Text Mining / NLP in Azure

Julian Lee
ANZ AI
GBBs

ANZ
AI-Hub

Architecture
Braintrust

Session NN of 11
November 2022
AI Lighthouse Tech. Enablement Series

Agenda

- Introduction to me
- Use Case Overview – Downer Use Case
- AzureML Nuts and Bolts - Labs
- Introduction to NLP/ Text Mining (Simple SpaCY examples) - Labs
- Text Data Labelling + AutoML NLP - Labs
- Hugging Face + Azure OpenAI - Presentation
- Summary and Resources (5 min)
- Q&A

Innovation is in our culture… developing large AI NLG models

Turing-NLG: A 17-billion-parameter language model by Microsoft

Published February 13, 2020

By Corby Rosset, Applied Scientist

This figure was adapted

Turing Natural Languag

parameter language mo

‹ Return to Blog Home

Microsoft Research Blog

Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, the World's Largest and Most Powerful Generative Language Model

Published October 11, 2021

By Ali Alvi, Group Program Manager (Microsoft Turing); Paresh Kharya, Senior Director of Product Management, Accelerated Computing, NVIDIA

Research Area

Artificial intelligence

our vision

Jennifer Langston

May 19, 2020

Microsoft has built one of the top five publicly disclosed supercomputers in the world, making new infrastructure available in Azure to train extremely large artificial intelligence models, the company is announcing at its Build developers conference.

Built in collaboration with and exclusively for OpenAI, the supercomputer hosted in Azure was designed specifically to train that company's AI models. It represents a key milestone in a partnership announced last year to jointly create new supercomputing technologies in Azure.

It's also a first step toward making the next generation of very large AI models and the infrastructure needed to train them available as a platform for other organizations and developers to build upon.

"The exciting thing about these models is the breadth of things they're going to enable," said Microsoft Chief Technical Officer Kevin Scott, who said the potential benefits extend far beyond narrow advances in one type of AI model.

"This is about being able to do a hundred exciting things in natural language processing at once and a hundred exciting things in computer vision, and when you start to see combinations of these perceptual domains, you're going to have new applications that are hard to even imagine right now," he said.

Downer NLP – Risk Management

OpenAI Content Generation

The below examples are actual responses from the OpenAI (Davinci GPT-3 engine) with no specific training on an incident dataset.

User to OpenAi: Rewrite the following in incident report style in third person: *"Bumped helmet into leveling bar. Carrying ladder in the pit passed under the first bar and hit the second bar, knocking the headlamp back and helmet off. threw my head back and had to sit down for a minute to recover."*

OpenAI to User: *"While carrying a ladder in the pit, the individual bumped their helmet into a leveling bar. The ladder then hit a second bar, knocking the headlamp back and the helmet off the individual's head. The individual then threw their head back and had to sit down for a minute to recover."*

User to OpenAi: Write a technical description of a Fall arrest system:

OpenAI to User: *"A fall arrest system is a device used to arrest a person in a free fall. The system consists of an anchor point, a body harness, a lanyard, and a deceleration device. The anchor point is typically a steel ring or bar, and is attached to a structure above the person. The body harness is worn by the person and is attached to the anchor point. The lanyard is a length of rope or webbing that connects the body harness to the deceleration device. The deceleration device is typically a friction brake or a shock absorber, and is designed to arrest the person's fall and prevent them from hitting the ground."*

User to OpenAi: Write a paragraph about fall arrest system requirements in the style of a corporate procedure:

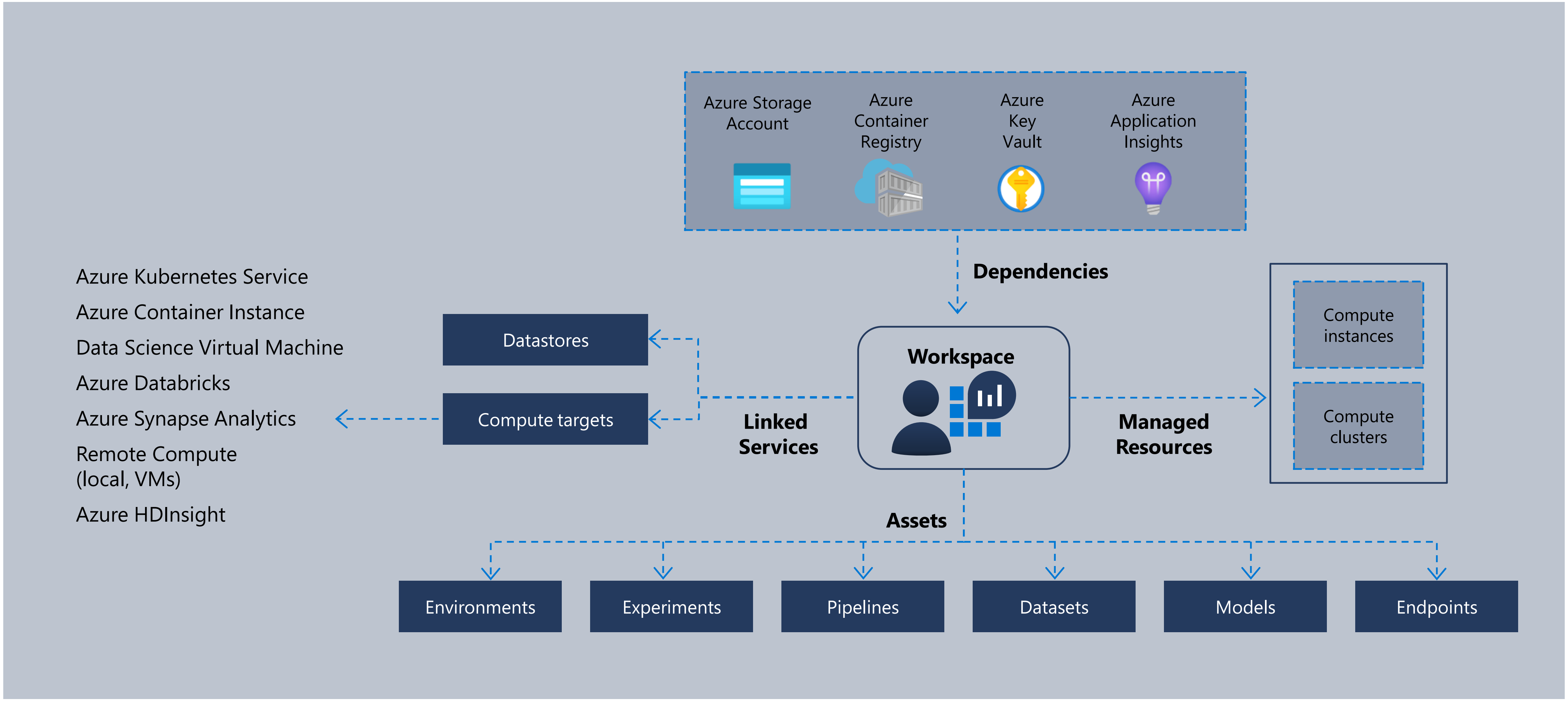
OpenAI to User: *"The fall arrest system requirements are as follows:*

1. *The system must be able to support the weight of the user and provide a comfortable fit.*
2. *The system must be easy to put on and take off.*
3. *The system must be adjustable to fit a range of body sizes.*
4. *The system must be comfortable to wear for extended periods of time.*
5. *The system must be durable and able to withstand repeated use."*

