Artificial Intelligence

Deep Learning for Natural Language Processing

Min-Yuh Day, Ph.D, Associate Professor

Institute of Information Management, National Taipei University

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

1111AI08
MBA, IM, NTPU (M6132) (Fall 2022)
Wed 2, 3, 4 (9:10-12:00) (B8F40)

https://meet.google.com/miy-fbif-max

2022-11-16
<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Subject/Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2022/09/14</td>
<td>Introduction to Artificial Intelligence</td>
</tr>
<tr>
<td>2</td>
<td>2022/09/21</td>
<td>Artificial Intelligence and Intelligent Agents</td>
</tr>
<tr>
<td>3</td>
<td>2022/09/28</td>
<td>Problem Solving</td>
</tr>
<tr>
<td>4</td>
<td>2022/10/05</td>
<td>Knowledge, Reasoning and Knowledge Representation; Uncertain Knowledge and Reasoning</td>
</tr>
<tr>
<td>5</td>
<td>2022/10/12</td>
<td>Case Study on Artificial Intelligence I</td>
</tr>
<tr>
<td>6</td>
<td>2022/10/19</td>
<td>Machine Learning: Supervised and Unsupervised Learning</td>
</tr>
<tr>
<td>Week</td>
<td>Date</td>
<td>Subject/Topics</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>7</td>
<td>2022/10/26</td>
<td>The Theory of Learning and Ensemble Learning</td>
</tr>
<tr>
<td>8</td>
<td>2022/11/02</td>
<td>Midterm Project Report</td>
</tr>
<tr>
<td>9</td>
<td>2022/11/09</td>
<td>Deep Learning and Reinforcement Learning</td>
</tr>
<tr>
<td>10</td>
<td>2022/11/16</td>
<td>Deep Learning for Natural Language Processing</td>
</tr>
<tr>
<td>11</td>
<td>2022/11/23</td>
<td>Invited Talk: AI for Information Retrieval</td>
</tr>
<tr>
<td>12</td>
<td>2022/11/30</td>
<td>Case Study on Artificial Intelligence II</td>
</tr>
</tbody>
</table>
## Syllabus

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Subject/Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>2022/12/07</td>
<td>Computer Vision and Robotics</td>
</tr>
<tr>
<td>14</td>
<td>2022/12/14</td>
<td>Philosophy and Ethics of AI and the Future of AI</td>
</tr>
<tr>
<td>15</td>
<td>2022/12/21</td>
<td>Final Project Report I</td>
</tr>
<tr>
<td>16</td>
<td>2022/12/28</td>
<td>Final Project Report II</td>
</tr>
<tr>
<td>17</td>
<td>2023/01/04</td>
<td>Self-learning</td>
</tr>
<tr>
<td>18</td>
<td>2023/01/11</td>
<td>Self-learning</td>
</tr>
</tbody>
</table>
Deep Learning for Natural Language Processing
Outline

• Word Embeddings
• Recurrent Neural Networks for NLP
• Sequence-to-Sequence Models
• The Transformer Architecture
• Pretraining and Transfer Learning
• State of the art (SOTA)
Stuart Russell and Peter Norvig (2020),
Artificial Intelligence: A Modern Approach,


Artificial Intelligence: A Modern Approach

1. Artificial Intelligence
2. Problem Solving
3. Knowledge and Reasoning
4. Uncertain Knowledge and Reasoning
5. Machine Learning
6. Communicating, Perceiving, and Acting
7. Philosophy and Ethics of AI

Artificial Intelligence: Communicating, perceiving, and acting

Artificial Intelligence:
6. Communicating, Perceiving, and Acting

• Natural Language Processing
• Deep Learning for Natural Language Processing
• Computer Vision
• Robotics

Artificial Intelligence: Natural Language Processing

• Language Models
• Grammar
• Parsing
• Augmented Grammars
• Complications of Real Natural Language
• Natural Language Tasks

Artificial Intelligence: Deep Learning for Natural Language Processing

• Word Embeddings
• Recurrent Neural Networks for NLP
• Sequence-to-Sequence Models
• The Transformer Architecture
• Pretraining and Transfer Learning
• State of the art (SOTA)

Reinforcement Learning (DL)

Agent

Environment

Reinforcement Learning (DL)

Agent

Environment

1 observation

2 action

3 reward

Reinforcement Learning (DL)

1. Observation: $O_t$
2. Action: $A_t$
3. Reward: $R_t$

Agents interact with environments through sensors and actuators.

Figure 2.1: Agents interact with environments through sensors and actuators.

AI Acting Humanly: The Turing Test Approach (Alan Turing, 1950)

• Knowledge Representation
• Automated Reasoning
• Machine Learning (ML)
  • Deep Learning (DL)
• Computer Vision (Image, Video)
• Natural Language Processing (NLP)
• Robotics

Deep Learning for Natural Language Processing
AI for Text Analytics

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
### Foundations
- ML for NLP
- NLP Pipelines
- Data Gathering
- Multilingual NLP
- Text Representation
- Covered in Chapters 1 to 3

### Core Tasks
- Text Classification
- Information Extraction
- Conversational Agents
- Information Retrieval
- Question Answering
- Covered in Chapters 3 to 7

### General Applications
- Spam Classification
- Calendar Event Extraction
- Personal Assistants
- Search Engines
- Jeopardy!
- Covered in Chapters 4 to 7

