Quickly Build Analyst NLP Capability in Your Organization

MAJ John Scudder

AORS 2023

Motivation

- NLP permeates our daily lives

- How can you quickly learn and leverage NLP for your own use cases?

- **<u>My Goal</u>: Equip you with a framework and resources to bring NLP capability to analysts in your organization.**

Agenda

- Background
- Methodology: Environment / Audience Considerations
- Lesson Plan
- Best Practices
- Resources for YOU!
  - Curated opensource resources
  - Code repository with Jupyter Notebooks and slides

Background

UNITED STATES MILITARY ACADEMY WEST POINT

Background

- <u>Problem set</u> within USMA D/Math:
  - Existing research projects have a requirement to analyze unstructured text data → NLP knowledge
  - All D/Math Faculty have experience with w/ structured data
  - ~5 members w/ NLP experience out of > 90 faculty

- <u>Solution</u>: Combine local experience & offer 5x interactive lectures w/ runnable code

- I'm providing you the product we used → tailor it for your purposes!

Methodology

UNITED STATES MILITARY ACADEMY
WEST POINT

Environment

- Requirements
  - Collaborative coding environment
  - Common Python packages installed
  - Classroom for instruction
- Possible Solutions
  - CoCalc
  - Google Colab
  - Institutional Compute Cluster
  - Cloud-based (e.g., AWS, Azure)

Audience

- D/Math Population
  - Experienced Analysts / Researchers
  - Range of Python skills

- Method of Instruction
  - Demos and slides
  - Executable code to discuss
  - Jupyter Notebooks
  - Common dataset to our organization:
    course-end feedback

Lesson Plan

5x Lesson Agenda

Lesson Plan

1. NLP Tasks & Terminology; Preprocessing
2. Basic Lexical Analysis & Word Representations
3. Word Embeddings & Model Training
4. SOTA models & Python Packages (Hugging Face)
5. Additional packages & Ongoing Research in Department

UNITED STATES MILITARY ACADEMY
WEST POINT

Lesson 1: NLP Tasks & Terminology; Preprocessing

MATHEMATICAL SCIENCES

Common NLP Tasks

Syntax

Morphology

Word Segmentation

Part-of-Speech Tagging

Parsing

- Constituency
- Dependency

Discourse

Summarization

Coreference Resolution

Semantics

Sentiment Analysis

Topic Modelling

Named Entity Recognition (NER)

Relation Extraction

Word Sense Disambiguation

Natural Language Understanding (NLU)

Natural Language Generation (NLG)

Machine Translation

Entailment

Question Answering

Language Modelling

Common NLP Tasks

Syntax

Morphology

Word Segmentation

Part-of-Speech Tagging

Parsing

- Constituency
- Dependency

Discourse

Summarization

Coreference Resolution

Semantics

Sentiment Analysis

Topic Mode

- Really slow wait. Took forever to get food.
- Freshest ingredients ever. New favorite restaurant. Will be back!
- Found a hair in the food. Horrible.
- Waited 6 months to get a reservation at this place. Totally worth it.

Question Answering

Language Modelling

Tokenization

"I particularly thought getting up to the boards was helpful."

→ `[I, particularly, thought, getting, up, to, ...]`

```python
# tokenize our corpus
tokens = helpful.apply(lambda x: nltk.word_tokenize(str(x)))
print("NLTK Tokenization: \n", tokens.head())
tokens = helpful.apply(lambda x: TextBlob(str(x)).words)
print("TextBlob Tokenization:\n",tokens.head())
```

Python

```
NLTK Tokenization:
 0    [!, !, !, 12354, In, class, examples, were, gr...
1    [I, particularly, thought, getting, up, to, th...
2    [every, time, we, would, go, to, boards, we, w...
3                   [humor, that, motivated, learning]
4    [During, almost, every, class, period, ,, we, ...
Name: text, dtype: object
TextBlob Tokenization:
 0    [12354, In, class, examples, were, grate, for,...
1    [I, particularly, thought, getting, up, to, th...
2    [every, time, we, would, go, to, boards, we, w...
3                   [humor, that, motivated, learning]
4    [During, almost, every, class, period, we, got...
Name: text, dtype: object
```

Lesson 2:
Basic Lexical Analysis & Word Representations

Basic Sentiment Analysis

"Mr. XXX is the best teacher in this course."

→ {'neg': 0.0, 'neu': 0.656, 'pos': 0.344, 'compound': 0.6369}

"Mr. XXX is not the best teacher in this course."

→ {'neg': 0.272, 'neu': 0.728, 'pos': 0.0, 'compound': -0.5216}

```python
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer #Valence Aware Dictionary for sEnt
analyzer = SentimentIntensityAnalyzer()
for sentence in blob.sentences:
    print(sentence,analyzer.polarity_scores(sentence))
```

Python

```
Mr. XXX is the best teacher in this course. {'neg': 0.0, 'neu': 0.656, 'pos': 0.344, 'compound': 0.6369}
Mr. XXX is not the best teacher in this course. {'neg': 0.272, 'neu': 0.728, 'pos': 0.0, 'compound': -0.5216}
```

Simplistic Summarization: Keywords

What do students think the most helpful classroom experiences include?

```python
import random
nouns = list()
blob = TextBlob(allResponses)
for word, tag in blob.tags:
    if tag == 'NN': # Try noun? NN
        nouns.append(word.lemmatize())

print("Students think that helpful classroom experien
for item in random.sample(nouns,10):
    print(item)
```

```
Students think that helpful classroom experiences include:
room
use
instructor
assignment
test
board
```

Lesson 3:
Word Embeddings & Model Training

Task: find similar comments based on cosine similarity of word2vec embeddings

```
Input: doing practice problems and having class examples were very helpful
```

Lesson 3: Word Embeddings & Model Training

Input: doing practice problems and having class examples were very helpful

****** word2vec Recommendations ******

- the various class practice problems were especially helpful
- doing problems in class was very helpful for my learning
- this path course offered a lot of opportunities to go to the boards during class and practice class examples this was extremely helpful to not only practice problems but to also have the instructor critique our work
- doing board problems were very helpful
- working through problems as a class was very helpful to me