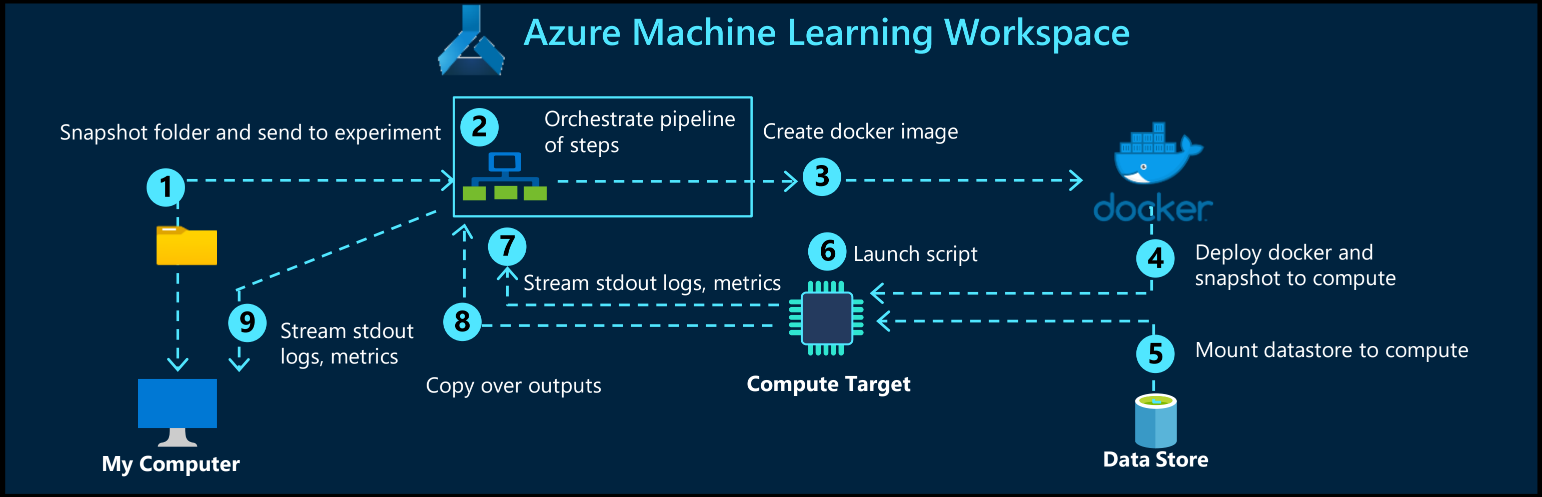
User to OpenAi: *Summarise the following in less than 20 words:*
Whilst completing wet mixing for heavy patching the sub-contract watercart, whilst being pushed by the Stabilizer impacted the rear of a hired spreader truck. The spreader truck was stationary at the time, in neutral with the park break on. The operator of the spreader truck was in the cab of the truck at the time of impact. After spreading lime on the patch the spreader truck drove approximately 22m from the end of the patch to park and await further gravel to be placed on the next patch (an unknown distance away at the time of reporting). Wet mixing of the patch commenced with the subcontract water cart "hooked up" to the mixer as usual using a push bar. During mixing the supervisor noted an excessive amount of lime extending from the front of the mixer (also known as a bow-wave) and instructed the mixer operator to continue mixing past the finish point of the patch to ensure the lime was correctly incorporated. This instruction is said to have been communicated to the mixer operator both verbally through the open window of the mixer and via UHF. The mixer extended the patch length by 3.5m to incorporate the excess lime. As the mixer continued mixing the watercart operator noticed that the front of the water cart was getting very close to the spreader and at approximately 2m from the spreader he called out on the radio to tell the mixer operator to stop. The water-cart operator noted that at the time of trying to communicate with the mixer operator that the mixer operator was talking on the UHF to a roller operator and was looking towards the back of the mixer. The watercart operator begin to wave his arm out of the window to get the mixer operator to notice. The water cart operator did not engage the foot brake (would have likely been ineffective in stopping the incident) and did not sound the horn. Other relevant factors include a severe storm was approaching the job site and all efforts where being made to get the patches completed to a water-safe standard.