### Industry Specific
- Social Media Analysis
- Retail Data Extraction
- Health Records Analysis
- Financial Analysis
- Legal Entity Extraction
- Covered in Chapters 8 to 10

### AI Project Playbook
- Project Processes
- Best Practices
- Model Iterations
- MLOps
- AI Teams & Hiring
- Covered in Chapters 2 & 11

NLP with Transformers Github Notebooks

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

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

- **Encoder**
  - DistilBERT
  - BERT
  - RoBERTa
  - XLM-R
  - XLM
  - ALBERT
  - ELECTRA
  - DeBERTa

- **Decoder**
  - T5
  - BART
  - M2M-100
  - BigBird
  - GPT
  - GPT-2
  - CTRL
  - GPT-3
  - GPT-Neo
  - GPT-J
  - BLOOM

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O’Reilly Media.
Language Models Sizes
(GPT-3, PaLM, BLOOM)

Source: https://lifearchitect.ai/models/
"translate English to German: That is good."

"cola sentence: The course is jumping well."

"sts1 sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."

"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi..."

"Das ist gut."

"not acceptable"

"3.8"

"six people hospitalized after a storm in attala county."

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

Fine-tuning BERT on Different Tasks

Sentiment Analysis:
Single Sentence Classification

Fine-tuning BERT on Question Answering (QA)

Fine-tuning BERT on Dialogue Intent Detection (ID; Classification)

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

Fine-tuning BERT on Dialogue Slot Filling (SF)

Task-Oriented Dialogue (ToD) System
Speech, Text, NLP

"Book me a cab to Russell Square"

Conversational AI
to deliver contextual and personal experience to users

Source: Huynh-The, Thien, Quoc-Viet Pham, Xuan-Qui Pham, Thanh Thi Nguyen, Zhu Han, and Dong-Seong Kim (2022).
wav2vec 2.0:
A framework for self-supervised learning of speech representations

Whisper: Robust Speech Recognition via Large-Scale Weak Supervision

Multitask training data (650k hours)

- **English transcription**
  - "Ask not what your country can do for ..."
  - Ask not what your country can do for ...
- **Any-to-English speech translation**
  - "El rápido zorro marrón salta sobre ..."
  - The quick brown fox jumps over ...
- **Non-English transcription**
  - "안녕하세요 그녀의 이름은 쿠키입니다 ..."
  - 안녕하세요 그녀의 이름은 쿠키입니다 ...
- **No speech**
  - (background music playing)
  - 0

Sequence-to-sequence learning

Transformer Encoder Blocks

Transformer Decoder Blocks

Sinusoidal Positional Encoding

Log-Mel Spectrogram

Tokens in Multitask Training Format

Multitask training format

- **PREV**
  - previous text tokens
- **START OF TRANSCRIPT**
  - Custom vocabulary / prompting
  - special tokens
- **TRANSCRIBE**
  - no speech
- **TRANSLATE**
  - no timestamps
- **X → Y**
  - X → English Translation
  - voice activity detection (VAD)

Time-aligned transcription

Text-only transcription (allows dataset-specific fine-tuning)

# Microsoft Azure

## Text to Speech (TTS)

You can replace this text with any text you wish. You can either write in this text box or paste your own text here.

Try different languages and voices. Change the speed and the pitch of the voice. You can even tweak the SSML (Speech Synthesis Markup Language) to control how the different sections of the text sound. Click on SSML above to **give it a try**!

Enjoy using Text to Speech!

### Language

<table>
<thead>
<tr>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>English (United States)</td>
</tr>
</tbody>
</table>

### Voice

<table>
<thead>
<tr>
<th>Voice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jenny (Neural)</td>
</tr>
</tbody>
</table>

### Speaking style

<table>
<thead>
<tr>
<th>Speaking style</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
</tr>
</tbody>
</table>

### Speaking speed

- Speaking speed: 1.00

### Pitch

- Pitch: 0.00

Hugging Face

The AI community building the future.

Build, train and deploy state of the art models powered by the reference open source in machine learning.

https://huggingface.co/
BLOOM

BigScience Large Open-science Open-access Multilingual Language Model

BigScience Large Open-science Open-access Multilingual Language Model
Version 1.3 / 6 July 2022

Current Checkpoint: Training Iteration 95000

Total seen tokens: 366B

Source: https://huggingface.co/bigscience/bloom
Whisper

Whisper is a general-purpose speech recognition model. It is trained on a large dataset of diverse audio and is also a multi-task model that can perform multilingual speech recognition as well as speech translation and language identification. This demo cuts audio after around 30 secs.

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

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

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

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

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

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

Text Classification

```python
from transformers import pipeline

classifier = pipeline("text-classification")

import pandas as pd

outputs = classifier(text)

pd.DataFrame(outputs)
```

<table>
<thead>
<tr>
<th>label</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NEGATIVE</td>
</tr>
</tbody>
</table>
```python
ner_tagger = pipeline("ner", aggregation_strategy="simple")
outputs = ner_tagger(text)
pd.DataFrame(outputs)
```

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

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

<table>
<thead>
<tr>
<th>score</th>
<th>start</th>
<th>end</th>
<th>answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.631292</td>
<td>335</td>
<td>358</td>
<td>an exchange of Megatron</td>
</tr>
</tbody>
</table>
Bumblebee ordered an Optimus Prime action figure from your online store in Germany. Unfortunately, when I opened the package, I discovered to my horror that I had been sent an action figure of Megatron instead.
translator = pipeline("translation_en_to_de",
                     model="Helsinki-NLP/opus-mt-en-de")
outputs = translator(text, clean_up_tokenization_spaces=True, min_length=100)
print(outputs[0]["translation_text"])

from transformers import set_seed
set_seed(42)  # Set the seed to get reproducible results

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

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

Sincerely, Bumblebee.