Lesson 4:
SOTA Models &
Python Packages

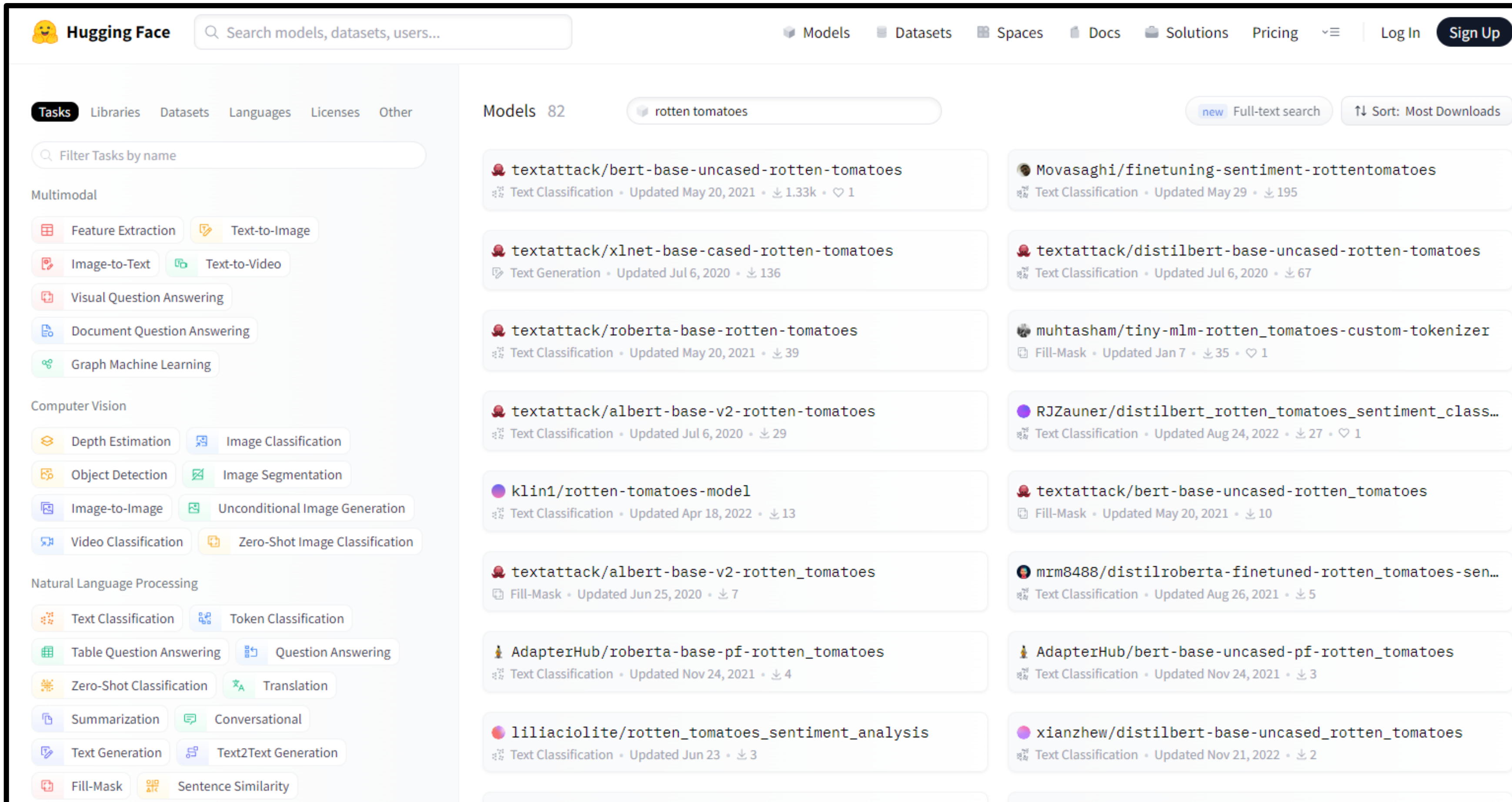
Sentiment Analysis with a Pre-Trained LLM (SOTA)

```python
MODEL_NAME = f"cardiffnlp/twitter-roberta-base-sentiment-latest"
device = torch.device(0 if torch.cuda.is_available() else -1)

classifier = pipeline("sentiment-analysis",model = MODEL_NAME, device = device) # device should make

posCom = "MA104 is my favorite class -- I especially enjoy the self-study days."
negCom = "This was the worst class I've ever had. The work was tedious and not related to my major."
print()
print("Comment: ", posCom,"\n",classifier(posCom))
print("Comment: ", negCom,"\n",classifier(negCom))
```

```
Comment:  MA104 is my favorite class -- I especially enjoy the self-study days.
 [{'label': 'positive', 'score': 0.9886686205863953}]
Comment:  This was the worst class I've ever had. The work was tedious and not related to my major.
 [{'label': 'negative', 'score': 0.9576689004898071}]
```

Lesson 4: SOTA Models & Hugging Face Tools

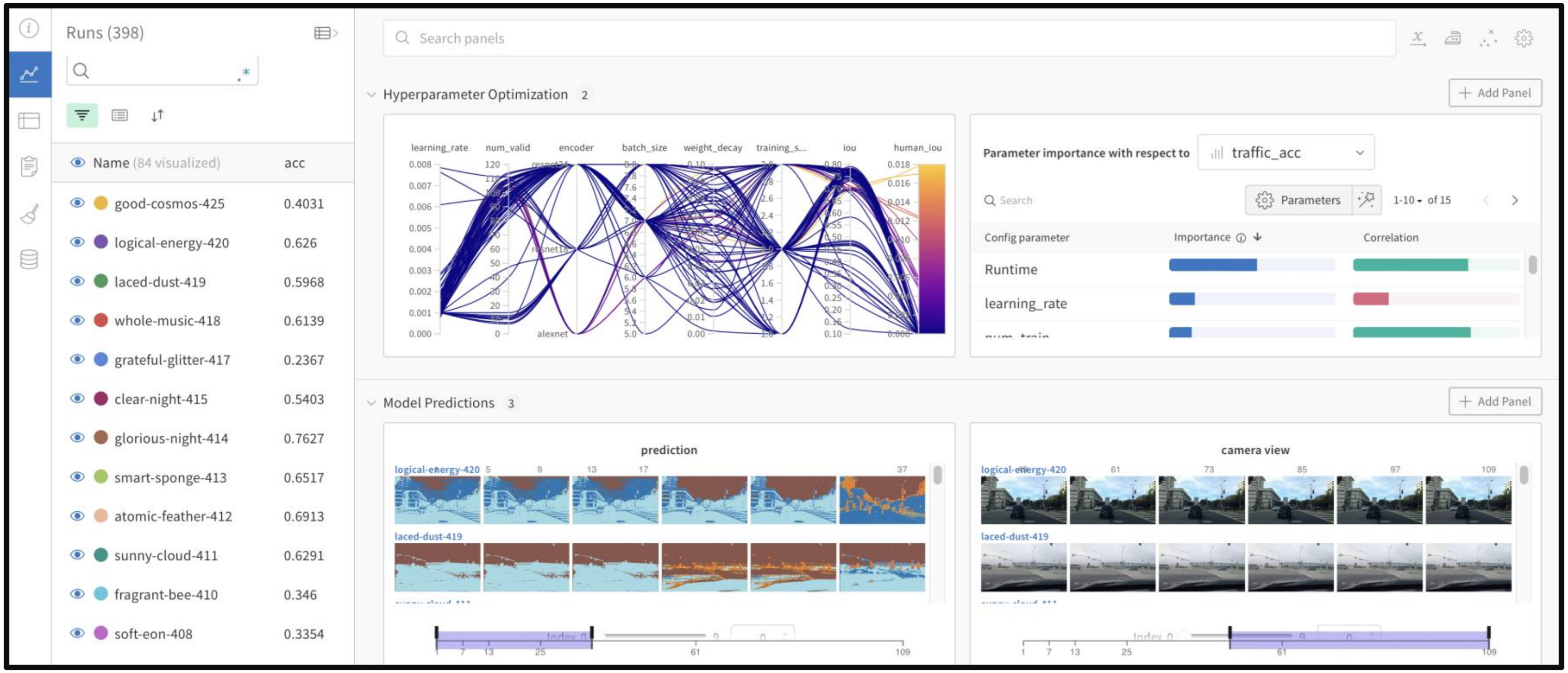
Where do I get SOTA models? Benchmark datasets?