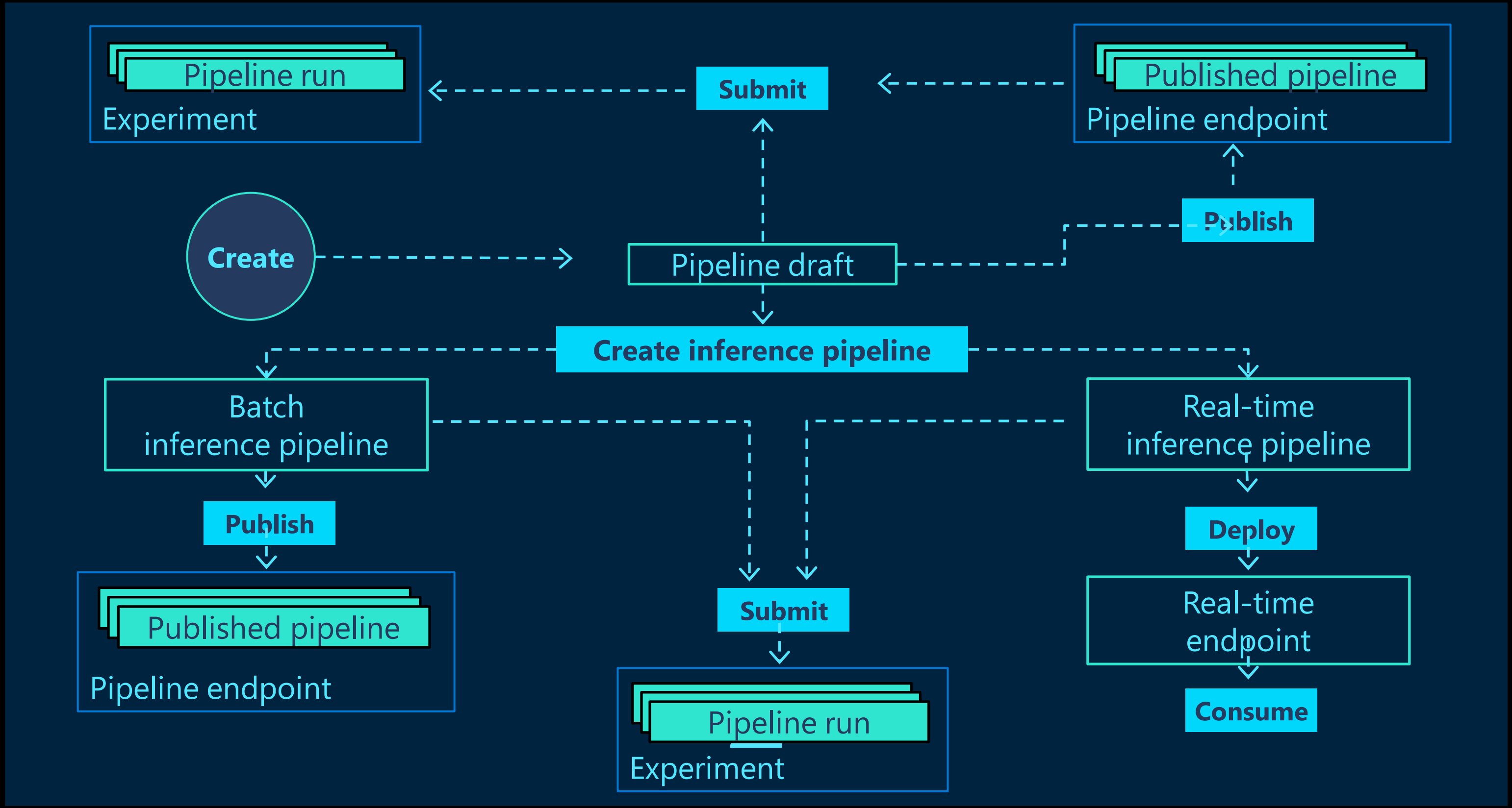
OpenAI to User: *"While wet mixing for heavy patching, the sub-contract watercart, pushed by the Stabilizer, impacted the rear of a stationary hired spreader truck. The spreader truck operator was in the cab at the time of impact."*

AzureML – Nuts and Bolts



When Runs/Pipelines are orchestrated





Lab 1 – Setup / Basics

- ml.azure.com
- Conda basics / virtual environments / yml
- AML Authentication
- Run using Environments / what curated environments are available
- Run Train-on-local Compute Instance
- Run Train-on-AML compute

