文字探勘
(Text Mining)

文本分類
(Text Classification)

1082TM06
MBA, BDABI, TKU (E3611) (8480) (Spring 2020)
Mon, 7, 8, 9 (14:10-17:00) (B206)

Chichang Jou
周清江
Associate Professor
副教授
cjou@mail.tku.edu.tw

Min-Yuh Day
戴敏育
Associate Professor
副教授
myday@mail.tku.edu.tw

Dept. of Information Management, Tamkang University
淡江大學 資訊管理學系

2020-04-13
<table>
<thead>
<tr>
<th>週次 (Week)</th>
<th>日期 (Date)</th>
<th>內容 (Subject/Topics)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2020/03/02</td>
<td>2020/03/02</td>
<td>文字探勘課程介紹 (Course Orientation on Text Mining)</td>
</tr>
<tr>
<td>2 2020/03/09</td>
<td>2020/03/09</td>
<td>文字探勘基礎：自然語言處理 (Foundations of Text Mining: Natural Language Processing; NLP)</td>
</tr>
<tr>
<td>3 2020/03/16</td>
<td>2020/03/16</td>
<td>Python自然語言處理 (Python for Natural Language Processing)</td>
</tr>
<tr>
<td>4 2020/03/23</td>
<td>2020/03/23</td>
<td>處理和理解文本 (Processing and Understanding Text)</td>
</tr>
<tr>
<td>5 2020/03/30</td>
<td>2020/03/30</td>
<td>文本表達特徵工程 (Feature Engineering for Text Representation)</td>
</tr>
<tr>
<td>6 2020/04/06</td>
<td>2020/04/06</td>
<td>人工智慧文本分析個案研究 I (Case Study on Artificial Intelligence for Text Analytics I)</td>
</tr>
</tbody>
</table>
課程大綱 (Syllabus)

週次 (Week) 日期 (Date)  內容 (Subject/Topics)
7 2020/04/13 文本分類  
（Text Classification）
8 2020/04/20 文本摘要和主題模型  
（Text Summarization and Topic Models）
9 2020/04/27 期中報告 (Midterm Project Report)
10 2020/05/04 文本相似度和分群  
（Text Similarity and Clustering）
11 2020/05/11 語意分析和命名實體識別  
（Semantic Analysis and Named Entity Recognition; NER）
12 2020/05/18 情感分析  
（Sentiment Analysis）
<table>
<thead>
<tr>
<th>週次 (Week)</th>
<th>日期 (Date)</th>
<th>內容 (Subject/Topics)</th>
</tr>
</thead>
</table>
| 13        | 2020/05/25 | 人工智慧文本分析個案研究 II  
(Case Study on Artificial Intelligence for Text Analytics II) |
| 14        | 2020/06/01 | 深度學習和通用句子嵌入模型  
(Deep Learning and Universal Sentence-Embedding Models) |
| 15        | 2020/06/08 | 問答系統與對話系統  
(Question Answering and Dialogue Systems) |
| 16        | 2020/06/15 | 期末報告 I (Final Project Presentation I) |
| 17        | 2020/06/22 | 期末報告 II (Final Project Presentation II) |
| 18        | 2020/06/29 | 教師彈性補充教學 |
Outline

• Text Classification

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

Classical NLP

Deep Learning-based NLP

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

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

Documents → Preprocessing → Dense Word Embeddings → Deep Neural Network

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

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

char

Vectorization mode

S/W < 1500

Yes

N-gram

N-gram range

unigram

bigram

trigram

Count mode

binary

tf-idf

count

No

sequence

Text Classification S/W<1500: N-gram

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]

Source: https://developers.google.com/machine-learning/guides/text-classification/step-3
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, 0]</td>
</tr>
<tr>
<td>mouse</td>
<td>2</td>
<td>[0, 0, 1, 0, 0, 0, 0, 0, 0]</td>
</tr>
<tr>
<td>ran</td>
<td>3</td>
<td>[0, 0, 0, 1, 0, 0, 0, 0, 0]</td>
</tr>
<tr>
<td>up</td>
<td>4</td>
<td>[0, 0, 0, 0, 1, 0, 0, 0, 0]</td>
</tr>
<tr>
<td>the</td>
<td>5</td>
<td>[0, 1, 0, 0, 0, 0, 0, 0, 0]</td>
</tr>
<tr>
<td>clock</td>
<td>5</td>
<td>[0, 0, 0, 0, 0, 0, 1, 0, 0]</td>
</tr>
</tbody>
</table>