Customer service response:

Dear Bumblebee, I am sorry to hear that your order was mixed up. The order was completely mislabeled, which is very common in our online store, but I can appreciate it because it was my understanding from this site and our customer service of the previous day that your order was not made correct in our mind and that we are in a process of resolving this matter. We can assure you that your order was
from transformers import pipeline
import pandas as pd
classifier = pipeline("ner")
text = "My name is Michael and I live in Berkeley, California."
outputs = classifier(text)
pd.DataFrame(outputs)
Question Answering

```python
!pip install transformers
from transformers import pipeline
qamodel = pipeline("question-answering")
question = "Where do I live?"
context = "My name is Michael and I live in Taipei."
qamodel(question = question, context = context)

{'answer': 'Taipei', 'end': 39, 'score': 0.9730741381645203, 'start': 33}
```
Question Answering

```python
from transformers import pipeline
qamodel = pipeline("question-answering", model = 'deepset/roberta-base-squad2')
question = "Where do I live?"
context = "My name is Michael and I live in Taipei."
output = qamodel(question = question, context = context)
print(output['answer'])
```

Taipei
from transformers import pipeline
qamodel = pipeline("question-answering", model = 'deepset/roberta-base-squad2')
question = "What causes precipitation to fall?"
context = """In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called "showers"."
"""
output = qamodel(question = question, context = context)
print(output['answer'])
```python
!pip install transformers
from transformers import pipeline
generator = pipeline('text-generation', model = 'gpt2')
generator("Hello, I'm a language model", max_length = 30, num_return_sequences=3)
```

```
[{"generated_text": "Hello, I'm a language model. It's like looking at it, where is each word of the sentence? That's what I mean. Like"},
{"generated_text": "Hello, I'm a language modeler. I'm using this for two purposes: I'm having a lot fewer bugs and faster performance. If I"},
{"generated_text": "Hello, I'm a language model, and I was born to code."}
```

Now, I am thinking about this from a different perspective with a'}

https://tinyurl.com/aintpuppython101
Once upon a time, every person who ever saw Jesus, knew that He was Christ. And even though he might not have known Him, He was
Once upon a time we should be able to speak to people who
have lost children, so we try to take those that have lost
the children to our institutions — but the first time is very
hard for us because of our institutions. To me, it's
important to acknowledge that in an institution of faith and
love they are not children. And that there are many people
who are still hurting the child and there are many in need of
help, if not a system. So I'm very curious
from transformers import pipeline

text2text_generator = pipeline("text2text-generation", model = 't5-base')
outputs = text2text_generator("translate from English to French: I am a student")
print(outputs[0][\'generated_text\'])

I am a student
Je suis un étudiant
Text2Text Generation

```python
from transformers import pipeline

text2text_generator = pipeline("text2text-generation")

text2text_generator("question: What is 42 ? context: 42 is the answer to life, the universe and everything")

['generated_text': 'the answer to life, the universe and everything']
```

[https://tinyurl.com/aintpupython101](https://tinyurl.com/aintpupython101)
NLP

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

Modern NLP Pipeline

Source: http://mattfortier.me/2017/01/31/nlp-intro-2017-1-2-overview/
Deep Learning NLP

Source: http://mattfortier.me/2017/01/31/nlp-intro-pt-1-overview/
Natural Language Processing (NLP) and Text Mining

- Raw text
- Sentence Segmentation
- Tokenization
- Part-of-Speech (POS)
- Stop word removal
- Stemming / Lemmatization
- Dependency Parser
- String Metrics & Matching

---

word’s stem
am → am
having → hav

word’s lemma
am → be
having → have

Source: Nitin Hardeniya (2015), NLTK Essentials, Packt Publishing; Florian Leitner (2015), Text mining - from Bayes rule to dependency parsing
Outline

• Word Embeddings
• Recurrent Neural Networks for NLP
• Sequence-to-Sequence Models
• The Transformer Architecture
• Pretraining and Transfer Learning
• State of the art (SOTA)
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, 1, 0, 0]</td>
</tr>
</tbody>
</table>

[0, 1, 2, 3, 4, 5, 6]
Word embedding
GloVe (trained on 6 billion words of text)
100-dimensional word vectors are projected down onto two dimensions

**Word Embedding model**

answer the question “A is to B as C is to [what]?”