- Get your Subscription ID, Resource Group, AML Workspace ready

BREAK

My first TM / NLP project - 2012

Introduction to the **tm** Package
Text Mining in R

Ingo Feinerer

October 18, 2022

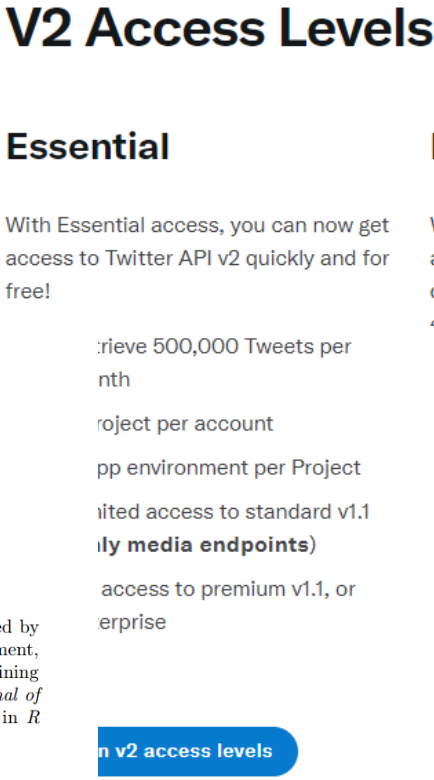
Introduction

This vignette gives a short introduction to text mining in R utilizing the text mining framework provided by the **tm** package. We present methods for data import, corpus handling, preprocessing, metadata management, and creation of term-document matrices. Our focus is on the main aspects of getting started with text mining in R—an in-depth description of the text mining infrastructure offered by **tm** was published in the *Journal of Statistical Software* (Feinerer et al., 2008). An introductory article on text mining in R was published in *R News* (Feinerer, 2008).

Data Import

The main structure for managing documents in **tm** is a so-called Corpus, representing a collection of text documents. A corpus is an abstract concept, and there can exist several implementations in parallel. The default implementation is the so-called VCorpus (short for *Volatile Corpus*) which realizes a semantics as known from most R objects: corpora are R objects held fully in memory. We denote this as volatile since once the R object is destroyed, the whole corpus is gone. Such a volatile corpus can be created via the constructor `VCorpus(x, readerControl)`. Another implementation is the PCorpus which implements a *Permanent Corpus* semantics, i.e., the documents are physically stored outside of R (e.g., in a database), corresponding R objects are basically only pointers to external structures, and changes to the underlying corpus are reflected to all R objects associated with it. Compared to the volatile corpus the corpus encapsulated by a permanent corpus object is not destroyed if the corresponding R object is released.

Within the corpus constructor, x must be a Source object which abstracts the input location. **tm** provides a set of predefined sources, e.g., DirSource, VectorSource, or DataframeSource, which handle a directory, a vector interpreting each component as document, or data frame like structures (like CSV files), respectively. Except



V2 Access Levels

Essential

With Essential access, you can now get access to Twitter API v2 quickly and for free!

trieve 500,000 Tweets per
nth

roject per account

pp environment per Project

ited access to standard v1.1

ly media endpoints)

access to premium v1.1, or
erprise

n v2 access levels

Elevated

With Elevated access, you can get free, additional access to endpoints and data, as well as additional App

Academic Research

If you qualify for our Academic Research access level, you can get access to even more data and

Common Text Mining / Neuro Linguistic Programming (NLP) Tasks

- Distributions – Term Frequency
- POS Tagging
- Categorization / Text Classification
- Named Entity Recognition
- Entity Linking / Knowledge Bases
- Sentence Segmentation
- Word Vectors and Semantic Similarity

Cambridge University Press 978-0-521-83657-9 - The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data
Ronen Feldman and James Sanger

Resources

- ***Deep Text: Using Text Analytics to Conquer Information Overload, Get Real Value from Social Media, and Add Bigger Text to Big Data*** by Tom Reamy
- ***Introduction to Information Retrieval* by Christopher** D. Manning, Prabhakar Raghavan and Hinrich Schütze,
- ***The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*** by Ronen Feldman and James Sanger
- ***Text Mining: Predictive Methods for Analyzing Unstructured Information*** by Sholom M. Weiss, Nitin Indurkhya, Tong Zhang and Fred J. Damerau
- ***Foundations of Statistical Natural Language Processing*** by Christopher D. Manning and Hinrich Schütze

