文本表達特徵工程
(Feature Engineering for Text Representation)

Time: 2020/05/29 (Fri) (9:10 -12:00)
Place: 國立臺北護理健康大學 (台北市明德路365號) G210
Host: 祝國忠院長 (健康科技學院院長)

Min-Yuh Day
戴敏育
Associate Professor
副教授
Dept. of Information Management, Tamkang University
淡江大學 資訊管理學系

http://mail.tku.edu.tw/myday/
2020-05-29
Topics

1. Core Technologies of Natural Language Processing and Text Mining
2. Artificial Intelligence for Text Analytics: Foundations and Applications
3. Feature Engineering for Text Representation
4. Semantic Analysis and Named Entity Recognition; NER
5. Deep Learning and Universal Sentence-Embedding Models
6. Question Answering and Dialogue Systems
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
Feature Engineering for Text Representation
Text Analytics and Text Mining

TEXT ANALYTICS

- Document Matching
- Link Analysis
- Information Retrieval
- Search Engines
- POS Tagging
- Lemmatization
- Word Disambiguation

Text Mining “Knowledge Discovery in Textual Data”

- Web Mining
  - Web Content Mining
  - Web Structure Mining
  - Web Usage Mining

- Data Mining
  - Classification
  - Clustering
  - Association

- Natural Language Processing

- Statistics
- Machine Learning
- Management Science
- Artificial Intelligence
- Computer Science
- Other Disciplines

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

Documents → Language Detection → Pre-processing

- Tokenize
- POS Tagging
- Token Filtering

EN

CN

Pre-processed Documents

Pre-processed Documents → Build Vocabulary

Bag-of-Words & Vectorization

Word Embeddings

- word2vec
- doc2vec
- GloVe

Machine Learning

(Deep) Neural Network

Task / Output

- Classification
- Sentiment Analysis
- Entity Extraction
- Topic Modeling
- Similarity

Modern NLP Pipeline
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

Source: http://mattfortier.me/2017/01/31/nlp-intro-pt-1-overview/
# Overview of Text Vectorization Methods

<table>
<thead>
<tr>
<th>Vectorization Method</th>
<th>Function</th>
<th>Good For</th>
<th>Considerations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>Counts term frequencies</td>
<td>Bayesian models</td>
<td>Most frequent words not always most informative</td>
</tr>
<tr>
<td>One-Hot Encoding</td>
<td>Binarizes term occurrence (0, 1)</td>
<td>Neural networks</td>
<td>All words equidistant, so normalization extra important</td>
</tr>
<tr>
<td>TF–IDF</td>
<td>Normalizes term frequencies across documents</td>
<td>General purpose</td>
<td>Moderately frequent terms may not be representative of document topics</td>
</tr>
<tr>
<td>Distributed Representations</td>
<td>Context-based, continuous term similarity encoding</td>
<td>Modeling more complex relationships</td>
<td>Performance intensive; difficult to scale without additional tools (e.g., Tensorflow)</td>
</tr>
</tbody>
</table>

Encoding Documents as Vectors

The elephant sneezed at the sight of potatoes.

Bats can see via echolocation. See the bat sight sneeze!

Wondering, she opened the door to the studio.

Token Frequency as Vector Encoding

The elephant sneezed at the sight of potatoes.

Bats can see via echolocation. See the bat sight sneeze!

Wondering, she opened the door to the studio.

One-hot Encoding

The elephant sneezed at the sight of potatoes.

Bats can see via echolocation. See the bat sight sneeze!

Wondering, she opened the door to the studio.

TF-IDF Encoding

The elephant sneezed at the sight of potatoes.

Bats can see via echolocation. See the bat sight sneeze!

Wondering, she opened the door to the studio.

Distributed Representation

The elephant sneezed at the sight of potatoes.

Bats can see via echolocation. See the bat sight sneeze!

Wondering, she opened the door to the studio.

Pipelines for Text Vectorization and Feature Extraction

Feature Unions for Branching Vectorization

HTML Parser → Feature Union

- Entity Extractor → CountVectorizer
- Keyphrase Extractor → TfidfVectorizer

Feature Union → LogisticRegression

Feature Extraction and Union

Python in Google Colab (Python101)

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

- Feature Engineering for Text Representation

- Feature Engineering Text Data - Traditional Strategies

- Import necessary dependencies and settings

```python
import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np
import re
import nltk
import matplotlib.pyplot as plt
pd.options.display.max_colwidth = 200
%matplotlib inline
```

```python
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('stopwords')
```

Downloading package punkt to /root/nltk_data...
Unzipping tokenizers/punkt.zip.

https://tinyurl.com/imtkupython101
corpus = ['The sky is blue and beautiful.',
'Love this blue and beautiful sky!',
'The quick brown fox jumps over the lazy dog.',
'A king\'s breakfast has sausages, ham, bacon, eggs, toast and beans',
'I love green eggs, ham, sausages and bacon!',
'The brown fox is quick and the blue dog is lazy!',
'The sky is very blue and the sky is very beautiful today',
'The dog is lazy but the brown fox is quick!']
labels = ['weather', 'weather', 'animals', 'food', 'food', 'animals', 'weather', 'animals']
corpus = np.array(corpus)
corpus_df = pd.DataFrame({'Document': corpus, 'Category': labels})
corpus_df = corpus_df[['Document', 'Category']]
corpus_df
corpus = np.array(corpus)
corpus_df = pd.DataFrame({'Document': corpus, 'Category': labels})
corpus_df = corpus_df[['Document', 'Category']]
corpus_df