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D = C + (B − A)</th>
<th>Relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>Athens</td>
<td>Greece</td>
<td>Oslo</td>
<td>Norway</td>
<td>Capital</td>
</tr>
<tr>
<td>Astana</td>
<td>Kazakhstan</td>
<td>Harare</td>
<td>Zimbabwe</td>
<td>Capital</td>
</tr>
<tr>
<td>Angola</td>
<td>kwanza</td>
<td>Iran</td>
<td>rial</td>
<td>Currency</td>
</tr>
<tr>
<td>copper</td>
<td>Cu</td>
<td>gold</td>
<td>Au</td>
<td>Atomic Symbol</td>
</tr>
<tr>
<td>Microsoft</td>
<td>Windows</td>
<td>Google</td>
<td>Android</td>
<td>Operating System</td>
</tr>
<tr>
<td>New York</td>
<td>New York Times</td>
<td>Baltimore</td>
<td>Baltimore Sun</td>
<td>Newspaper</td>
</tr>
<tr>
<td>Berlusconi</td>
<td>Silvio</td>
<td>Obama</td>
<td>Barack</td>
<td>First name</td>
</tr>
<tr>
<td>Switzerland</td>
<td>Swiss</td>
<td>Cambodia</td>
<td>Cambodian painter</td>
<td>Nationality</td>
</tr>
<tr>
<td>Einstein</td>
<td>scientist</td>
<td>Picasso</td>
<td>painter</td>
<td>Occupation</td>
</tr>
<tr>
<td>brother</td>
<td>sister</td>
<td>grandson</td>
<td>granddaughter</td>
<td>Family Relation</td>
</tr>
<tr>
<td>Chicago</td>
<td>Illinois</td>
<td>Stockton</td>
<td>California</td>
<td>State</td>
</tr>
<tr>
<td>possibly</td>
<td>impossibly</td>
<td>ethical</td>
<td>unethical</td>
<td>Negative</td>
</tr>
<tr>
<td>mouse</td>
<td>mice</td>
<td>dollar</td>
<td>dollars</td>
<td>Plural</td>
</tr>
<tr>
<td>easy</td>
<td>easiest</td>
<td>lucky</td>
<td>luckiest</td>
<td>Superlative</td>
</tr>
<tr>
<td>walking</td>
<td>walked</td>
<td>swimming</td>
<td>swam</td>
<td>Past tense</td>
</tr>
</tbody>
</table>

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

Embedding layer (output dim = 4)

Source: https://developers.google.com/machine-learning/guides/text-classification/step-3
Feedforward part-of-speech (POS) tagging model

Class = PastTenseVerb

Output Layer

Hidden Layer 2

Hidden Layer 1

Embedding lookup

Yesterday

Embedding lookup

they

Embedding lookup

cut

Embedding lookup

the

Embedding lookup

rope

Universal Sentence Encoder (USE)

• The Universal Sentence Encoder encodes text into high-dimensional vectors that can be used for text classification, semantic similarity, clustering and other natural language tasks.

• The universal-sentence-encoder model is trained with a deep averaging network (DAN) encoder.

Source: https://tfhub.dev/google/universal-sentence-encoder/4
Universal Sentence Encoder (USE) Semantic Similarity

"How old are you?" [0.3, 0.2, ...]
"What is your age?" [0.2, 0.1, ...]
"My phone is good." [0.9, 0.6, ...]

Source: https://tfhub.dev/google/universal-sentence-encoder/4
Universal Sentence Encoder (USE)
Classification

"How old are you?"
[0.3, 0.2, ...]
Confidence is a question (96%) "How old are you?"

"What is your age?"
[0.2, 0.1, ...]
(98%) "What is your age?"

"My phone is good."
[0.9, 0.6, ...]
(7%) "My phone is good."

...
import tensorflow_hub as hub

embed = hub.Module("https://tfhub.dev/google/
  "universal-sentence-encoder/1")

embedding = embed(["The quick brown fox jumps over the lazy dog."])

import tensorflow_hub as hub

module = hub.Module("https://tfhub.dev/google/universal-sentence-encoder-multilingual/1")

multilingual_embeddings = module(["Hola Mundo!", "Bonjour le monde!", "Ciao mondo!
"Hello World!", "Hallo Welt!", "Hallo Wereld!",
"你好世界!", "Привет, мир!", "مرحبا بالعالم!"])
Python in Google Colab (Python101)

https://colab.research.google.com/drive/1FEG6DnGwvfwUbeo4z1zTunjMqf2RkCrT

https://tinyurl.com/aintpupython101
Python in Google Colab (Python101)

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

https://tinyurl.com/aintpuppython101
Outline

• Word Embeddings
• Recurrent Neural Networks for NLP
• Sequence-to-Sequence Models
• The Transformer Architecture
• Pretraining and Transfer Learning
• State of the art (SOTA)
RNN
Bidirectional RNN network for POS tagging

Class = Adverb
Feedforward
RNN
RNN
Embedding lookup
Yesterday

Class = Pronoun
Feedforward
RNN
RNN
Embedding lookup
they

Class = PastTenseVerb
Feedforward
RNN
RNN
Embedding lookup
cut

Class = Determiner
Feedforward
RNN
RNN
Embedding lookup
the

Class = Noun
Feedforward
RNN
RNN
Embedding lookup
rope

LSTM Recurrent Neural Network

-one to one

Traditional Neural Network

-one to many

Music Generation

-many to one

Sentiment Classification

-many to many

Name Entity Recognition

-many to many

Machine Translation

Source: https://github.com/Vict0rSch/deep_learning/tree/master/keras/recurrent
Outline

• Word Embeddings
• Recurrent Neural Networks for NLP
• **Sequence-to-Sequence Models**
• The Transformer Architecture
• Pretraining and Transfer Learning
• State of the art (SOTA)
Sequence-to-Sequence model

Attentional Sequence-to-Sequence model for English-to-Spanish translation

The Sequence to Sequence model (seq2seq)