LAB

- spaCy · Industrial-strength Natural Language Processing in Python
- Explore the spacy course
- Run spacy on a compute instance in AML
- Run spacy on a remote compute in AML

Encoders - BERT

- BERT Explained: State of the art language model for NLP | by Rani Horev | Towards Data Science

Spacy Cheat Sheet - datacamp

datacamp

Python For Data Science
spaCy Cheat Sheet

Learn spaCy online at www.DataCamp.com

spaCy

spaCy is a free, open-source library for advanced Natural Language processing (NLP) in Python. It's designed specifically for production use and helps you build applications that process and "understand" large volumes of text. Documentation: spacy.io

```
>>> $ pip install spacy
>>> import spacy
```

> Statistical models

Download statistical models

Predict part-of-speech tags, dependency labels, named entities and more. See here for available models: spacy.io/models

```
>>> $ python -m spacy download en_core_web_sm
```

Check that your installed models are up to date

```
>>> $ python -m spacy validate
```

Loading statistical models

```
>>> import spacy
>>> nlp = spacy.load("en_core_web_sm") # Load the installed model "en_core_web_sm"
```

> Documents and tokens

Processing text

> Spans

Accessing spans

Span indices are exclusive. So `doc[2:4]` is a span starting at token 2, up to – but not including! – token 4.

```
>>> doc = nlp("This is a text")
>>> span = doc[2:4]
>>> span.text
'a text'
```

Creating a span manually

```
>>> from spacy.tokens import Span #Import the Span object
>>> doc = nlp("I live in New York") #Create a Doc object
>>> span = Span(doc, 3, 5, label="GPE") #Span for "New York" with label GPE (geopolitical)
>>> span.text
'New York'
```

> Linguistic features

Attributes return label IDs. For string labels, use the attributes with an underscore. For example, `token.pos_` .

Part-of-speech tags
Predicted by Statistical model

```
>>> doc = nlp("This is a text.")
>>> [token.pos_ for token in doc] #Coarse-grained part-of-speech tags
['DET', 'VERB', 'DET', 'NOUN', 'PUNCT']
>>> [token.tag_ for token in doc] #Fine-grained part-of-speech tags
['DT', 'VBZ', 'DT', 'NN', '.']
```

Syntactic dependencies
Predicted by Statistical model

```
>>> doc = nlp("This is a text.")
>>> [token.dep_ for token in doc] #Dependency labels
['nsubj', 'ROOT', 'det', 'attr', 'punct']
>>> [token.head.text for token in doc] #Syntactic head token (governor)
['is', 'is', 'text', 'is', 'is']
```

Named entities
Predicted by Statistical model

```
>>> doc = nlp("Larry Page founded Google")
>>> [(ent.text, ent.label_) for ent in doc.ents] #Text and label of named entity span
[('Larry Page', 'PERSON'), ('Google', 'ORG')]
```

> Pipeline components

Functions that take a Doc object, modify it and return it.

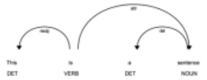
> Visualizing

If you're in a Jupyter notebook, use `displacy.render` otherwise, use `displacy.serve` to start a web server and show the visualization in your browser.

```
>>> from spacy import displacy
```

Visualize dependencies

```
>>> doc = nlp("This is a sentence")
>>> displacy.render(doc, style="dep")
```



Visualize named entities

```
>>> doc = nlp("Larry Page founded Google")
>>> displacy.render(doc, style="ent")
```

Larry Page PERSON founded Google ORG

> Word vectors and similarity

To use word vectors, you need to install the larger models ending in `md` or `lg` , for example `en_core_web_lg` .