<table>
<thead>
<tr>
<th>Document</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>The sky is blue and beautiful.</td>
<td>weather</td>
</tr>
<tr>
<td>Love this blue and beautiful sky!</td>
<td>weather</td>
</tr>
<tr>
<td>The quick brown fox jumps over the lazy dog.</td>
<td>animals</td>
</tr>
<tr>
<td>A king's breakfast has sausages, ham, bacon, eggs, toast and beans</td>
<td>food</td>
</tr>
<tr>
<td>I love green eggs, ham, sausages and bacon!</td>
<td>food</td>
</tr>
<tr>
<td>The brown fox is quick and the blue dog is lazy!</td>
<td>animals</td>
</tr>
<tr>
<td>The sky is very blue and the sky is very beautiful today</td>
<td>weather</td>
</tr>
<tr>
<td>The dog is lazy but the brown fox is quick!</td>
<td>animals</td>
</tr>
</tbody>
</table>
wpt = nltk.WordPunctTokenizer()
stop_words = nltk.corpus.stopwords.words('english')

def normalize_document(doc):
    # lower case and remove special characters\whitespaces
    doc = re.sub(r'[^a-zA-Z\s]', '', doc, re.I|re.A)
    doc = doc.lower()
    doc = doc.strip()
    # tokenize document
    tokens = wpt.tokenize(doc)
    # filter stopwords out of document
    filtered_tokens = [token for token in tokens if token not in stop_words]
    # re-create document from filtered tokens
    doc = ' '.join(filtered_tokens)
    return doc

normalize_corpus = np.vectorize(normalize_document)
norm_corpus = normalize_corpus(corpus)
norm_corpus

https://tinyurl.com/imtkupython101
from sklearn.feature_extraction.text import CountVectorizer
# get bag of words features in sparse format
cv = CountVectorizer(min_df=0., max_df=1.)
cv_matrix = cv.fit_transform(norm_corpus)
cv_matrix

# view non-zero feature positions in the sparse matrix
print(cv_matrix)

# view dense representation
# warning might give a memory error if data is too big
cv_matrix = cv_matrix.toarray()
cv_matrix

array(
[0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0],
[0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0],
[0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0],
[1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0],
[1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0],
[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0],
[0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]])
# get all unique words in the corpus
vocab = cv.get_feature_names()

# show document feature vectors
pd.DataFrame(cv_matrix, columns=vocab)

<table>
<thead>
<tr>
<th></th>
<th>bacon</th>
<th>beans</th>
<th>beautiful</th>
<th>blue</th>
<th>breakfast</th>
<th>brown</th>
<th>dog</th>
<th>eggs</th>
<th>fox</th>
<th>green</th>
<th>ham</th>
<th>jumps</th>
<th>kings</th>
<th>lazy</th>
<th>love</th>
<th>quick</th>
<th>sausages</th>
<th>sky</th>
<th>toast</th>
<th>today</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```python
# you can set the n-gram range to 1,2 to get unigrams as well as bigrams
bv = CountVectorizer(ngram_range=(2,2))
bv_matrix = bv.fit_transform(norm_corpus)

bv_matrix = bv_matrix.toarray()
vocab = bv.get_feature_names()
pd.DataFrame(bv_matrix, columns=vocab)
```

| bacon | beautiful | sky | beautiful | today | blue | beautiful | dog | brown | eggs | eggs | fox | fox | green | eggs | ham | sausages | ham | jumps | lazy | kings | breakfast |
|-------|-----------|-----|-----------|-------|------|-----------|-----|-------|------|------|-----|-----|--------|------|------|---------|------|-------|-------|--------|
| 0     | 0         | 0   | 0         | 0     | 0    | 0         | 0   | 0     | 0    | 0    | 0   | 0   | 0      | 0    | 0    | 0       | 0    | 0     | 0     | 0      |
| 1     | 0         | 1   | 0         | 0     | 0    | 0         | 0   | 0     | 0    | 0    | 0   | 0   | 0      | 0    | 0    | 0       | 0    | 0     | 0     | 0      |
| 2     | 0         | 0   | 0         | 0     | 0    | 0         | 0   | 1     | 0    | 0    | 0   | 0   | 0      | 0    | 0    | 0       | 0    | 0     | 0     | 0      |
| 3     | 1         | 0   | 0         | 0     | 0    | 0         | 1   | 0     | 0    | 0    | 0   | 0   | 1      | 0    | 0    | 0       | 0    | 0     | 0     | 0      |
| 4     | 0         | 0   | 0         | 0     | 0    | 0         | 0   | 0     | 1    | 0    | 0   | 0   | 0      | 0    | 1    | 0       | 0    | 0     | 0     | 0      |
| 5     | 0         | 0   | 0         | 0     | 0    | 0         | 1   | 0     | 1    | 0    | 0   | 0   | 1      | 0    | 0    | 0       | 0    | 0     | 0     | 0      |
| 6     | 0         | 0   | 0         | 0     | 0    | 0         | 0   | 0     | 0    | 0    | 0   | 0   | 0      | 0    | 0    | 0       | 0    | 0     | 0     | 0      |
| 7     | 0         | 0   | 0         | 0     | 0    | 0         | 1   | 1     | 0    | 0    | 0   | 0   | 1      | 0    | 0    | 0       | 0    | 0     | 0     | 0      |

[https://tinyurl.com/imtkuppython101](https://tinyurl.com/imtkuppython101)
```python
from sklearn.feature_extraction.text import TfidfTransformer

tt = TfidfTransformer(norm='l2', use_idf=True, smooth_idf=True)
tt_matrix = tt.fit_transform(cv_matrix)

tt_matrix = tt_matrix.toarray()
vocab = cv.get_feature_names()
pd.DataFrame(np.round(tt_matrix, 2), columns=vocab)
```

---

https://tinyurl.com/imtkupyter101
from sklearn.feature_extraction.text import TfidfVectorizer

tv = TfidfVectorizer(min_df=0., max_df=1., norm='l2',
                     use_idf=True, smooth_idf=True)
tv_matrix = tv.fit_transform(norm_corpus)
tv_matrix = tv_matrix.toarray()

cols = tv.get_feature_names()

df = pd.DataFrame(np.round(tv_matrix, 2), columns=cols)

https://tinyurl.com/imtkupython101
```python
from scipy.cluster.hierarchy import dendrogram, linkage