Source: http://suriyadeepan.github.io/2016-12-31-practical-seq2seq/
Sequence to Sequence (Seq2Seq)

Source: https://google.github.io/seq2seq/
Outline

• Word Embeddings
• Recurrent Neural Networks for NLP
• Sequence-to-Sequence Models
• The Transformer Architecture
• Pretraining and Transfer Learning
• State of the art (SOTA)
Single-layer Transformer consists of self-attention, a feedforward network, and residual connection.
Transformer Architecture for POS Tagging

Class = Adverb
Feedforward

Class = Pronoun
Feedforward

Class = PastTenseVerb
Feedforward

Class = Determiner
Feedforward

Class = Noun
Feedforward

Transformer Layer

Transformer Layer

Transformer Layer

Positional Embedding 1 + Embedding lookup + Yesterday

Positional Embedding 2 + Embedding lookup + they

Positional Embedding 3 + Embedding lookup + cut

Positional Embedding 4 + Embedding lookup + the

Positional Embedding 5 + Embedding lookup + rope

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

INPUT: Je suis étudiant

OUTPUT: I am a student

Transformer

Encoder Decoder Stack

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

Outline

• Word Embeddings
• Recurrent Neural Networks for NLP
• Sequence-to-Sequence Models
• The Transformer Architecture
• Pretraining and Transfer Learning
• State of the art (SOTA)
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

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

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

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

**BERT**
(Bidirectional Encoder Representations from Transformers)

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**

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

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

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

Fine-tuning BERT on Question Answering (QA)

(c) Question Answering Tasks: SQuAD v1.1

Fine-tuning BERT on Dialogue Intent Detection (ID; Classification)

Fine-tuning BERT on Dialogue Slot Filling (SF)

# 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>BERTBASE</td>
<td>84.6/83.4</td>
<td>71.2</td>
<td>90.1</td>
<td>93.5</td>
<td>52.1</td>
<td>85.8</td>
<td>88.9</td>
<td>66.4</td>
<td>79.6</td>
</tr>
<tr>
<td>BERTLARGE</td>
<td>86.7/85.9</td>
<td>72.1</td>
<td>91.1</td>
<td>94.9</td>
<td>60.5</td>
<td>86.5</td>
<td>89.3</td>
<td>70.1</td>
<td>81.9</td>
</tr>
</tbody>
</table>

- **MNLI**: Multi-Genre Natural Language Inference
- **QQP**: Quora Question Pairs
- **QNLI**: Question Natural Language Inference
- **SST-2**: The Stanford Sentiment Treebank
- **CoLA**: The Corpus of Linguistic Acceptability
- **STS-B**: The Semantic Textual Similarity Benchmark
- **MRPC**: Microsoft Research Paraphrase Corpus
- **RTE**: Recognizing Textual Entailment

Training Contextual Representations
using a left-to-right Language Model

Masked Language Modeling:
Pretrain a Bidirectional Model

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](image)

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

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

Supervised Learning Step

- **Model:** BERT (pre-trained in step #1)
- **Dataset:**

<table>
<thead>
<tr>
<th>Email message</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buy these pills</td>
<td>Spam</td>
</tr>
<tr>
<td>Win cash prizes</td>
<td>Spam</td>
</tr>
<tr>
<td>Dear Mr. Alreides, please find attached…</td>
<td>Not Spam</td>
</tr>
</tbody>
</table>

BERT Classification Input Output

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

[Image of BERT Encoder Input with layers and tokens]

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/
Sentiment Analysis:
Single Sentence Classification
A Visual Guide to Using BERT for the First Time

(Jay Alammar, 2019)

# Sentiment Classification: SST2

Sentences from movie reviews

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

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 imagining 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>

<table>
<thead>
<tr>
<th>Sentence Embeddings</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-0.215</td>
<td>1</td>
</tr>
<tr>
<td>-0.1402</td>
<td>...</td>
</tr>
<tr>
<td>0.201</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>-0.172</td>
<td>0</td>
</tr>
<tr>
<td>-0.144</td>
<td>...</td>
</tr>
<tr>
<td>0.371</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>1,999</td>
<td>1</td>
</tr>
<tr>
<td>0.124</td>
<td></td>
</tr>
<tr>
<td>0.014</td>
<td>...</td>
</tr>
<tr>
<td>0.274</td>
<td></td>
</tr>
</tbody>
</table>

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>-0.215</td>
</tr>
<tr>
<td>1</td>
<td>-0.1402</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>

Tokenization

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

1) Break words into tokens
2) Add [CLS] and [SEP] tokens

“a visually stunning rumination on love”

Tokenization

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

1) Break words into tokens
2) Add [CLS] and [SEP] tokens
3) Substitute tokens with their ids

"a visually stunning rumination on love"

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 #2

Logistic Regression

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 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 wo n'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)))

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/
# 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

Sentence to last_hidden_state[0]

BERT’s output for the [CLS] tokens

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

Sentence Embeddings

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/
Dataset for Logistic Regression
(768 Features)

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

labels = df[1]
train_features, test_features, train_labels, test_labels =
train_test_split(features, labels)

<table>
<thead>
<tr>
<th>Sentence Embeddings</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>-0.215</td>
<td>0.1402</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>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>1,999</td>
<td>0.124</td>
</tr>
</tbody>
</table>