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

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

The mouse ran up the clock

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>the</td>
<td>mouse</td>
<td>ran</td>
<td>up</td>
<td>clock</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Embedding layer (output dim = 4)

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

[0.236, -0.141, 0.000, 0.045],
[0.006, 0.652, 0.270, -0.556],
[0.305, 0.569, -0.028, 0.496],
[0.421, 0.195, -0.058, 0.477],
[0.236, -0.141, 0.000, 0.045],
[0.844, -0.001, 0.763, 0.201]]

The mouse ran down

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>the</td>
<td>mouse</td>
<td>ran</td>
<td>up</td>
<td>clock</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Embedding layer (output dim = 4)

[1, 2, 3, 6]

[[0.236, -0.141, 0.000, 0.045],
[0.006, 0.652, 0.270, -0.556],
[0.305, 0.569, -0.028, 0.496],
[0.466, -0.326, 0.884, 0.007]]
Text Classification
Top K Features (20K) vs Accuracy

Text Classification

Linear stack of layers

Text Classification

Last layer

Sequence to Sequence (Seq2Seq)

Encoder

Knowledge is power

Decoder

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,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.

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>392k</td>
<td>363k</td>
<td>108k</td>
<td>67k</td>
<td>8.5k</td>
<td>5.7k</td>
<td>3.5k</td>
<td>2.5k</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><strong>86.7/85.9</strong></td>
<td><strong>72.1</strong></td>
<td><strong>91.1</strong></td>
<td><strong>94.9</strong></td>
<td><strong>60.5</strong></td>
<td><strong>86.5</strong></td>
<td><strong>89.3</strong></td>
<td><strong>70.1</strong></td>
<td><strong>81.9</strong></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

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
MultiFiT

Cross-lingual

Multi-task

+ Generation

MT-DNN

XLM

XLMify

Knowledge distillation

MT-DNN

SpanBERT
RoBERTa

Span prediction
Remove NSP

Longer time
Remove NSP
More data

MASS

UniLM

Mass

Permuation LM
Transformer-XL
More data

+Knowledge Graph

Cross-modal

ERNE (Tsinghua)

ERNIE (Baidu)