Z = linkage(similarity_matrix, 'ward')

pd.DataFrame(Z, columns=['Document Cluster 1', 'Document Cluster 2', 'Distance', 'Cluster Size'], dtype='object')
```

<table>
<thead>
<tr>
<th>Document Cluster 1</th>
<th>Document Cluster 2</th>
<th>Distance</th>
<th>Cluster Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>7</td>
<td>0.253098</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>6</td>
<td>0.308539</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>8</td>
<td>0.386952</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>9</td>
<td>0.489845</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>4</td>
<td>0.732945</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>12</td>
<td>2.69565</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>13</td>
<td>3.45108</td>
</tr>
</tbody>
</table>

[https://tinyurl.com/imtkupython101](https://tinyurl.com/imtkupython101)
```python
1 plt.figure(figsize=(8, 3))
2 plt.title('Hierarchical Clustering Dendrogram')
3 plt.xlabel('Data point')
4 plt.ylabel('Distance')
5 dendrogram(Z)
6 plt.axhline(y=1.0, c='k', ls='--', lw=0.5)
```

<matplotlib.lines.Line2D at 0x7ff7b5d793c8>

https://tinyurl.com/imtkupython101
```python
from scipy.cluster.hierarchy import fcluster
max_dist = 1.0

cluster_labels = fcluster(Z, max_dist, criterion='distance')
cluster_labels = pd.DataFrame(cluster_labels, columns=['ClusterLabel'])

pd.concat([corpus_df, cluster_labels], axis=1)
```

<table>
<thead>
<tr>
<th>Document</th>
<th>Category</th>
<th>ClusterLabel</th>
</tr>
</thead>
<tbody>
<tr>
<td>The sky is blue and beautiful.</td>
<td>weather</td>
<td>2</td>
</tr>
<tr>
<td>Love this blue and beautiful sky!</td>
<td>weather</td>
<td>2</td>
</tr>
<tr>
<td>The quick brown fox jumps over the lazy dog.</td>
<td>animals</td>
<td>1</td>
</tr>
<tr>
<td>A king's breakfast has sausages, ham, bacon, eggs, toast and beans</td>
<td>food</td>
<td>3</td>
</tr>
<tr>
<td>I love green eggs, ham, sausages and bacon!</td>
<td>food</td>
<td>3</td>
</tr>
<tr>
<td>The brown fox is quick and the blue dog is lazy!</td>
<td>animals</td>
<td>1</td>
</tr>
<tr>
<td>The sky is very blue and the sky is very beautiful today</td>
<td>weather</td>
<td>2</td>
</tr>
<tr>
<td>The dog is lazy but the brown fox is quick!</td>
<td>animals</td>
<td>1</td>
</tr>
</tbody>
</table>
```python
from sklearn.decomposition import LatentDirichletAllocation

lda = LatentDirichletAllocation(n_components=3, max_iter=10000, random_state=0)
# lda = LatentDirichletAllocation(n_topics=3, max_iter=10000, random_state=0)
dt_matrix = lda.fit_transform(cv_matrix)
features = pd.DataFrame(dt_matrix, columns=['T1', 'T2', 'T3'])
features
```

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.832191</td>
<td>0.083480</td>
<td>0.084329</td>
</tr>
<tr>
<td>1</td>
<td>0.863554</td>
<td>0.069100</td>
<td>0.067346</td>
</tr>
<tr>
<td>2</td>
<td>0.047794</td>
<td>0.047776</td>
<td>0.904430</td>
</tr>
<tr>
<td>3</td>
<td>0.037243</td>
<td>0.925559</td>
<td>0.037198</td>
</tr>
<tr>
<td>4</td>
<td>0.049121</td>
<td>0.903076</td>
<td>0.047802</td>
</tr>
<tr>
<td>5</td>
<td>0.054902</td>
<td>0.047778</td>
<td>0.897321</td>
</tr>
<tr>
<td>6</td>
<td>0.888287</td>
<td>0.055697</td>
<td>0.056016</td>
</tr>
<tr>
<td>7</td>
<td>0.055704</td>
<td>0.055689</td>
<td>0.888607</td>
</tr>
</tbody>
</table>

[https://tinyurl.com/imtkupython101](https://tinyurl.com/imtkupython101)
```
1  tt_matrix = lda.components_
2  for topic_weights in tt_matrix:
3      topic = [(token, weight) for token, weight in zip(vocab, topic_weights)]
4      topic = sorted(topic, key=lambda x: -x[1])
5      topic = [item for item in topic if item[1] > 0.6]
6      print(topic)
7      print()

```
```python
from sklearn.cluster import KMeans

km = KMeans(n_clusters=3, random_state=0)
k.m.fit_transform(features)
cluster_labels = km.labels_

cluster_labels = pd.DataFrame(cluster_labels, columns=["ClusterLabel"])
pd.concat([corpus_df, cluster_labels], axis=1)
```

<table>
<thead>
<tr>
<th>Document</th>
<th>Category</th>
<th>ClusterLabel</th>
</tr>
</thead>
<tbody>
<tr>
<td>The sky is blue and beautiful.</td>
<td>weather</td>
<td>1</td>
</tr>
<tr>
<td>Love this blue and beautiful sky!</td>
<td>weather</td>
<td>1</td>
</tr>
<tr>
<td>The quick brown fox jumps over the lazy dog.</td>
<td>animals</td>
<td>2</td>
</tr>
<tr>
<td>A king's breakfast has sausages, ham, bacon, eggs, toast and beans</td>
<td>food</td>
<td>0</td>
</tr>
<tr>
<td>I love green eggs, ham, sausages and bacon!</td>
<td>food</td>
<td>0</td>
</tr>
<tr>
<td>The brown fox is quick and the blue dog is lazy!</td>
<td>animals</td>
<td>2</td>
</tr>
<tr>
<td>The sky is very blue and the sky is very beautiful today</td>
<td>weather</td>
<td>1</td>
</tr>
<tr>
<td>The dog is lazy but the brown fox is quick!</td>
<td>animals</td>
<td>2</td>
</tr>
</tbody>
</table>

[https://tinyurl.com/imtkupyter101](https://tinyurl.com/imtkupyter101)
```python
from gensim.models import word2vec

# tokenize sentences in corpus
wpt = nltk.WordPunctTokenizer()
tokenized_corpus = [wpt.tokenize(document) for document in norm_bible]