Score Benchmarks
Logistic Regression Model on SST-2 Dataset

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

# Testing
lr_clf.score(test_features, test_labels)

# Accuracy: 81%
# Highest accuracy: 96.8%
# Fine-tuned DistilBERT: 90.7%
# Full size BERT model: 94.9%

<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

Pre-trained Language Model (PLM)

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

• Word Embeddings
• Recurrent Neural Networks for NLP
• Sequence-to-Sequence Models
• The Transformer Architecture
• Pretraining and Transfer Learning
• State of the art (SOTA)
The Transformers Timeline

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

Transformer

Encoder  Decoder

DistilBERT  BERT  T5  GPT

RoBERTa  XLM  BART  GPT-2  CTRL

XLM-R  XLM  M2M-100  GPT-3

ALBERT  ELECTRA  BigBird  GPT-Neo

DeBERTa  DeBERTa  BigBird  GPT-J

ALBERT  ELECTRA  DeBERTa

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
Language Models Sizes
(GPT-3, PaLM, BLOOM)

Source: https://lifearchive.ai/models/
Pre-trained Models (PTM)

Pre-trained Models (PTM)

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
<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://summar.es/">https://summar.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>CiCIR</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.qizhixie.com/data/RACE_leaderboard">http://www.qizhixie.com/data/RACE_leaderboard</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></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>Semantic Parsing</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>Sentiment Analysis</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>Subjectivity Dataset</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>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>Natural Language Inference</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>MultiNL1</td>
<td><a href="https://www.nyu.edu/projects/bowman/multinli/">https://www.nyu.edu/projects/bowman/multinli/</a></td>
</tr>
<tr>
<td></td>
<td>SciTail</td>
<td><a href="http://data.allenai.org/scitail/">http://data.allenai.org/scitail/</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>

Question Answering (QA)
SQuAD
Stanford Question Answering Dataset
SQuAD

SQuAD 2.0
The Stanford Question Answering Dataset

What is SQuAD?
Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage, or the question might be unanswerable.

SQuAD 2.0 combines the 100,000 questions in SQuAD 1.1 with over 50,000 unanswerable questions written adversarially by crowdworkers to look similar to answerable ones. To do well on SQuAD 2.0, systems must not only answer questions when possible, but also determine when no answer is supported by the paragraph and abstain from answering.

Leaderboard
SQuAD 2.0 tests the ability of a system to not only answer reading comprehension questions, but also abstain when presented with a question that cannot be answered based on the provided paragraph.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Model</th>
<th>EM</th>
<th>F1</th>
</tr>
</thead>
</table>
| 1    | SA-Net on Albert (ensemble)  
(QIANXIN) | 90.724 | 93.011 |
| 2    | SA-Net-V2 (ensemble)  
(QIANXIN) | 90.679 | 92.948 |
| 3    | Retro-Reader (ensemble) | 90.578 | 92.978 |

https://rajpurkar.github.io/SQuAD-explorer/
Abstract

We present the Stanford Question Answering Dataset (SQuAD), a new reading comprehension dataset consisting of 100,000+ questions posed by crowdworkers on a set of Wikipedia articles, where the answer to each question is a segment of text from the corresponding reading passage. We analyze the dataset to understand the types of reasoning required to answer the questions, learning heavily on dependency and constituency trees. We build a strong logistic regression model, which achieves an F1 score of 51.0%, a significant improvement over a simple baseline (20%). However, human performance (86.8%) is much higher, indicating that the dataset presents a good challenge problem for future research. The dataset is freely available at https://stanford-qa.com.

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called 'showers'.

What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud

Figure 1: Question-answer pairs for a sample passage in the
In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity from clouds. The main forms of precipitation include drizzle, rain, sleet, snow, ice pellets, graupel and hail. Precipitation occurs when a portion of the atmosphere becomes saturated with water vapor (reaching 100% relative humidity), so that the water condenses and "precipitates". Thus, fog and mist are not precipitation but suspensions, because the water vapor does not condense sufficiently to precipitate. Two processes, possibly acting together, can lead to air becoming saturated: cooling the air or adding water vapor to the air. Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called "showers."
In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called “showers”.

Q: What causes precipitation to fall?
In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called “showers”.

Q: What causes precipitation to fall?
A: gravity
In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called “showers”.

Q: What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?

A: graupel
In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called “showers”.

Q: Where do water droplets collide with ice crystals to form precipitation?

A: within a cloud
In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called “showers”.

Q: What causes precipitation to fall?
A: gravity

Q: What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
A: graupel

Q: Where do water droplets collide with ice crystals to form precipitation?
A: within a cloud
Natural Language Processing with Python
– Analyzing Text with the Natural Language Toolkit

Steven Bird, Ewan Klein, and Edward Loper

This version of the NLTK book is updated for Python 3 and NLTK 3. The first edition of the book, published by O'Reilly, is available at http://nltk.org/book_1ed/. (There are currently no plans for a second edition of the book.)