Comparing similarity

```
>>> doc1 = nlp("I like cats")
>>> doc2 = nlp("I like dogs")
>>> doc1.similarity(doc2) #Compare 2 documents
>>> doc1[2].similarity(doc2[2]) #Compare 2 tokens
>>> doc1[0].similarity(doc2[1:3]) # Comparetokens and spans
```

Accessing word vectors

```
>>> doc = nlp("I like cats") #Vector as a numpy array
>>> doc[2].vector #The L2 norm of the token's vector
>>> doc[2].vector_norm
```

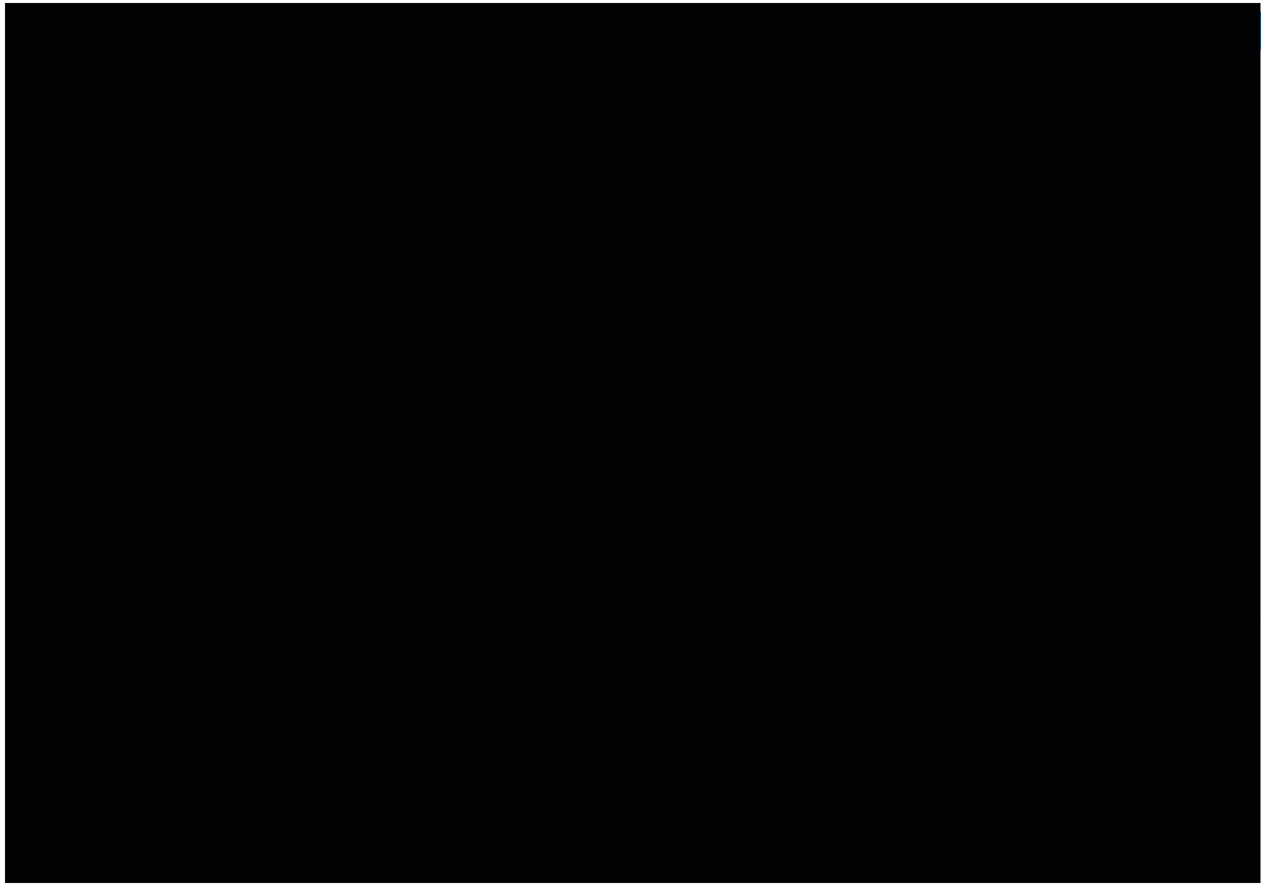
> Syntax iterators

Sentences
Ususally needs the dependency parser

BREAK

Text & Computer vision Data Labeling

- Azure Machine Learning data labeling is a central place to create, manage, and monitor labeling projects:
- Coordinate data, labels, and team members to efficiently manage labeling tasks.
- Track progress and maintains the queue of incomplete labeling tasks.
- Start and stop the project and control the labeling progress.
- Review the labeled data and export labeled in COCO format or as an Azure Machine Learning dataset.