# Set values for various parameters
feature_size = 100  # Word vector dimensionality
window_context = 30  # Context window size
min_word_count = 1  # Minimum word count
sample = 1e-3  # Downsample setting for frequent words

w2v_model = word2vec.Word2Vec(tokenized_corpus, size=feature_size, window=window_context, min_count=min_word_count, sample=sample, iter=50)

# view similar words based on gensim's model
similar_words = {search_term: [item[0] for item in w2v_model.wv.most_similar([search_term], topn=5)]
for search_term in ['god', 'jesus', 'noah', 'egypt', 'john', 'gospel', 'moses', 'famine']}
similar_words
```

https://tinyurl.com/imtkupython101
w2v_model.wv.most_similar(["search_term"], topn=5)


https://tinyurl.com/imtkupython101
from sklearn.decomposition import PCA

df = pd.read_csv('data.csv')

w2v_feature_array = df['W2V'].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(w2v_feature_array, labels, test_size=0.2, random_state=42)

cols = ['Feature1', 'Feature2', 'Feature3']
corr_matrix = df[cols].corr()

corr_matrix['Feature1'].plot(kind='bar')
plt.show()

from sklearn.decomposition import PCA
pca = PCA(n_components=2, random_state=0)
pcs = pca.fit_transform(w2v_feature_array)

labels = ap.labels_
categories = list(corpus_df['Category'])
plt.figure(figsize=(8, 6))
for i in range(len(labels)):
    label = labels[i]
    color = 'orange' if label == 0 else 'blue' if label == 1 else 'green'
    annotation_label = categories[i]
    x, y = pcs[i]
    plt.scatter(x, y, c=color, edgecolors='k')
    plt.annotate(annotation_label, xy=(x+1e-4, y+1e-3), xytext=(0, 0), textcoords='offset points')

https://tinyurl.com/imtkupyter101
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin  Ming-Wei Chang  Kenton Lee  Kristina Toutanova
Google AI Language
{jacobdevlin, mingweichang, kentonl, kristout}@google.com

BERT uses a bidirectional Transformer.
OpenAI GPT uses a left-to-right Transformer.
ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM to generate features for downstream tasks.
Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

BERT input representation

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

BERT Sequence-level tasks

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

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

BERT Token-level tasks

(c) Question Answering Tasks: SQuAD v1.1

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

### General Language Understanding Evaluation (GLUE) benchmark

**GLUE Test results**

<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&lt;sub&gt;BASE&lt;/sub&gt;</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&lt;sub&gt;LARGE&lt;/sub&gt;</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

Pre-trained word vectors
Word2Vec
wiki.zh.vec (861MB)
332647 word
300 vec

Pre-trained word vectors for 90 languages, trained on Wikipedia using fastText.

These vectors in dimension 300 were obtained using the skip-gram model with default parameters.

https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

Facebook Research FastText

Word2Vec: wiki.zh.vec

(861MB) (332647 word 300 vec)

https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md
Word Embeddings in LSTM RNN

Time Expanded LSTM Network

LSTM Internal States

Fixed length question vector encoded by the LSTM

Word Embeddings

Input Question

Source: https://avisingh599.github.io/deeplearning/visual-qa/
Transformer (Attention is All You Need) (Vaswani et al., 2017)

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

BERT (Bidirectional Encoder Representations from Transformers)

Overall pre-training and fine-tuning procedures for BERT

Pre-training

Fine-Tuning

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

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
Multi-lingual
Transformer
Bert
Bidirectional LM
GPT
Larger model
More data
GPT-2
Defense
Grover

XLM
UDify
MT-DNN
MASS
UniLM
MT-DNN
KD
XLM
RoBERTa
SpanBERT
ERNE (Tsinghua)
XLNet
ERNIE (Baidu)
BERT-wmm

Source: https://github.com/thunlp/PLMpapers

By Xiaozhi Wang & Zhengyan Zhang @THUNLP
Turing Natural Language Generation (T-NLG)

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
Transfer Learning in Natural Language Processing

# 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://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><strong>Reading Comprehension</strong></td>
<td>ARC</td>
<td><a href="http://data.alenai.org/arc/">http://data.alenai.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/BDhingra/quasar">https://github.com/BDhingra/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><strong>Semantic Parsing</strong></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><strong>Sentiment Analysis</strong></td>
<td>IMDB Reviews</td>
<td><a href="http://nlplab.stanford.edu/sentiment/index.html">http://nlplab.stanford.edu/sentiment/index.html</a></td>
</tr>
<tr>
<td></td>
<td>SST</td>
<td><a href="https://www.yelp.com/dataset/challenge">https://www.yelp.com/dataset/challenge</a></td>
</tr>
<tr>
<td></td>
<td>Yelp 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>Subjectivity Dataset</td>
<td></td>
</tr>
<tr>
<td><strong>Text Classification</strong></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="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></td>
<td>20 Newsgroups</td>
<td><a href="http://qwone.com/~jason/20Newsgroups/">http://qwone.com/~jason/20Newsgroups/</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.nyudl/proj/bowman/multinli/">https://www.nyudl/proj/bowman/multinli/</a></td>
</tr>
<tr>
<td></td>
<td>SciTail</td>
<td><a href="http://data.alenai.org/scitail/">http://data.alenai.org/scitail/</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>

Aurélien Géron (2019),
O’Reilly Media, 2019

https://github.com/ageron/handson-ml2
Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

Notebooks
1. The Machine Learning landscape
2. End-to-end Machine Learning project
3. Classification
4. Training Models
5. Support Vector Machines
6. Decision Trees
7. Ensemble Learning and Random Forests
8. Dimensionality Reduction
9. Unsupervised Learning Techniques
10. Artificial Neural Nets with Keras
11. Training Deep Neural Networks
12. Custom Models and Training with TensorFlow
13. Loading and Preprocessing Data
14. Deep Computer Vision Using Convolutional Neural Networks
15. Processing Sequences Using RNNs and CNNs
16. Natural Language Processing with RNNs and Attention
17. Representation Learning Using Autoencoders
18. Reinforcement Learning
19. Training and Deploying TensorFlow Models at Scale

https://github.com/ageron/handson-ml2
Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

https://github.com/ageron/handson-ml2
Sequences using RNNs and CNNs