0. Preface
1. Language Processing and Python
2. Accessing Text Corpora and Lexical Resources
3. Processing Raw Text
4. Writing Structured Programs
5. Categorizing and Tagging Words (minor fixes still required)
6. Learning to Classify Text
7. Extracting Information from Text
8. Analyzing Sentence Structure
9. Building Feature Based Grammars
10. Analyzing the Meaning of Sentences (minor fixes still required)
11. Managing Linguistic Data (minor fixes still required)
12. Afterword: Facing the Language Challenge

Bibliography
Term Index

This book is made available under the terms of the Creative Commons Attribution Noncommercial No-Derivative-Works 3.0 US License. Please post any questions about the materials to the nltk-users mailing list. Please report any errors on the issue tracker.

http://www.nltk.org/book/
spaCy

Industrial-Strength Natural Language Processing in Python

Fastest in the world
spaCy excels at large-scale information extraction tasks. It’s written from the ground up in carefully memory-managed Cython. Independent research has confirmed that spaCy is the fastest in the world. If your application needs to process entire web dumps, spaCy is the library you want to be using.

Get things done
spaCy is designed to help you do real work — to build real products, or gather real insights. The library respects your time, and tries to avoid wasting it. It’s easy to install, and its API is simple and productive. I like to think of spaCy as the Ruby on Rails of Natural Language Processing.

Deep learning
spaCy is the best way to prepare text for deep learning. It interoperates seamlessly with TensorFlow, Keras, Scikit-Learn, Gensim and the rest of Python’s awesome AI ecosystem. spaCy helps you connect the statistical models trained by these libraries to the rest of your application.

https://spacy.io/
gensim

https://radimrehurek.com/gensim/
TextBlob: Simplified Text Processing

Release v0.12.0 (Changelog)

TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

```python
from textblob import TextBlob
text = "The titular threat of The Blob has always struck me as the ultimate movie monster: an insatiably hungry, amoeba-like mass able to penetrate virtually any safeguard, capable of—"as a doomed doctor chillingly describes it—"assimilating flesh on contact. Snide comparisons to gelatin be damned, it's a concept with the most devastating of potential consequences, not unlike the grey goo scenario proposed by technological theorists fearful of artificial intelligence run rampant.
"

blob = TextBlob(text)

# [('The', 'DT'), ('titular', 'JJ'), ('threat', 'NN'), ('of', 'IN'), ...]

blob.noun_phrases = WordList(['titular threat', 'blob', 'ultimate movie monster', 'amoeba-like mass', ...])

for sentence in blob.sentences:
    print(sentence.sentiment.polarity)
    # 0.050
```

https://textblob.readthedocs.io
Welcome to polyglot’s documentation!

polyglot

Polyglot is a natural language pipeline that supports massive multilingual applications.

- Free software: GPLv3 license
- Documentation: http://polyglot.readthedocs.org

Features

- Tokenization (165 Languages)
- Language detection (196 Languages)
- Named Entity Recognition (40 Languages)
- Part of Speech Tagging (16 Languages)
- Sentiment Analysis (136 Languages)
- Word Embeddings (137 Languages)
- Morphological analysis (135 Languages)
- Transliteration (69 Languages)

https://polyglot.readthedocs.io/
Hugging Face Tasks
Natural Language Processing

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

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

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

https://github.com/nlp-with-transformers/notebooks
Python in Google Colab (Python101)

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

NLP with Transformers

https://tinyurl.com/aintpupython101
Python in Google Colab (Python101)

https://colab.research.google.com/drive/1FEG6DnGvwwfUbeo4zJ1zTunjMqf2RkCrT

Text Classification

https://tinyurl.com/aintpupython101
Python in Google Colab (Python101)

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

Named Entity Recognition (NER)

https://tinyurl.com/aintpupython101
Text Summarization

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

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

```python
# Source: https://huggingface.co/tasks/summarization
!pip install transformers
def pipeline():
    return pipeline("summarization")
def classify(text):
    return "paris is the capital and most populous city of france, with an estimated population of 2,175,601 residents as of 2018, in an area of more than 6.6 million"(text, max_length=30)

No model was supplied, default to sshleifer/distilbart-cnn-12-6 (https://huggingface.co/ssshleifer/distilbart-cnn-12-6)
Your min_length=56 must be inferior than your max_length=30.

[{'summary_text': 'Paris is the capital and most populous city of France, with an estimated population of 2,175,601 residents. The City of Paris'}]
```

https://tinyurl.com/aintpupython101
Python in Google Colab (Python101)

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

Text Generation

https://tinyurl.com/aintpupython101
Python in Google Colab (Python101)

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

Question Answering and Dialogue Systems

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

Question Answering

```python
1!pip install transformers
2 from transformers import pipeline
3 qamodel = pipeline("question-answering")
4 question = "Where do I live?"
5 context = "My name is Michael and I live in Taipei."
6 qamodel(question = question, context = context)
```

https://tinyurl.com/aintpupython101
Python in Google Colab (Python101)

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

Question Answering

https://tinyurl.com/aintpupython101
Python in Google Colab (Python101)

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

Question Answering and Dialogue Systems

Question Answering (QA)

- **BERT for Question Answering**

  Source: Apoorv Nandan (2020), BERT (from HuggingFace Transformers) for Text Extraction, https://keras.io/examples/nlp/text_extraction_with_bert/

  **Description:** Fine tune pretrained BERT from HuggingFace Transformers on SQuAD.

  **Introduction**

  This demonstration uses SQuAD (Stanford Question-Answering Dataset). In SQuAD, an input consists of a question, and a paragraph for context. The goal is to find the span of text in the paragraph that answers the question. We evaluate our performance on this data with the "Exact Match" metric, which measures the percentage of predictions that exactly match any one of the ground-truth answers.