Model Hub: https://huggingface.co/models

Lesson 5:
Additional Packages & Ongoing Research in Department

Why use Weights & Biases? It automates your record keeping!



[W&B Docs | Weights & Biases Documentation (wandb.ai)](https://docs.wandb.ai)

[Intro_to_Weights_&_Biases.ipynb - Colaboratory (google.com)](https://colab.research.google.com)

Why use Label Studio? **It's an intuitive environment to label data!**

[Open Source Data Labeling | Label Studio]()

Best Practices

Best Practices

- Use a dataset familiar for your faculty
- Consider audience coding skills and desired coding environment
- Include orientation to using Graphics Processing Units (GPUs) for large language models (LLM)
- Provide executable code – live coding is slow

What are **<u>your</u>** next steps?

UNITED STATES MILITARY ACADEMY
WEST POINT®

Resources for YOU

MATHEMATICAL SCIENCES

Curated Open-Source Resources

- **The Hugging Face community** (particularly their Transformers library) is one of the most popular communities/platforms for NLP/AI/ML. They have great tutorials and readable documentation. Hugging Face models and datasets are designed to work well with and/or build on other popular NLP packages [nltk, AllenNLP, spacy].

- **CS287: Deep Learning for NLP** is a course I took in grad school. It was only taught for one semester, but the lecture slides are all available online (for now) [link - scroll to "Lectures"]. There's also a Supplemental Resources page for NLP.

- If you want straightforward explanations of how the most used NLP models work, Jay Alammar has a blog with great visuals and explanations for word2vec (2013) and BERT (2018+) toward the bottom of his blog posts.

- Sebastian Ruder [his site] [his Google Scholar] has a great blog that covers new ideas in NLP, NLP Progress, his highlights from NLP conferences…etc. In 2018 he gave a brief timeline of NLP. https://ruder.io/a-review-of-the-recent-history-of-nlp/.

UNITED STATES MILITARY ACADEMY
WEST POINT

Questions?

Scan QR code for GitHub repository access!

Email: john.scudder@westpoint.edu

MATHEMATICAL SCIENCES

Backup Slides

5x Lesson Agenda

- 15 March @ 1500 in TH120
  - Preprocessing, Tokenization, Part of Speech Tagging
- 22 March @ 1000 in TH120
  - Basic lexical analysis – sentiment, cosine similarity
  - Representations – N-Grams, Bag of Words
- 5 April @ 1100 in TH120
  - TF-IDF & word2vec
  - Visualizations
  - Training models
- 19 April @ 1000 in TH120
  - State-of-the-art models: transformers, BERT
  - Hugging Face packages
- 9 May @ 1000 in TH120
  - 2x packages: Weights & Biases + Label Studio
  - Hone your craft! Ongoing D/Math research with NLP

UNITED STATES MILITARY ACADEMY
WEST POINT®

LESSON 1

MATHEMATICAL SCIENCES

Common Tasks

Common Tasks (aka problems)

Syntax

Morphology

Word Segmentation

Part-of-Speech Tagging

Parsing

- Constituency
- Dependency

Discourse

Summarization

Coreference Resolution

Semantics

Sentiment Analysis

Topic Modelling

Named Entity Recognition (NER)

Relation Extraction

Word Sense Disambiguation

Natural Language Understanding (NLU)

Natural Language Generation (NLG)

Machine Translation

Entailment

Question Answering

Language Modelling

Common Tasks (aka problems)

Syntax

Morphology

Word Segmentation

Part-of-Speech Tagging

Parsing

- Constituency
- Dependency

Discourse

Summarization

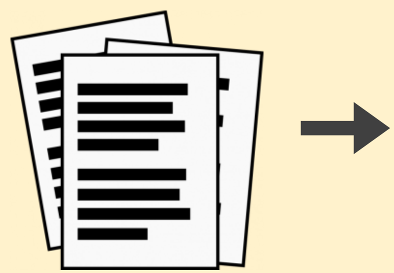
Coreference Resolution

Semantics

Sentiment Analysis

Topic Modelling

Language Modelling



“The U.S. secretly modified the Himars rocket launchers it gave Ukraine to keep Kyiv from firing missiles into Russia.”

Adapted from Chris Tanner's CS287 Lecture Slides.

Common Tasks (aka problems)

Syntax

Morphology

Word Segmentation

Part-of-Speech Tagging

Parsing

- Constituency
- Dependency

Discourse

Summarization

Coreference Resolution

Semantics

Sentiment Analysis

Topic Mo ing

LU)

Question Answering

Language Modelling

- Really slow wait. Took forever to get food.
- Freshest ingredients ever. New favorite restaurant. Will be back!
- Found a hair in the food. Horrible.
- Waited 6 months to get a reservation at this place. Totally worth it.