```python
np.random.seed(43)
series = generate_time_series(1, 50 + 10)
X_new, Y_new = series[:, :50, :], series[:, 50:, :]
Y_pred = model.predict(X_new[:, -1]..., np.newaxis)

plot_multiple_forecasts(X_new, Y_new, Y_pred)
plt.show()
```
An end-to-end open source machine learning platform

The core open source library to help you develop and train ML models. Get started quickly by running Colab notebooks directly in your browser.

Get started with TensorFlow

https://www.tensorflow.org/
TensorFlow

is an

Open Source

Software Library

for

Machine Intelligence

https://www.tensorflow.org/
```python
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([  
    tf.keras.layers.Flatten(input_shape=(28, 28)),  
    tf.keras.layers.Dense(128, activation='relu'),  
    tf.keras.layers.Dropout(0.2),  
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
               loss='sparse_categorical_crossentropy',
               metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

https://www.tensorflow.org/overview/
TensorFlow Playground

Tinker With a **Neural Network** Right Here in Your Browser.
Don’t Worry, You Can’t Break It. We Promise.

http://playground.tensorflow.org/
Tensor

- 3
  - # a rank 0 tensor; this is a **scalar** with shape []
- [1., 2., 3.]
  - # a rank 1 tensor; this is a **vector** with shape [3]
- [[1., 2., 3.], [4., 5., 6.]]
  - # a rank 2 tensor; a **matrix** with shape [2, 3]
- [[[1., 2., 3.], [7., 8., 9.]]
  - # a rank 3 **tensor** with shape [2, 1, 3]

https://www.tensorflow.org/
Scalar

Vector

Matrix

Tensor
Deep Learning and Neural Networks
Deep Learning and Neural Networks

Input Layer
\( (X) \)

Hidden Layer
\( (H) \)

Output Layer
\( (Y) \)
Deep Learning and Neural Networks

Input Layer (X)  

Hidden Layer (H)  

Output Layer (Y)
Deep Learning and Neural Networks

- Input Layer (X)
- Hidden Layers (H)
- Output Layer (Y)

Deep Neural Networks
Deep Learning
Convolutional Neural Networks
(CNN or Deep Convolutional Neural Networks, DCNN)


Source: http://www.asimovinstitute.org/neural-network-zoo/
Recurrent Neural Networks (RNN)

Source: http://www.asimovinstitute.org/neural-network-zoo/
Long / Short Term Memory (LSTM)

Source: http://www.asimovinstitute.org/neural-network-zoo/
Gated Recurrent Units (GRU)


Source: http://www.asimovinstitute.org/neural-network-zoo/
Generative Adversarial Networks (GAN)


Source: http://www.asimovinstitute.org/neural-network-zoo/
Support Vector Machines (SVM)


Source: http://www.asimovinstitute.org/neural-network-zoo/
Neural Networks

Input Layer (X)  Hidden Layer (H)  Output Layer (Y)

Source: https://www.youtube.com/watch?v=bxetV8XRs&index=1&list=PLiaHhY2lBX9hdHaRr6b7XevZtgZRa1PoU
Neural Networks

Input Layer (X)  Hidden Layer (H)  Output Layer (Y)

Source: https://www.youtube.com/watch?v=bxetV8XRgs&index=1&list=PLiahY2lBX9hdHaRt6b7XevZtgZRa1PoU
The Neuron

\[ x_1 \rightarrow w_1 \rightarrow y \]

\[ x_2 \rightarrow w_2 \rightarrow y \]

\[ \ldots \rightarrow \ldots \rightarrow y \]

\[ x_n \rightarrow w_n \rightarrow y \]
The Neuron

\[ y = F \left( \sum_{i} w_{i} x_{i} \right) \]

\[ F(x) = \max(0, x) \]
\[ y = \max \left( 0, -0.21 \times x_1 + 0.3 \times x_2 + 0.7 \times x_3 \right) \]
Neural Networks
Neural Networks

Input Layer (X)  Hidden Layer (H)  Output Layer (Y)

X1

X2

Source: https://www.youtube.com/watch?v=bxet-V8XRs&index=1&list=PLiaHhY2IBX9hdHar6b7XevZtgZRa1PoU
Neural Networks

Input Layer (X)

Hidden Layers (H)

Output Layer (Y)

Deep Neural Networks
Deep Learning

Source: https://www.youtube.com/watch?v=bxetV8XR&s=index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
Neural Networks

Input Layer (X)  Hidden Layer (H)  Output Layer (Y)

Neuron

Synapse

X1

X2

Neuron

Synapse

Source: https://www.youtube.com/watch?v=bxerT-V8XR&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
Neural Networks

Input Layer (X)  Hidden Layer (H)  Output Layer (Y)

Hours
Sleep

Hours
Study

Score

Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
Neural Networks

Input Layer (X)          Hidden Layer (H)          Output Layer (Y)

Source: https://www.youtube.com/watch?v=P2HPcj8lRJE&list=PLjJh1vIseyqGod9wWiydumYl8hOXixNu&index=2
Neural Networks

Input Layer (X)  Hidden Layer (H)  Output Layer (Y)

X1  X2

Source: https://www.youtube.com/watch?v=bxе2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRt6b7XevZtgZRa1P0U
<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hours Sleep</td>
<td>Hours Study</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
</tr>
</tbody>
</table>

Source: https://www.youtube.com/watch?v=bxе2T-V8XR&s=index=1&list=PLLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
<table>
<thead>
<tr>
<th></th>
<th>Hours Sleep</th>
<th>Hours Study</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>3</td>
<td>5</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2</td>
<td>93</td>
</tr>
<tr>
<td>Testing</td>
<td>8</td>
<td>3</td>
<td>?</td>
</tr>
</tbody>
</table>
\[ Y = W X + b \]
Output

\[ Y = WX + b \]