LAB

- Find a Text Document – csv/txt format – each row is a response
  - Use a small one to highlight ML-assisted labelling capabilities
- Use YELP review on Kaggle if can’t be found
- Upload onto AzureML Datasets
- Create a AzureML Data labelling project

AutoML NLP

- Support for natural language processing (NLP) tasks in automated ML allows you to easily generate models trained on text data for text classification and named entity recognition scenarios. Authoring automated ML trained NLP models is supported via the Azure Machine Learning Python SDK. The resulting experimentation jobs, models, and outputs can be accessed from the Azure Machine Learning studio UI.

- The NLP capability supports:
  - End-to-end deep neural network NLP training with the latest pre-trained BERT models
  - Seamless integration with Azure Machine Learning data labeling
  - Use labeled data for generating NLP models
  - Multi-lingual support with 104 languages
  - Distributed training with Horovod

Supported Model Algorithms

- bert_base_cased
- bert_large_uncased
- bert_base_multilingual_cased
- bert_base_german_cased
- bert_large_cased
- distilbert_base_cased
- distilbert_base_uncased
- roberta_base
- roberta_large
- distilroberta_base
- xlm_roberta_base
- xlm_roberta_large
- xlnet_base_cased
- xlnet_large_cased

Supported Hyperparameters

Parameter name	Description	Syntax
gradient_accumulation_steps	The number of backward operations whose gradients are to be summed up before performing one step of gradient descent by calling the optimizer's step function. This is leveraged to use an effective batch size which is gradient_accumulation_steps times larger than the maximum size that fits the GPU.	Must be a positive integer.
learning_rate	Initial learning rate.	Must be a float in the range (0, 1).
learning_rate_scheduler	Type of learning rate scheduler.	Must choose from `linear, cosine, cosine_with_restarts, polynomial, constant, constant_with_warmup`.
model_name	Name of one of the supported models.	Must choose from `bert_base_cased, bert_base_uncased, bert_base_multilingual_cased, bert_base_german_cased, bert_large_cased, bert_large_uncased, distilbert_base_cased, distilbert_base_uncased, roberta_base, roberta_large, distilroberta_base, xlm_roberta_base, xlm_roberta_large, xlnet_base_cased, xlnet_large_cased`.
number_of_epochs	Number of training epochs.	Must be a positive integer.
training_batch_size	Training batch size.	Must be a positive integer.
validation_batch_size	Validation batch size.	Must be a positive integer.
warmup_ratio	Ratio of total training steps used for a linear warmup from 0 to learning_rate.	Must be a float in the range [0, 1].

- [Set up AutoML for NLP - Azure Machine Learning | Microsoft Learn]()

LAB

- Execute AutoML NLP on AzureML

- Consider
  - Text Classification (Single Label)
  - Text Classification (Multiple Labels)
  - Text Named Entity Recognition (NER)

BREAK

Hugging Face + OpenAI – Feature Presentation (no labs)

Hugging Face - We are on a mission to democratize *good* machine learning, one commit at a time.

Models, Datasets – all hosted on AzureML Endpoints

OpenAI - Their mission is to ensure that artificial general intelligence (AGI) benefits humanity

OpenAI models hosted on Azure (currently in gated public preview)

Generative Pre-trained Transformer 3 (GPT-3) – OpenAI

- 170B Parameter autoregressive language model that uses deep learning to produce human-like text.
- Pre-trained on a large text corpus of internet + books
- Works by predicting the most likely next word based on input text
- Text-in/text-out interface
- Has unique capabilities with few-shot learning
- Models can be fine-tuned for specific tasks

Prompt:

Negate the following sentence.

The price of bubblegum decreased on Tuesday.

Response:

The price of bubblegum increased on Tuesday.

Q&A