Common Tasks (aka problems)

Syntax

Morphology

Word Segmentation

Part-of-Speech Tagging

Parsing

- Constituency
- Dependency

Discourse

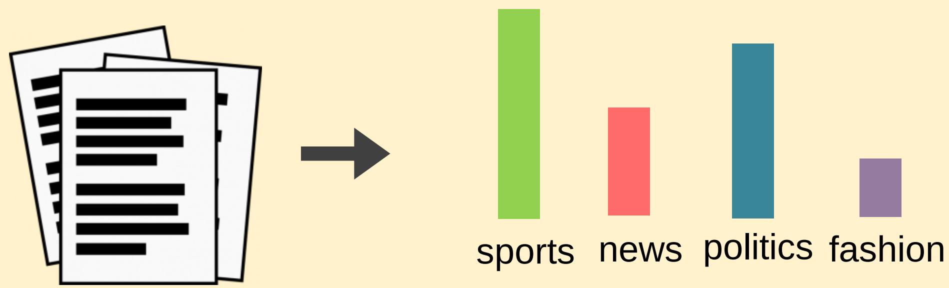
Summarization

Coreference Resolution

Semantics

Sentiment Analysis

Topic Modelling



Named E… Recognition (NER)

…LU)

Language Modelling

Common Tasks (aka problems)

Syntax

Morphology

Word Segme

Part-of-Spee

Parsing

Constituency

Dependency

“Alexa, play White Christmas by Bing Crosby”

“Alexa, play White Christmas by Bing Crosby”

INTENT SONG ARTIST

on (NER)

ation

Natural Language Understanding (NLU)

Natural Language Generation (NLG)

Machine Translation

Entailment

Question Answering

Language Modelling

Discourse

Summarization

Coreference Resolution

Common Tasks (aka problems)

Syntax

Morphology

Word Segmentation

Part-of-Speech Tagging

Parsing

- Constituency
- Dependency

Discourse

Summarization

Coreference Resolution

Semantics

Sentiment Analysis

Topic Modelling

Named Entity Recognition (NER)

…on

…anding (NLU)

…ion (NLG)

Machine Translation

Entailment

Question Answering

Language Modelling

Dzień dobry → Good morning

POLISH → ENGLISH

Hands-on Coding

LESSON 2

Terminology

Representations

How do we get *any* system to process, "understand", leverage language?

- **Representation**: how we transform symbolic meaning (e.g., words, signs, braille, speech audio) into something the computer can use
- **Modelling**: given these represented symbols, how we use them to model the task at hand

Representations

Representing images is relatively easy. There is a meaningful relationship between the byte values and color.

170 33 71

r g b

255 33 71

r g b

Representations

In contrast – words are represented by strings and there are no meaningful relationships between the byte values and language.

h	a	t	e
68	61	74	65

a	t	e
61	74	65

hate and ate. No relation but similar byte values.

H	a	t
48	61	74

h	a	t
68	61	74

Hat and hat. Identical concept but different byte values.

A Language Model estimates the probability of any sequence of words

Let $\boldsymbol{X}$ = "Jack was late for class"
$w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5$

$\mathrm{P}(\boldsymbol{X})$ = $P($"Jack was late for class"$)$ = $2.3\ x\ 10^{-14}$

Scenario: assume we have a finite vocabulary V

V^* represents the **infinite set** of strings/sentences that we could construct

e.g., V^*= {a, a dog, a frog, dog a, dog dog, frog dog, frog a dog, …}

Data: we have a training set of sentences $x \in V^*$

Problem: estimate a probability distribution:

$$\sum_{x \in V^*} p(x) = 1$$

$$p(the) = 10^{-2}$$

$$p(the, sun, okay) = 2.5x10^{-13}$$

$$p(waterfall, the, icecream) = 3.2x10^{-18}$$

Token vs. Type

A word **token** is a specific occurrence of a word in a text

A word **type** refers to the general form of the word, defined by its lexical representation

If our corpus were just "I ran and ran and ran", you'd say we have:

- 6x word **tokens** [I, ran , and , ran , and , ran]
- 3x word **types**: {I, ran, and}

Bigram Model Example – condition each word on its immediate predecessor

Let $\boldsymbol{X}$ = "Jack was late for class"
$w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5$

$$P(w'|w) = P(\text{"w,w'"}) = \frac{n_{w,w'}(\boldsymbol{d})}{n_{w,w*}(\boldsymbol{d})}$$

$$P(\text{class}|\text{for}) = P(\text{for, class}) = \frac{12}{3{,}000}$$

Let's say our corpus $\boldsymbol{d}$ has 100,000 words

word	# occurrences
Jack	15
was	1,000
late	400
for	3,000
class	350

$$n_{w_*}(\boldsymbol{d}) = 100{,}000$$

$n_{w,w'}(\boldsymbol{d})$ = # of times words $\boldsymbol{w}$ and $\boldsymbol{w'}$ appear together as a bigram in $\boldsymbol{d}$

$n_{w,w*}(\boldsymbol{d})$ = # of times word $\boldsymbol{w}$ is the first token of a bigram in $\boldsymbol{d}$

Adapted from Chris Tanner's CS287 Lecture Slides.

Bag of Words

Simple idea: let's represent each document as a feature vector, which can serve as the input to any of your favorite supervised ML models

Let's say our dataset's entire *vocabulary* is just 10 words. Each unique word can have its own dimension (feature index). Each document's vector has a 1 if the word is present. Otherwise, 0.

e.g., "the dog jumped" is represented as

dog	the	quick	went	brown	a	jumped	fast	over	store
1	1	0	0	0	0	1	0	0	0

Adapted from Chris Tanner's CS287 Lecture Slides.

Bag of Words

Imagine a document is a sports broadcast transcript, which concerns a few teams but mostly discusses the local home team, the Cubs. We have no indication of *how much* the document is about the Cubs.

baseball	chicago	cubs	the	wringley	padres	shohei	mvp	homerun	crowd
1	1	1	1	0	1	0	0	0	1

Using a count-based approach, now we can see that it's much more about the **Chicago Cubs** than the **Padres**.

baseball	chicago	cubs	the	wringley	padres	shohei	mvp	homerun	crowd
2	9	17	8	0	2	0	0	0	2

Adapted from Chris Tanner's CS287 Lecture Slides.

Bag of Words

Weaknesses:

- Flattened view of the document
- Context-insensitive ("the horse ate" = "ate the horse")
- Curse of Dimensionality (vocab could be over 100k)
- Orthogonality: no concept of semantic similarity at the word-level
  - e.g., $d(\text{dog, cat}) = d(\text{dog, chair})$