KnowBert

Neural entity linker

VideoBERT
CiBT
ViLBERT
VisualBERT
B2T2
Uniconer-VL
LXMERT
VL-BERT
UNITER

Defense

Whole Word Masking

Grover

Source: https://github.com/thunlp/PLMpapers
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>Text Summarization</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://summari.es/">https://summari.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>Reading Comprehension</td>
<td>ARC</td>
<td><a href="http://data.allenai.org/arc/">http://data.allenai.org/arc/</a></td>
</tr>
<tr>
<td>Question Answering</td>
<td>CliCR</td>
<td><a href="http://aclweb.org/anthology/N18-1140">http://aclweb.org/anthology/N18-1140</a></td>
</tr>
<tr>
<td>Question Generation</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>RACE</td>
<td><a href="http://www.qizhexie.com/data/RACE_leaderboard">http://www.qizhexie.com/data/RACE_leaderboard</a></td>
</tr>
<tr>
<td></td>
<td>SQuAD</td>
<td><a href="https://rajpurkar.github.io/SQuAD-leaderboard/">https://rajpurkar.github.io/SQuAD-leaderboard/</a></td>
</tr>
<tr>
<td></td>
<td>NarrativeQA</td>
<td><a href="https://github.com/deepmind/narrativeqa">https://github.com/deepmind/narrativeqa</a></td>
</tr>
<tr>
<td></td>
<td>Quasar</td>
<td><a href="https://github.com/bdvingra/quasar">https://github.com/bdvingra/quasar</a></td>
</tr>
<tr>
<td></td>
<td>SearchQA</td>
<td><a href="https://github.com/nyu-dl/SearchQA">https://github.com/nyu-dl/SearchQA</a></td>
</tr>
<tr>
<td>Semantic Parsing</td>
<td>AMR parsing</td>
<td><a href="http://ai.stanford.edu/index.html">http://ai.stanford.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>Sentiment Analysis</td>
<td>IMDB Reviews</td>
<td><a href="http://www.cs.cornell.edu/people/pabo/movie-review-data/">http://www.cs.cornell.edu/people/pabo/movie-review-data/</a></td>
</tr>
<tr>
<td></td>
<td>SST</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yelp Reviews</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Subjectivity Dataset</td>
<td></td>
</tr>
<tr>
<td>Text Classification</td>
<td>AG News</td>
<td><a href="http://www-di.unipi.it/~gulli/AG_corpus_of_news_articles.html">http://www-di.unipi.it/~gulli/AG_corpus_of_news_articles.html</a></td>
</tr>
<tr>
<td></td>
<td>DBpedia</td>
<td><a href="http://wiki.dbpedia.org/Datasets">http://wiki.dbpedia.org/Datasets</a></td>
</tr>
<tr>
<td></td>
<td>TREC</td>
<td><a href="http://trec.nist.gov/data.html">http://trec.nist.gov/data.html</a></td>
</tr>
<tr>
<td></td>
<td>20 NewsGroup</td>
<td><a href="http://qwone.com/~jason/20Newsgroups/">http://qwone.com/~jason/20Newsgroups/</a></td>
</tr>
<tr>
<td>Natural Language Inference</td>
<td>SNLI Corpus</td>
<td><a href="http://www-nlpir.nist.gov/projects/snli/">http://www-nlpir.nist.gov/projects/snli/</a></td>
</tr>
<tr>
<td></td>
<td>MultiNLI</td>
<td><a href="https://www-di.unipi.it/~gulli/AG_corpus_of_news_articles.html">https://www-di.unipi.it/~gulli/AG_corpus_of_news_articles.html</a></td>
</tr>
<tr>
<td></td>
<td>SciTail</td>
<td><a href="http://www.nyu.edu/projects/bowman/multinli/">http://www.nyu.edu/projects/bowman/multinli/</a></td>
</tr>
<tr>
<td>Semantic Role Labeling</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 visually stunning rumination on love”
Reviewer #1

That’s a positive thing to say

“reassembled from the cutting room floor of any given daytime soap”
Reviewer #2

That’s negative
## Sentiment Classification: SST2

### Sentences from movie reviews

<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”

DistilBERT

Logistic Regression

positive

Movie Review Sentiment Classifier Model Training

Movie Review Sentiment Classifier

DistilBERT

Already (pre-)trained

Logistic Regression

We will train in this tutorial

Step # 1 Use distilBERT to Generate Sentence Embeddings

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

#### Sentence Embeddings

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>...</th>
<th>767</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.215</td>
<td>-0.1402</td>
<td>...</td>
<td>0.201</td>
</tr>
</tbody>
</table>

#### Label

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Training set** 75% of examples

#### Sentence Embeddings

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>...</th>
<th>767</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.215</td>
<td>-0.1402</td>
<td>...</td>
<td>0.201</td>
</tr>
</tbody>
</table>

#### Label

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Testing set** 25% of examples

#### Sentence Embeddings

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>...</th>
<th>767</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.124</td>
<td>0.014</td>
<td>...</td>
<td>0.274</td>
</tr>
</tbody>
</table>

#### Label

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

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

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

Tokenization

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

Tokenization

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

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/
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

15% 0 (negative) Model #2 Output 1 (positive)
85%

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>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a stirring, funny and finally...</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>apparently reassembled from the...</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>they presume their audience...</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>this is a visually stunning...</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>jonathan parker's bartleby...</td>
<td>1</td>
</tr>
</tbody>
</table>

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>0</td>
<td>101 1037 ... 0</td>
</tr>
<tr>
<td>1</td>
<td>101 2027 ... 0</td>
</tr>
<tr>
<td>...</td>
<td>... ... ...</td>
</tr>
<tr>
<td>1,999</td>
<td>101 1996 ... 0</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

2,000 Output rows (one per sequence)