Input

Weights

Trained

Bias

Source: https://www.youtube.com/watch?v=G8eNWzOgqE
$W X + b = Y$

Scores $\rightarrow$ Probabilities

Source: https://www.youtube.com/watch?v=G8eNWzxOgqE
SoftMAX

\[ W \mathbf{x} + b = \mathbf{y} \]

\[
\begin{bmatrix}
2.0 \\
1.0 \\
0.1 \\
\end{bmatrix}
\rightarrow
\begin{bmatrix}
0.7 \\
0.2 \\
0.1 \\
\end{bmatrix}
\]

\[ S(\mathbf{y}_i) = \frac{e^{\mathbf{y}_i}}{\sum_j e^{\mathbf{y}_j}} \]

Logits \rightarrow Scores \rightarrow Probabilities

Source: https://www.youtube.com/watch?v=G8eNWzxOggE
\[ S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} = \frac{e^{2.0}}{e^{2.0} + e^{1.0} + e^{0.1}} = \frac{2.7182^{2.0}}{2.7182^{2.0} + 2.7182^{1.0} + 2.7182^{0.1}} = 0.7 \]

\[ S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} = \frac{e^{1.0}}{e^{2.0} + e^{1.0} + e^{0.1}} = \frac{2.7182^{1.0}}{2.7182^{2.0} + 2.7182^{1.0} + 2.7182^{0.1}} = 0.2 \]

\[ S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} = \frac{e^{0.1}}{e^{2.0} + e^{1.0} + e^{0.1}} = \frac{2.7182^{0.1}}{2.7182^{2.0} + 2.7182^{1.0} + 2.7182^{0.1}} = 0.1 \]

**W X + b = Y**

**Logits** → **Scores** → **Probabilities**

Source: https://www.youtube.com/watch?v=G8eNWzxOgqE
Training a Network

= Minimize the Cost Function

Source: https://www.youtube.com/watch?v=bxetV8XR&index=1&list=PLiThhY2iBX9hdHaRt6b7XevZtgZR1PoU
Training a Network

= Minimize the Cost Function

Minimize the Loss Function

Source: https://www.youtube.com/watch?v=bx6E2T-V8XR&s=index=1&list=PLiaHhY2iBX9hdHaR6b7XevZtgZRa1PoU
Error = Predict Y - Actual Y

Error : Cost : Loss

Test1
Test2
Test3

Source: https://www.youtube.com/watch?v=bx2T-V8XR&index=1&list=PLiaHhY2iBX9hdHaRt6b7XevZtgZRa1PoU
Error = Predict Y - Actual Y

Error : Cost : Loss

Source: https://www.youtube.com/watch?v=bxе2T-V8XR&index=1&list=PLiaHhY2iBX9hdHaRt6b7XevZtgZRa1PoU
Error = Predict Y - Actual Y

Error : Cost : Loss
Activation Functions
Activation Functions

**Sigmoid**

**TanH**

**ReLU**

( Rectified Linear Unit )

\[
f(x) = \max(0, x)
\]

**Sigmoid**

\([-1, 1]\)

**TanH**

\([0, 1]\)

**ReLU**

\([0, 1]\)

f(x) = \max(0, x)

Activation Functions

Sigmoid:
\[ f(x) = \frac{1}{1 + e^{-x}} \]

TanH:
\[ \tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \]

ReLU:
\[ f(x) = \begin{cases} 
0 & \text{for } x < 0 \\
x & \text{for } x \geq 0 
\end{cases} \]
Loss Function
Binary Classification: 2 Class

Activation Function: Sigmoid

Loss Function: Binary Cross-Entropy
Multiple Classification: 10 Class

Activation Function: SoftMAX

Loss Function: Categorical Cross-Entropy
Dropout: a simple way to prevent neural networks from overfitting.

(a) Standard Neural Net

(b) After applying dropout.

Learning Algorithm

While not done:

Pick a random training example "(input, label)"
Run neural network on "input"
Adjust weights on edges to make output closer to "label"

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
\[ y = \max ( 0, -0.21 \times x_1 + 0.3 \times x_2 + 0.7 \times x_3 ) \]
Next time:

\[
y = \max (0, -0.23 \, x_1 + 0.31 \, x_2 + 0.65 \, x_3)
\]

\[
y = \max (0, -0.21 \, x_1 + 0.3 \, x_2 + 0.7 \, x_3)
\]

Weights

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
Optimizer:
Stochastic Gradient Descent (SGD)
This shows a function of 2 variables: real neural nets are functions of hundreds of millions of variables!
Neural Network and Deep Learning

Source: 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning, https://www.youtube.com/watch?v=aircAruvnKk
Gradient Descent
how neural networks learn

Average cost of all training data...

\[
\text{Cost of 8} = (0.18 - 0.00)^2 + (0.29 - 0.00)^2 + (0.58 - 0.00)^2 + (0.77 - 0.00)^2 + (0.20 - 0.00)^2 + (0.36 - 0.00)^2 + (0.93 - 0.00)^2 + (1.00 - 0.00)^2 + (0.95 - 1.00)^2 + (0.35 - 0.00)^2
\]

What's the "cost" of this difference?

Source: 3Blue1Brown (2017), Gradient descent, how neural networks learn | Chapter 2, deep learning, https://www.youtube.com/watch?v=1HZwWFHWa-w
Backpropagation

Source: 3Blue1Brown (2017), What is backpropagation really doing? | Chapter 3, deep learning,
https://www.youtube.com/watch?v=Ilg3gGewQ5U
A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

From image to text

A group of people sitting on a boat in the water.
Convolutional Neural Networks (CNN)
Convolutional Neural Networks (CNN)

Architecture of LeNet-5 (7 Layers) (LeCun et al., 1998)


Convolutional Neural Networks (CNN)

- Convolution
- Pooling
- Fully Connection (FC) (Flattening)
CNN Architecture

Input → Conv → Pool → Conv → Pool → FC → FC → Softmax

CNN Convolution Layer

Convolution is a mathematical operation to merge two sets of information

Input

Filter / Kernel

3x3 convolution

CNN Convolution Layer
Input x Filter --> Feature Map

receptive field: 3x3

```
1x1 1x0 1x1 0 0
0x0 1x1 1x0 1 0
0x1 0x0 1x1 1 1
0 0 1 1 1 0
0 1 1 0 0
```

CNN Convolution Layer
Input x Filter --> Feature Map

receptive field: 3x3

Input x Filter

Feature Map
**CNN Convolution Layer**

Example convolution operation shown in 2D using a 3x3 filter.