Hands-on Coding

LESSON 3

TF-IDF

TF (term frequency) = f_{w_i} = # times word w_i appeared in the document

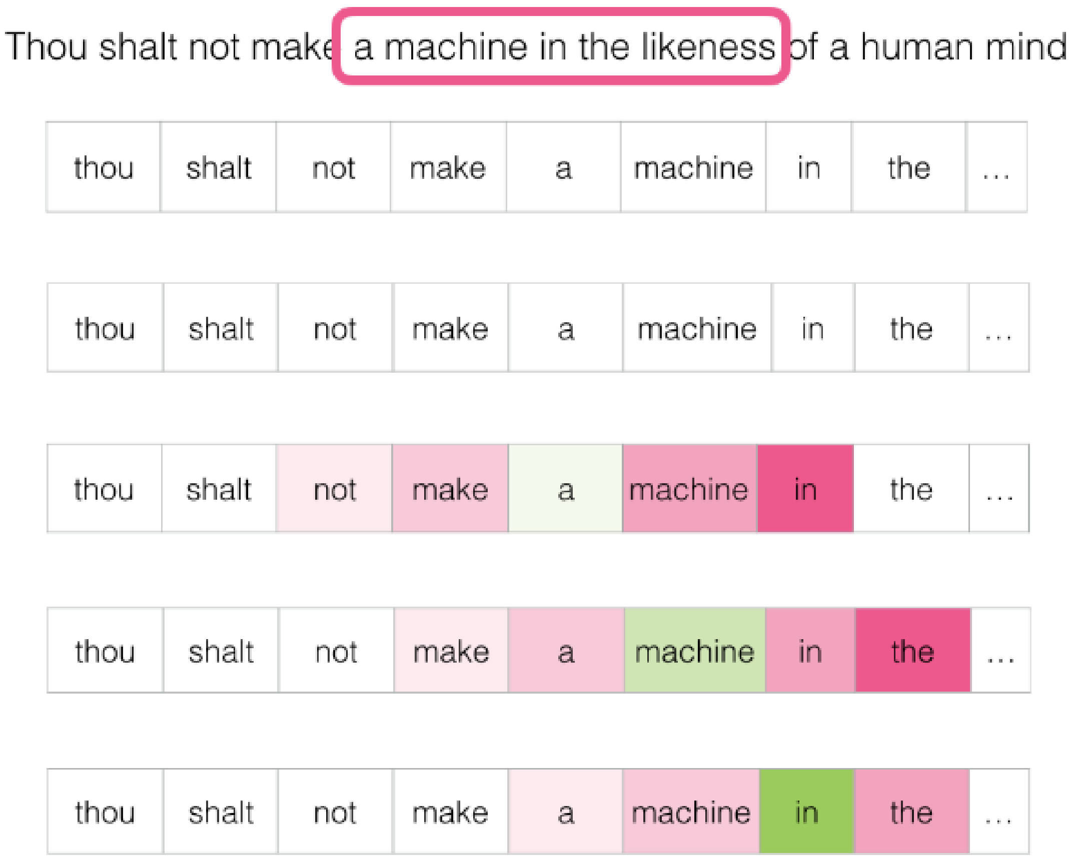
IDF (inverse document frequency) = $log\left(\frac{\text{\# docs in corpus}}{\text{\# docs containing } w_i}\right)$

$$\textbf{TFIDF} = f_{w_i} * log\left(\frac{\text{\# docs in corpus}}{\text{\# docs containing } w_i}\right)$$

- TF-IDF rewards “rare” words that frequently appear in a few documents. Words that appear in every document in the corpus have a TF-IDF score = 0.

- Hans Peter Luhn (1957) credited with TF; Karen Spärck Jones (1972) credited with IDF.

Word2Vec



input word	target word
not	thou
not	shalt
not	make
not	a
make	shalt
make	not
make	a
make	machine
a	not
a	make
a	machine
a	in
machine	make
machine	a
machine	in
machine	the
in	a
in	machine
in	the
in	likeness

Graphic:
https://jalammar.github.io/illustrated-word2vec/

Original Papers:
https://arxiv.org/abs/1301.3781
https://arxiv.org/abs/1309.4168

Word2Vec

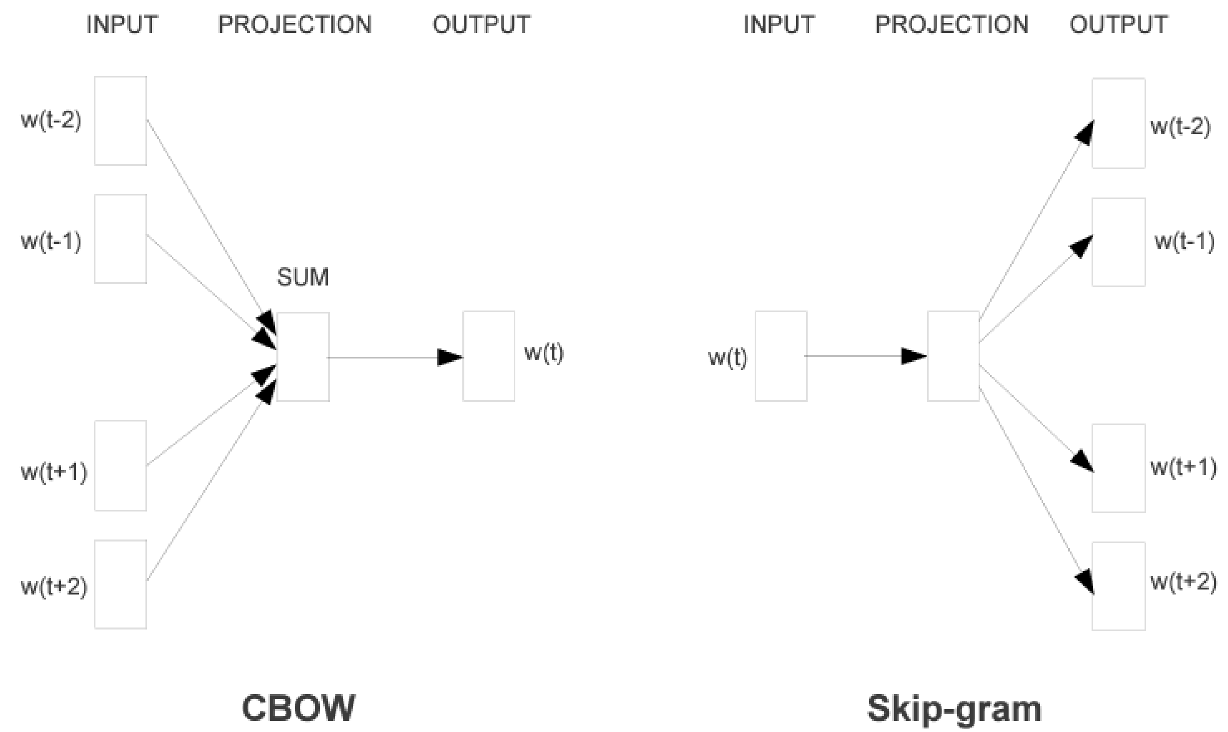
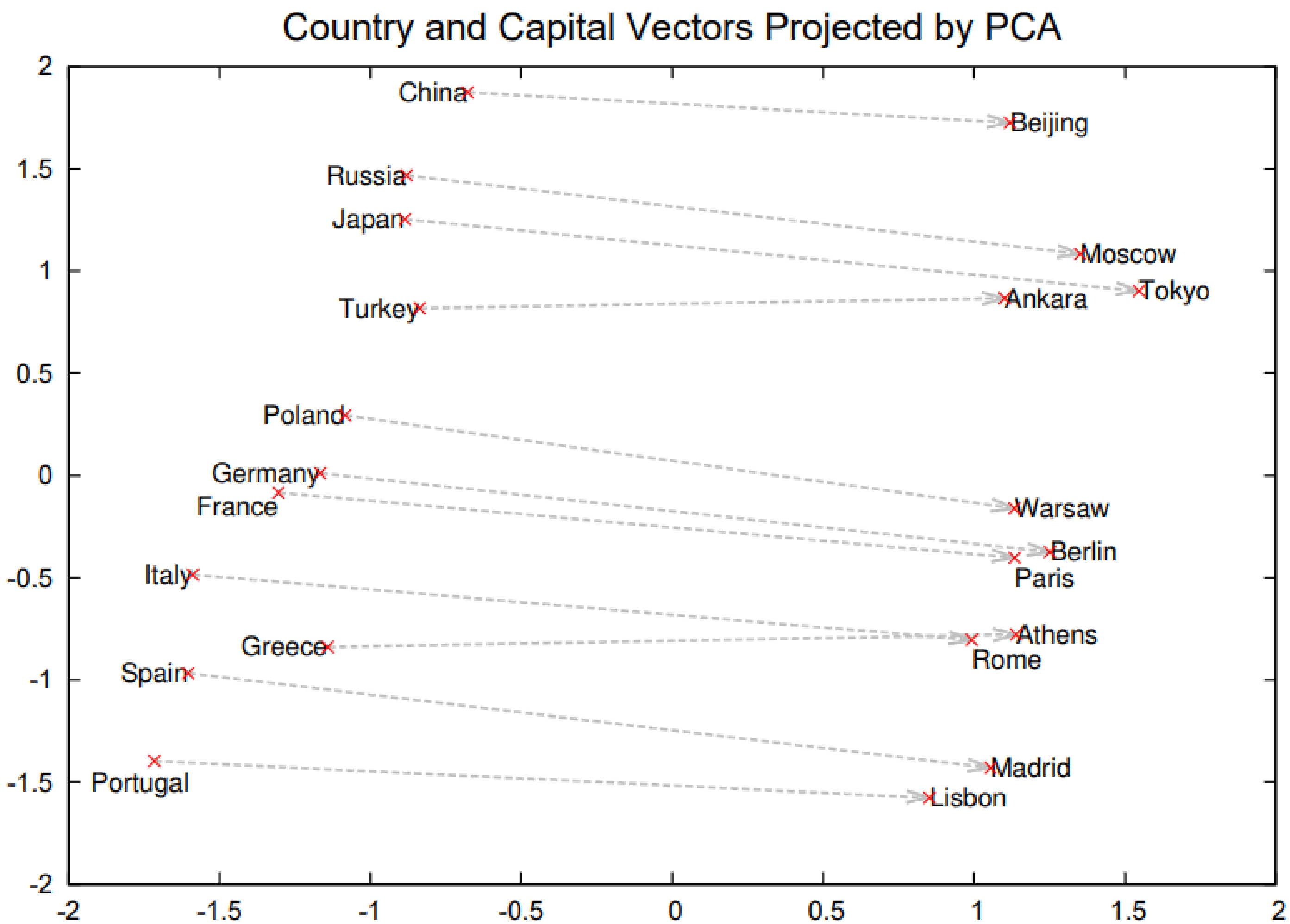


Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

Original Papers:
https://arxiv.org/abs/1301.3781
https://arxiv.org/abs/1309.4168

Word2Vec



Original Papers:
https://arxiv.org/abs/1301.3781
https://arxiv.org/abs/1309.4168
https://arxiv.org/abs/1310.4546

Word2Vec

Document 1	Document Vector Embedding --->	0.4, -0.0375, 0.725, ...
Word Embedding	the	0.4, 0.25, 0.1, ...
Word Embedding	board	0.9, -0.42, 0.01, ...
Word Embedding	was	0.6, 0.34, -0.21, ...
Word Embedding	great	-.3, -0.32, 3, ...

Document 2	Document Vector Embedding --->	0.285, 0.23, -0.34, ...
Word Embedding	the	0.4, 0.25, 0.1, ...
Word Embedding	elephant	0.34, 0.23, -0.3, ...
Word Embedding	was	0.6, 0.34, -0.21, ...
Word Embedding	big	-0.2, 0.1, 0.07, ...

Word2Vec

```python
print("Length of the vector: ",len(wv['board']), "\n Embedding for BOARD: \n",wv['board'])
```

```
Length of the vector:  300
 Embedding for BOARD: 
 [-0.14453125 -0.25976562 -0.01611328 -0.01074219 -0.01281738 -0.34765625
  0.10839844  0.00340271  0.07080078  0.04199219  0.0456543  -0.14160156
 -0.03808594 -0.19335938 -0.30273438  0.09619141  0.0703125  -0.11425781
 -0.02709961  0.01306152 -0.09863281  0.22070312  0.00118256  0.1328125
  0.02783203  0.14453125 -0.21386719  0.30664062 -0.20117188 -0.29101562
  0.07080078 -0.07861328 -0.07958984 -0.06738281  0.17675781 -0.23730469
  0.171875    0.31445312  0.13378906 -0.12109375 -0.09423828  0.13671875
  0.0390625  -0.09619141  0.07666016 -0.12695312  0.19140625 -0.04907227
  0.04589844  0.21679688 -0.00778198  0.08886719  0.05566406 -0.0859375
 -0.08349609  0.0559082  -0.17382812  0.04443359  0.10644531  0.00653076
  0.09863281  0.10205078 -0.15429688  0.09130859  0.06933594  0.06030273
  0.00701904  0.19433594  0.140625    0.19238281  0.01043701 -0.140625
  0.33984375  0.09326172  0.00125122  0.19628906  0.03613281 -0.09277344
  0.25        0.08496094  0.11083984 -0.05419922  0.09912109  0.09716797
  0.28710938 -0.11914062 -0.1640625  -0.05810547  0.15820312  0.11816406
  0.30078125 -0.00357056  0.02087402  0.20703125  0.0378418  -0.07324219
 -0.24707031  0.22558594 -0.06201172  0.12011719  0.0612793   0.00628662
  0.0267334   0.22949219 -0.06494141 -0.09521484 -0.02026367 -0.171875
  0.00494385 -0.2734375   0.0300293  -0.18554688  0.04321289 -0.22070312
  0.3203125  -0.10986328  0.12011719  0.07666016  0.22070312 -0.16503906
 -0.06176758 -0.11962891 -0.15039062  0.0859375  -0.13769531 -0.11962891
 -0.25976562 -0.08496094  0.05151367  0.04101562  0.06591797 -0.04907227
 -0.18457031 -0.11865234  0.02941895  0.26171875  0.04516602  0.11425781
  0.06640625  0.14160156 -0.12597656 -0.04467773  0.06005859 -0.08789062
  0.04150391 -0.02868652 -0.14160156 -0.0378418   0.34570312  0.17382812
 -0.08154297  0.05883789 -0.23632812  0.24511719 -0.06542969  0.0144043
  0.12988281  0.08105469  0.20019531  0.14941406 -0.0559082   0.00105286
  0.16308594  0.03198242  0.03173828 -0.18847656 -0.02893066  0.02001953
  0.26171875  0.12792969  0.10400391 -0.00041008  0.00915527 -0.13085938
```

UNITED STATES MILITARY ACADEMY
WEST POINT®

Hands-on Coding

MATHEMATICAL SCIENCES

LESSON 4

[Link to Behemoth Google Colab Notebook!]

