its own row

```
[CLS] a visually stunning run imagination on love [SEP] ... 0
```

---

**Tokenize**

```
101 1037 1745 14726 19379 12758 2006 2291 102 ... 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)
Score Benchmarks
Logistic Regression Model
on SST-2 Dataset

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

# Testing
```python
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 re imagining 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

“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

Text classification with preprocessed text: Movie reviews

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.

https://www.tensorflow.org/tutorials/keras/text_classification
Python in Google Colab (Python101)

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

Text Classification

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

Text Classification: IMDB Movie Reviews

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

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

https://tinyurl.com/aintpupython101
Yelp Dataset Download
4.5GB

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

<table>
<thead>
<tr>
<th>JSON</th>
<th>Photos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download JSON</td>
<td>Download photos</td>
</tr>
<tr>
<td>4.5 gigabytes compressed</td>
<td>7.0 gigabytes compressed</td>
</tr>
<tr>
<td>9.8 gigabytes uncompressed</td>
<td>7.2 gigabytes uncompressed</td>
</tr>
<tr>
<td>1 .tgz file compressed</td>
<td>1 .tar file compressed</td>
</tr>
<tr>
<td>1 .pdf file and 5 .json files uncompressed</td>
<td>1 .json file, 1 text file, 1 .pdf and 1 folder containing 200,000 photos</td>
</tr>
</tbody>
</table>

For more information on the JSON dataset, visit the main dataset documentation page.

Source: https://www.yelp.com/dataset/download
Evaluation

(Accuracy of Classification Model)
Assessment Methods for Classification

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

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems
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**

<table>
<thead>
<tr>
<th>True Class</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Class</td>
<td>True Positive Count (TP)</td>
<td>False Positive Count (FP)</td>
</tr>
</tbody>
</table>

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

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

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

\[
\text{Precision} = \frac{TP}{TP + FP} \quad \text{Recall} = \frac{TP}{TP + FN}
\]

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems
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%])

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems
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

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems
Estimation Methodologies for Classification – ROC Curve

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems
Sensitivity = True Positive Rate

Specificity = True Negative Rate
### True Class (Actual Value)

<table>
<thead>
<tr>
<th>Predictive Class (Prediction Outcome)</th>
<th>True Class (Actual Value)</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>Positive (TP)</td>
<td>P</td>
</tr>
<tr>
<td>False Positive</td>
<td>False Positive (FP)</td>
<td>P'</td>
</tr>
<tr>
<td>Negative</td>
<td>False Negative (FN)</td>
<td>N'</td>
</tr>
<tr>
<td>True Negative</td>
<td>True Negative (TN)</td>
<td>N</td>
</tr>
</tbody>
</table>

### Calculations

- **Accuracy**
  \[
  \text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}
  \]

- **True Positive Rate**
  \[
  \text{True Positive Rate} = \frac{TP}{TP + FN}
  \]

- **True Negative Rate**
  \[
  \text{True Negative Rate} = \frac{TN}{TN + FP}
  \]

- **Precision**
  \[
  \text{Precision} = \frac{TP}{TP + FP}
  \]

- **Recall**
  \[
  \text{Recall} = \frac{TP}{TP + FN}
  \]

### Source

See [Receiver operating characteristic](http://en.wikipedia.org/wiki/Receiver_operating_characteristic) for more details.

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

Sensitivity
= True Positive Rate
= Recall
= Hit rate
= \frac{TP}{(TP + FN)}

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

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

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

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

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

Precision
= Positive Predictive Value (PPV)
\[
\text{Precision} = \frac{TP}{TP + FP}
\]

Recall
= True Positive Rate (TPR)
= Sensitivity
= Hit Rate
\[
\text{Recall} = \frac{TP}{TP + FN}
\]

F1 score (F-score)(F-measure)
is the harmonic mean of precision and recall
= \(2TP / (P + P')\)
= \(2TP / (2TP + FP + FN)\)
\[
F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}
\]

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

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

Precision = Positive Predictive Value (PPV)
= TP / (TP + FP)

F1 score (F-score) (F-measure)
is the harmonic mean of precision and recall
= 2TP / (P + P')
= 2TP / (2TP + FP + FN)

TPR = 0.63
FPR = 0.28
PPV = 0.69
F1 = 0.66
Accuracy = 0.68

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

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

\text{Specificity} = \frac{TN}{TN + FP}

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

\text{F1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}

\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}
<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP</td>
<td>63 (91)</td>
<td>77 (154)</td>
</tr>
<tr>
<td>FN</td>
<td>37 (109)</td>
<td>23 (46)</td>
</tr>
<tr>
<td>FP</td>
<td>28 (100)</td>
<td>77 (100)</td>
</tr>
<tr>
<td>TN</td>
<td>72 (200)</td>
<td>23 (200)</td>
</tr>
<tr>
<td>TPR</td>
<td>0.63</td>
<td>0.77</td>
</tr>
<tr>
<td>FPR</td>
<td>0.28</td>
<td>0.77</td>
</tr>
<tr>
<td>PPV</td>
<td>0.69</td>
<td>0.50</td>
</tr>
<tr>
<td>F1</td>
<td>0.66</td>
<td>0.61</td>
</tr>
<tr>
<td>ACC</td>
<td>0.68</td>
<td>0.50</td>
</tr>
</tbody>
</table>

**Recall**

\[
Recall = \frac{TP}{TP + FN}
\]

**Precision**

\[
Precision = \frac{TP}{TP + FP}
\]
### Confusion Matrices

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>C'</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TP</strong></td>
<td>24</td>
<td>76</td>
</tr>
<tr>
<td><strong>FP</strong></td>
<td>88</td>
<td>12</td>
</tr>
<tr>
<td><strong>FN</strong></td>
<td>76</td>
<td>12</td>
</tr>
<tr>
<td><strong>TN</strong></td>
<td>100</td>
<td>88</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>200</td>
<td>200</td>
</tr>
</tbody>
</table>

**Metrics**

- **TPR (True Positive Rate)**: \( \frac{TP}{TP + FN} \)  
- **FPR (False Positive Rate)**: \( \frac{FP}{FP + TN} \)  
- **PPV (Positive Predictive Value)**: \( \frac{TP}{TP + FP} \)  
- **F1 Score**:
  \[ \frac{2 	imes TP}{2 	imes TP + FP + FN} \]  
- **Accuracy (ACC)**: \( \frac{TP + TN}{TP + TN + FP + FN} \)

For matrix C:
- TPR = 0.24
- FPR = 0.88
- PPV = 0.21
- F1 = 0.22
- ACC = 0.18

For matrix C':
- TPR = 0.76
- FPR = 0.12
- PPV = 0.86
- F1 = 0.81
- ACC = 0.82

### Recall

Recall = True Positive Rate (TPR)  
Recall = Sensitivity  
Recall = Hit Rate

### Precision

Precision = Positive Predictive Value (PPV)  
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


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