CNN Convolution Layer

10 different filters  10 feature maps of size 32x32x1

final output of the convolution layer:
a volume of size 32x32x10

CNN Convolution Layer
Sliding operation at 4 locations

CNN Convolution Layer

two feature maps

**CNN Convolution Layer**

**Stride** specifies how much we move the convolution filter at each step.

![Convolution Layer Diagram]

**Stride 1**

**Feature Map**

**CNN Convolution Layer**

**Stride** specifies how much we move the convolution filter at each step.

![Stride 2](image)

**Feature Map**

CNN Convolution Layer

Stride 1 with Padding

Feature Map

CNN Pooling Layer

Max Pooling

max pool with 2x2 window and stride 2

CNN Architecture

4 convolution + pooling layers, followed by 2 fully connected layers

Input → Conv + Maxpool → Conv + Maxpool → Conv + Maxpool → Conv + Maxpool → FC → FC → Output

CNN Architecture
4 convolution + pooling layers, followed by 2 fully connected layers

https://gist.github.com/ardendertat/0fc5515057c47e7386fe04e9334504e3

model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', name='conv_1',
                           input_shape=(150, 150, 3)))
model.add(MaxPooling2D((2, 2), name='maxpool_1'))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', name='conv_2'))
model.add(MaxPooling2D((2, 2), name='maxpool_2'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', name='conv_3'))
model.add(MaxPooling2D((2, 2), name='maxpool_3'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', name='conv_4'))
model.add(MaxPooling2D((2, 2), name='maxpool_4'))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu', name='dense_1'))
model.add(Dense(128, activation='relu', name='dense_2'))
model.add(Dense(1, activation='sigmoid', name='output'))
Dropout

No Dropout

With Dropout

Model Performance

Train Loss: 0.054, Val Loss: 1.345

Starts Overfitting

Train Accuracy: 0.981, Val Accuracy: 0.732

The activations of an example ConvNet architecture.

http://cs231n.github.io/convolutional-networks/
ConvNets

32x32x3 CIFAR-10 image

first Convolutional layer

http://cs231n.github.io/convolutional-networks/
ConvNets

http://cs231n.github.io/convolutional-networks/
Convolution Demo

Input Volume (+pad 1) (7x7x3)

<table>
<thead>
<tr>
<th>x[:,:,0]</th>
<th>w0[:,:,0]</th>
<th>w1[:,:,0]</th>
<th>o[:,:,0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 0</td>
<td>-1 -1 0</td>
<td>1 -1 0</td>
<td>6 3 6</td>
</tr>
<tr>
<td>0 1 2 0 2 1 0</td>
<td>1 1 1</td>
<td>0 1 1</td>
<td>7 -1 -2</td>
</tr>
<tr>
<td>0 2 2 2 1 1 0</td>
<td>-1 0 1</td>
<td>0 -1 1</td>
<td>2 3 -2</td>
</tr>
<tr>
<td>0 2 2 2 0 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 2 1 2 1 2 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 2 1 2 0 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Filter W0 (3x3x3)

<table>
<thead>
<tr>
<th>w0[:,:,1]</th>
<th>w0[:,:,2]</th>
<th>w1[:,:,1]</th>
<th>w1[:,:,2]</th>
<th>o[:,:,1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>-1 -1</td>
<td>0 0</td>
<td>1 0</td>
<td>7 -1 -3</td>
</tr>
<tr>
<td>0 2 2 1 2 0</td>
<td>1 0</td>
<td>0 0</td>
<td>0 -1</td>
<td>4 3 2</td>
</tr>
<tr>
<td>0 1 2 0 0 2 0</td>
<td>-1 0</td>
<td>1 0</td>
<td>0 1</td>
<td></td>
</tr>
<tr>
<td>0 0 1 2 1 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 2 2 2 2 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 2 2 2 0 2 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Filter W1 (3x3x3)

Output Volume (3x3x2)

<table>
<thead>
<tr>
<th>w1[:,:,0]</th>
<th>w1[:,:,1]</th>
<th>o[:,:,0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 -1 0</td>
<td>0 1 1</td>
<td>2 3 -2</td>
</tr>
<tr>
<td>0 -1 1</td>
<td>0 -1 1</td>
<td>4 3 2</td>
</tr>
<tr>
<td>0 0 0</td>
<td>-1 0</td>
<td>0 1</td>
</tr>
</tbody>
</table>

Bias b0 (1x1x1)

<table>
<thead>
<tr>
<th>b0[:,:,0]</th>
<th>b1[:,:,0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

http://cs231n.github.io/convolutional-networks/
ConvNets

input volume of size $[224 \times 224 \times 64]$ is pooled with filter size 2, stride 2 into output volume of size $[112 \times 112 \times 64]$
ConvNets
max pooling

Single depth slice

max pool with 2x2 filters and stride 2

http://cs231n.github.io/convolutional-networks/
Convolutional Neural Networks (CNN) (LeNet)

Source: [http://deeplearning.net/tutorial/lenet.html](http://deeplearning.net/tutorial/lenet.html)
Recurrent Neural Networks (RNN)
Recurrent Neural Networks (RNN)

Input $X_{t-2}$ $X_{t-1}$ $X_{t+1}$ $X_{t}$ $X_{t+1}$ $X_{t+2}$

Hidden $h_{t-2}$ $h_{t-1}$ $h_{t}$ $h_{t+1}$ $h_{t+2}$

Output $y_{t-2}$ $y_{t-1}$ $y_{t}$ $y_{t+1}$ $y_{t+2}$
Recurrent Neural Networks (RNN)

\[ X_t - X_{t-1} - X_{t-2} \quad h_{t-2} \rightarrow h_{t-1} \rightarrow h_t \rightarrow h_{t+1} \rightarrow h_{t+2} \]

\[ y \leftarrow h_{t+1} \]

input

output
Recurrent Neural Networks (RNN) for Sentiment Analysis

\[
\begin{align*}
X_t &- X_{t-1} - X_{t-2} \\
X_{t+1} &+ X_{t+2} \\
Y &
\end{align*}
\]

Input: This movie is very good

Output:

Diagram showing the flow of hidden states and inputs through time steps.
Recurrent Neural Networks (RNN)

Sentiment Analysis

This movie is very boring
Recurrent Neural Network (RNN)

RNN

RNN long-term dependencies

I grew up in France… I speak fluent French.