Separate Notebooks for Exploring vs. Training a Model

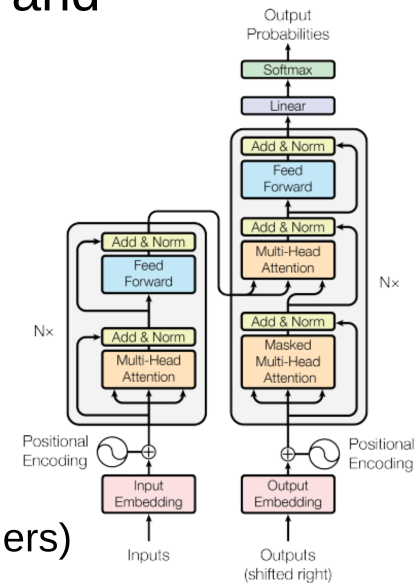
- Part 1: Explore Hugging Face Datasets and Models
- Part 2: Train a Model

Transformers
…and BERT

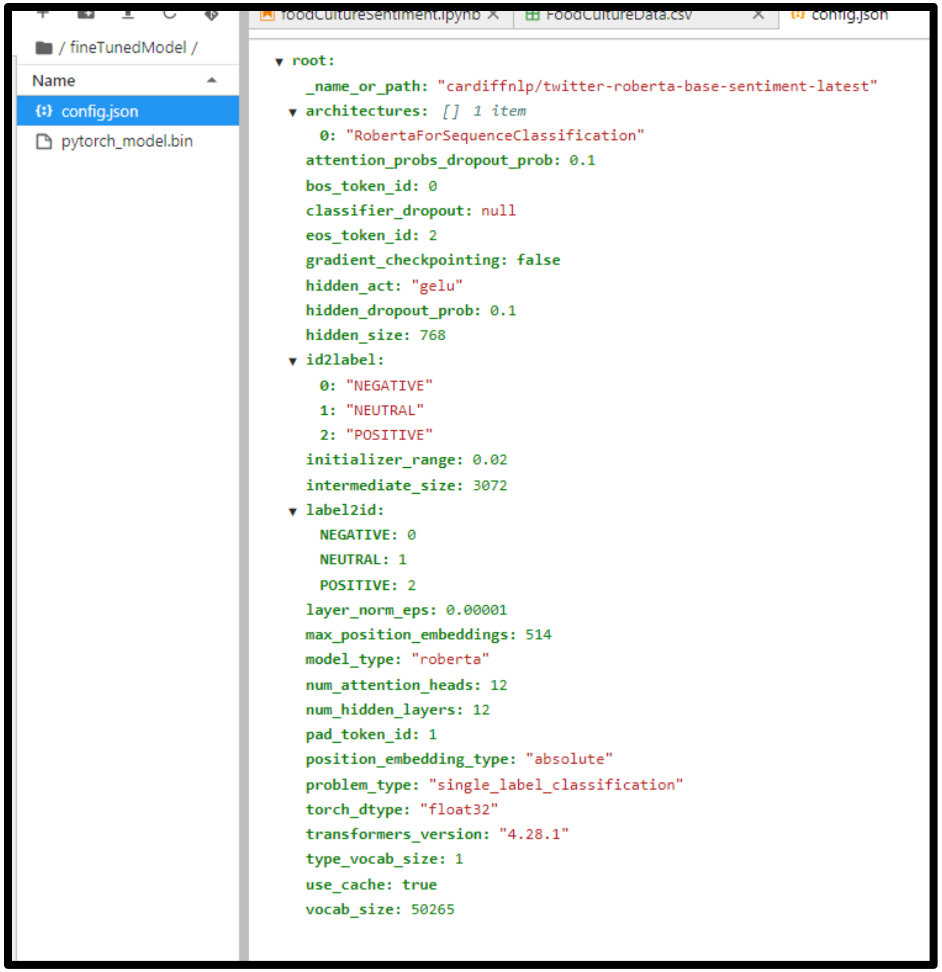
Transformers – The BIG deal…

- Transformers are an ***architecture*** that enables accurate and efficient transfer learning
  - Keeps track ONLY of model weights, biases, configuration
  - Customizable

- Encoder / Decoder
  - Encoder (extract meaning of input – all at once):
    - Document classification
    - Named entity recognition
    - Extractive Q&A
    - Ex: BERT (Bidirectional Encoder Representations from Transformers)
  - Decoder (access meaning of input – sequential):
    - Next word prediction
    - Text Generation
  - Encoder / Decoder (extract features, then produce sequential output):
    - Text Summarization
    - Translation



What do Transformers keep track of?




```
RobertaConfig {
  "attention_probs_dropout_prob": 0.1,
  "bos_token_id": 0,
  "classifier_dropout": null,
  "eos_token_id": 2,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "roberta",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 1,
  "position_embedding_type": "absolute",
  "transformers_version": "4.28.1",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 30522
}
```

Why is this nice?

- Common Architecture / Ecosystem enables **transfer learning**
  - "Reuse a pre-trained model as the starting point for a model on an adjacent task" – Layman's Terms
  - Repurposing on a related task is MUCH quicker
- Most language models begin with an MLM, trained at GREAT expense… then add a classifier.
  - MLMs vary based on tokenizers (BPE vs. WordPiece) and architecture (number of hidden layers, size of hidden layers, etc.)
  - You can even repurpose repurposed models (what we'll do today)
- Model Hub: https://huggingface.co/models