66 Tokens in each sequence

768 Number of hidden units

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>1</td>
<td>...</td>
<td>65</td>
<td></td>
</tr>
<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 #ination  on  love [SEP]   0
```

“a visually stunning rumination on love”

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>0</th>
<th>1</th>
<th>...</th>
<th>767</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></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>

label

1
0
1

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%

<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
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

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

Collecting tf-nightly
  Downloading https://files.pythonhosted.org/packages/2a/a0/7381cd278a8e1a9235f032ea811af07bbe31ed45ac9781f21
  517.6MB 24kB/s
Collecting tf-estimator-nightly
  Downloading https://files.pythonhosted.org/packages/0f/fb/984408ab3aee0bddf02c136a4fd76c4e58df128c458e20
  460kB 40.2MB/s
Requirement already satisfied: google-pasta>=0.1.8 in /usr/local/lib/python3.6/dist-packages (from tf-nightly)
Yelp Dataset Download
4.5GB

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

### JSON
- 4.5 gigabytes compressed
- 9.8 gigabytes uncompressed

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

For more information on the JSON dataset, visit the [main dataset documentation page](https://www.yelp.com/dataset/download).

### Photos
- 7.0 gigabytes compressed
- 7.2 gigabytes uncompressed

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

Source: [https://www.yelp.com/dataset/download](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 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 of Classification Models**

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

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

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

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

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

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>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positive (TP)</td>
<td>False Positive (FP)</td>
</tr>
<tr>
<td>False Negative (FN)</td>
<td>True Negative (TN)</td>
</tr>
</tbody>
</table>

Predictive Class (prediction outcome)

<table>
<thead>
<tr>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positive (TP)</td>
<td>False Positive (FP)</td>
</tr>
<tr>
<td>False Negative (FN)</td>
<td>True Negative (TN)</td>
</tr>
</tbody>
</table>

**Accuracy** = \( \frac{TP + TN}{TP + TN + FP + FN} \)

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

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

**Precision** = \( \frac{TP}{TP + FP} \)

**Recall** = \( \frac{TP}{TP + FN} \)

**True Positive Rate (Sensitivity)**

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

**Recall**

\[
Recall = \frac{TP}{TP + FN}
\]
**True Class (actual value)**

<table>
<thead>
<tr>
<th>Predictive Class (prediction outcome)</th>
<th>True Class (actual value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>Positive</td>
</tr>
<tr>
<td></td>
<td>True Positive (TP)</td>
</tr>
<tr>
<td></td>
<td>False Positive (FP)</td>
</tr>
<tr>
<td>Negative</td>
<td>False Negative (FN)</td>
</tr>
<tr>
<td></td>
<td>True Negative (TN)</td>
</tr>
</tbody>
</table>

**Total**

- **P**: Positive
- **N**: Negative
- **P'**: Predicted Positive
- **N'**: Predicted Negative
- **TN**: True Negative
- **FN**: False Negative
- **FP**: False Positive
- **TP**: True Positive

**True Negative Rate**

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

**Specificity**

\[
\text{Specificity} = \text{True Negative Rate} = \frac{TN}{N} = \frac{TN}{(TN + FP)}
\]

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

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

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
= \(2TP / (P + P')\)
= \(2TP / (2TP + FP + FN)\)

\[ F = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \]

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)

F1 score (F-score) (F-measure) is the harmonic mean of precision and recall

ACC = Accuracy = (TP + TN) / (TP + TN + FP + FN)

TPR = 0.63 = 63 / (63 + 28) = 63 / 91

FPR = 0.28 = 28 / (28 + 72) = 28 / 100

PPV = 0.69 = 63 / (63 + 28) = 63 / 91

F1 = 0.66 = 2 * (0.63 * 0.69) / (0.63 + 0.69) = (2 * 63) / (100 + 91) = (0.63 + 0.69) / 2 = 1.32 / 2 = 0.66

ACC = 0.68 = (63 + 72) / 200 = 135 / 200 = 67.5

A

\[
\begin{array}{c|c|c}
& (TP) & (FP) \\
\hline
(TP) & 63 & 28 \\
(FN) & 37 & 72 \\
\hline
100 & 109 & 200
\end{array}
\]

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

B

\[
\begin{array}{c|c|c}
& (TP) & (FP) \\
\hline
(TP) & 77 & 77 \\
(FN) & 23 & 23 \\
\hline
100 & 100 & 200
\end{array}
\]

TPR = 0.77
FPR = 0.77
PPV = 0.50
F1 = 0.61
ACC = 0.50

Recall = True Positive Rate (TPR) = Sensitivity = Hit Rate
Precision = Positive Predictive Value (PPV)

Source: http://en.wikipedia.org/wiki/Receiver_operating_characteristic
**Recall**

- True Positive Rate (TPR)
- Sensitivity
- Hit Rate

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

**Precision**

- Positive Predictive Value (PPV)

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

Summary

• Text Classification

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


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