Vanishing Gradient

Exploding Gradient

Recurrent Neural Networks (RNN)
RNN
Vanishing Gradient problem
Exploding Gradient problem

if $|W| < 1$ (Vanishing)
if $|W| > 1$ (Exploding)

RNN
Vanishing Gradient problem

\[ W = 0.9 < 1 \text{ (Vanishing)} \]

RNN
Exploding Gradient problem

\[ W = 1.1 > 1 \text{ (Exploding)} \]

Long Short Term Memory (LSTM)

Long Short Term Memory (LSTM)

Gated Recurrent Unit (GRU)
Gated Recurrent Unit (GRU)

reset gate  update gate

$\tilde{h}_t$

$h_t$

$h_{t-1}$

$x_t$
LSTM

LSTM vs GRU

**LSTM**

i, f and o are the input, forget and output gates, respectively. c and c˜ denote the memory cell and the new memory cell content.

**GRU**

r and z are the reset and update gates, and h and h˜ are the activation and the candidate activation.

Long Short Term Memory (LSTM)

Long Short Term Memory (LSTM)

LSTM

Memory state (C)
**LSTM**

*forget gate (f)*

\[
f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)
\]
**LSTM**

**input gate (i)**

\[ i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \]

\[ \tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \]

**LSTM**

**Memory state (C)**

\[ C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \]
LSTM

output gate \( (o) \)

\[
o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)
\]

\[
h_t = o_t \times \tanh(C_t)
\]

LSTM

forget (f), input (i), output (o) gates

\[ f_t = \sigma (W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f) \]
\[ i_t = \sigma (W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i) \]
\[ o_t = \sigma (W_o \cdot [C_t, h_{t-1}, x_t] + b_o) \]

Gated Recurrent Unit (GRU)

update (z), reset (r) gates

\[
\begin{align*}
z_t &= \sigma (W_z \cdot [h_{t-1}, x_t]) \\
r_t &= \sigma (W_r \cdot [h_{t-1}, x_t]) \\
\tilde{h}_t &= \tanh (W \cdot [r_t \cdot h_{t-1}, x_t]) \\
h_t &= (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t
\end{align*}
\]
LSTM Recurrent Neural Network

- **Traditional Neural Network**
- **Music Generation**
- **Sentiment Classification**
- **Name Entity Recognition**
- **Machine Translation**

Source: https://github.com/Vict0rSch/deep_learning/tree/master/keras/recurrent
The Sequence to Sequence model
(seq2seq)
Sequence to Sequence (Seq2Seq)

Encoder

\[ \text{Knowledge} \rightarrow \text{is} \rightarrow \text{power} \rightarrow <\text{end}> \]

Decoder

\[ \text{e}_0 \rightarrow \text{e}_1 \rightarrow \text{e}_2 \rightarrow \text{e}_3 \rightarrow \text{e}_4 \rightarrow \text{e}_5 \rightarrow \text{e}_6 \]

Attention

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

Transformer

INPUT
Je suis étudiant

THE TRANSFORMER

OUTPUT
I am a student

Transformer
Encoder Decoder

Transformer
Encoder Self-Attention

Transformer Decoder

Transformer Encoder with Tensors
Word Embeddings

Transformer
Self-Attention Visualization

Transformer

Positional Encoding Vectors

Transformer
Self-Attention Softmax Output

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

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

Illustrated BERT

1 - **Semi-supervised** training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.

**Semi-supervised Learning Step**

- **Model:** BERT
- **Dataset:** Predict the masked word (language modeling)

2 - **Supervised** training on a specific task with a labeled dataset.

**Supervised Learning Step**

- **Model:** BERT (pre-trained in step #1)
- **Classifier**
- **Dataset:**
  - **Email message**
    - Buy these pills
    - Win cash prizes
    - Dear Mr. Atteides, please find attached...
  - **Class**
    - Spam
    - Spam
    - Not Spam

BERT Classification Input Output

Input Features

Help Prince Mayuko Transfer
Huge Inheritance

Classifier
(Feed-forward neural network + softmax)

Output Prediction

85% Spam
15% Not Spam

Source: Jay Alammar (2019), The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning),
http://jalammar.github.io/illustrated-bert/
BERT Encoder Input

Source: Jay Alammar (2019), The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning), http://jalammar.github.io/illustrated-bert/
BERT Classifier

Source: Jay Alammar (2019), The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning), http://jalammar.github.io/illustrated-bert/
Summary

• 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
References


• Deep Learning Basics: Neural Networks Demystified, https://www.youtube.com/playlist?list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

• Deep Learning SIMPLIFIED, https://www.youtube.com/playlist?list=PLjJh1vlSEYgvGod9wWiydumYl8hOXixNu

• 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning, https://www.youtube.com/watch?v=aircAruvnKk

• 3Blue1Brown (2017), Gradient descent, how neural networks learn | Chapter 2, deep learning, https://www.youtube.com/watch?v=IHZwWFHWa-w

• 3Blue1Brown (2017), What is backpropagation really doing? | Chapter 3, deep learning, https://www.youtube.com/watch?v=Ilg3gGewQ5U


• Jay Alammar (2019), The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning), http://jalammar.github.io/illustrated-bert/

文本表達特徵工程

(Feature Engineering for Text Representation)

Time: 2020/05/29 (Fri) (9:10 -12:00)
Place: 國立臺北護理健康大學 (台北市明德路365號) G210
Host: 祝國忠 院長 (健康科技學院院長)

Min-Yuh Day
戴敏育
Associate Professor
副教授

Dept. of Information Management, Tamkang University

http://mail.tku.edu.tw/myday/
2020-05-29