  We fine-tune a BERT model to perform this task as follows:

  1. Feed the context and the question as inputs to BERT.
  2. Take two vectors S and T with dimensions equal to that of hidden states in BERT.
  3. Compute the probability of each token being the start and end of the answer span. The probability of a token being the start of the answer is given by a dot product between S and the representation of the token in the last layer of BERT, followed by a softmax over all tokens. The probability of a token being the end of the answer is compute similarly with the vector T.
  4. Fine-tune BERT and learn S and T along the way.

  **References:**
  - BERT
  - SQuAD

https://tinyurl.com/aintpupython101
Python in Google Colab (Python101)

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

https://tinyurl.com/aintpupython101
Python in Google Colab (Python101)

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

Dialogue Systems

Joint Intent Classification and Slot Filling with Transformers

The goal of this notebook is to fine-tune a pretrained transformer-based neural network model to convert a user query expressed in English into a representation that is structured enough to be processed by an automated service.

Here is an example of interpretation computed by such a Natural Language Understanding system:

```python
>>> nlu("Book a table for two at Le Ritz for Friday night",
    tokenizer, joint_model, intent_names, slot_names)

`

```

Intent classification is a simple sequence classification problem. The trick is to treat the structured knowledge extraction part ("Slot Filling") as token-level classification problem using BIO-annotations:

https://tinyurl.com/aintpupython101
Python in Google Colab (Python101)

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

<table>
<thead>
<tr>
<th>Table of contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentiment Analysis</td>
</tr>
<tr>
<td>Sentiment Analysis - Unsupervised</td>
</tr>
<tr>
<td>Sentiment Analysis - Unsupervised Lexical</td>
</tr>
<tr>
<td>Sentiment Analysis - Supervised</td>
</tr>
<tr>
<td>Sentiment Analysis - Supervised Machine Learning</td>
</tr>
<tr>
<td>Sentiment Analysis - Supervised Deep Learning Models</td>
</tr>
<tr>
<td>Sentiment Analysis - Advanced Deep Learning</td>
</tr>
<tr>
<td>Deep Learning and Universal Sentence-Embedding Models</td>
</tr>
<tr>
<td>Universal Sentence Encoder (USE)</td>
</tr>
<tr>
<td>Universal Sentence Encoder Multilingual (USEM)</td>
</tr>
<tr>
<td>Question Answering and Dialogue Systems</td>
</tr>
<tr>
<td>Question Answering (QA)</td>
</tr>
<tr>
<td>BERT for Question Answering</td>
</tr>
<tr>
<td>Dialogue Systems</td>
</tr>
<tr>
<td>Joint Intent Classification and Slot Filling with Transformers</td>
</tr>
</tbody>
</table>

```python
def show_predictions(text, tokenizer, model, intent_names, slot_names):
    inputs = tf.constant(tokenizer.encode(text))[None, :]
    outputs = model(inputs)
    slot_logits, intent_logits = outputs
    slot_ids = slot_logits.numpy().argmax(axis=1)[0, 1:1]
    intent_id = intent_logits.numpy().argmax(axis=1)[0]
    print("Text:", text)
    print("Intent:", intent_names[intent_id])
    print("Slots:")
    for token, slot_id in zip(tokenizer.tokenize(text), slot_ids):
        print(f"{token:>10} : {slot_names[slot_id]}")
show_predictions("Book a table for two at Le Ritz for Friday night!",
tokenizer, joint_model, intent_names, slot_names)
```

Text: Book a table for two at Le Ritz for Friday night!
Intent: BookRestaurant
Slots:
- Book: 0
- a: 0
- table: 0
- for: 0
- two: B-party_size_number
- at: 0
- Le: B-restaurant_name
- B: restaurant_name
- night: 0
- I: 0

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

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4z1zTunjMqf2RkCrT

https://tinyurl.com/aintpupy.python101
Python in Google Colab (Python101)

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

https://tinyurl.com/aintpupython101
Python in Google Colab (Python101)

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

https://tinyurl.com/aintpupython101
Python in Google Colab (Python101)

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

https://tinyurl.com/aintpupython101
Python in Google Colab (Python101)

https://colab.research.google.com/drive/1FEG6DnGvxfUbeo4zJ1zTunjMqf2RkCrT

https://tinyurl.com/aintpupython101
Python in Google Colab (Python101)

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

https://tinyurl.com/aintpuppython101
Python in Google Colab (Python101)
https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT

https://tinyurl.com/aintpupython101
Python in Google Colab (Python101)
https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT

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

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

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

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

https://tinyurl.com/aintpupyteron101
from spacy import dispyacy

text = "Stanford University is located in California. It is a great university."

doc = nlp(text)

displacy.render(doc, style="ent", jupyter=True)

displacy.render(doc, style="dep", jupyter=True)

Stanford University  **ORG**  is located in  **California GPE**  . It is a great university.

https://tinyurl.com/aintpupython101
Python in Google Colab (Python101)

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

https://tinyurl.com/aintpuppython101
Summary

• Word Embeddings
• Recurrent Neural Networks for NLP
• Sequence-to-Sequence Models
• The Transformer Architecture
• Pretraining and Transfer Learning
• State of the art (SOTA)
References

- Steven D'Ascoli (2022), Artificial Intelligence and Deep Learning with Python: Every Line of Code Explained For Readers New to AI and New to Python, Independently published.