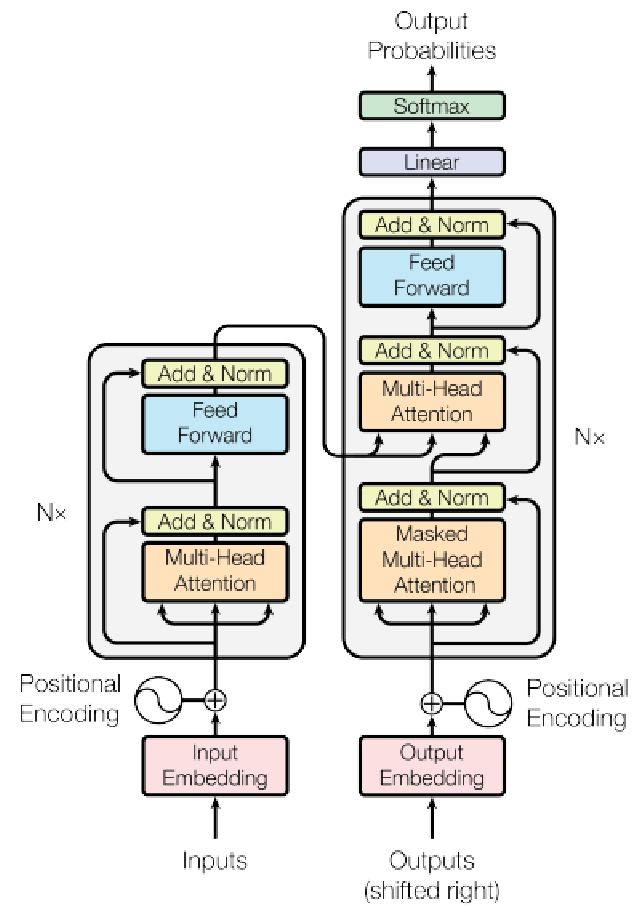
A look at BERT Base (110M parameters)

- Up to 512 tokens per sample
- Input Embedding (vocab size x embedding dim) 30522 x 768 + others = 24M
- 12 Layers: 7.1M x 12 = 85M
  – “Attention Head”: 7.1M
- Output Layer: 0.5M

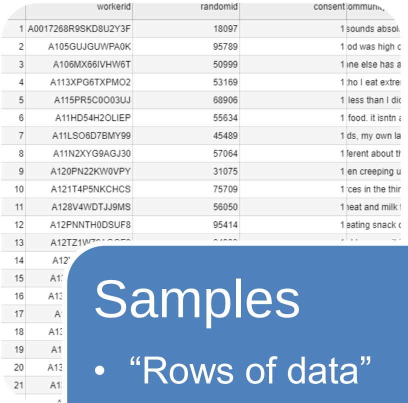
config.json

pytorch_model.bin

[Computing Parameters - Stack Overflow]

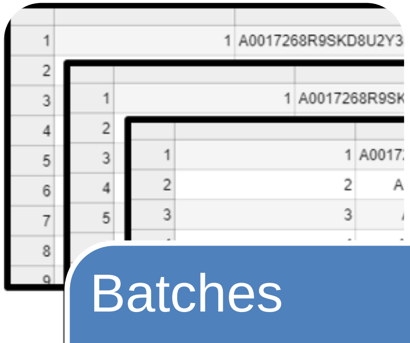


Batches, Samples, and Epochs, Oh, my!



Samples

- “Rows of data”
- documents or records – not tokens



Batches

- Enables efficient parallelism
- Batch Size = Samples / Batches
- Updates model weights each batch

Epochs

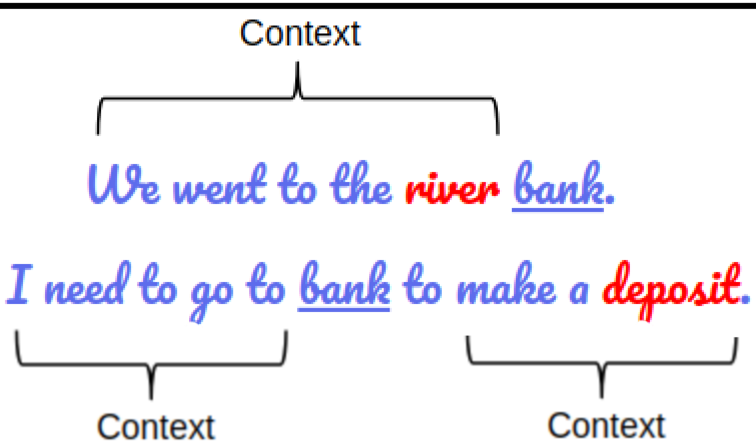
- How many times should the model see all data?
- Forward / Backward pass

MATHEMATICAL SCIENCES

Embedding Differences

Word Embeddings vs. Contextualized Embeddings

- *word embeddings*: words have the **same** vector representation **regardless** of context.
- *contextualized embeddings*: words have a **different** vector representation **dependent** on their context



Analytics Vidhya: Demistifying BERT.

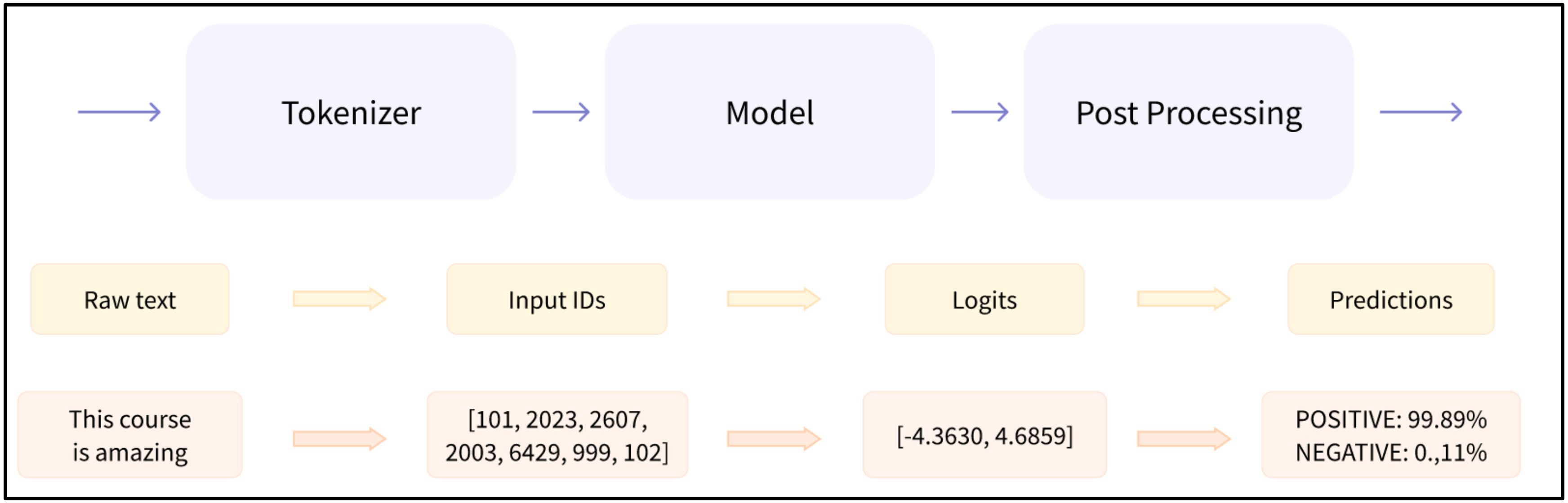
- Word2Vec: 'bank' vector is the same

- BERT: 'bank' vector is different.

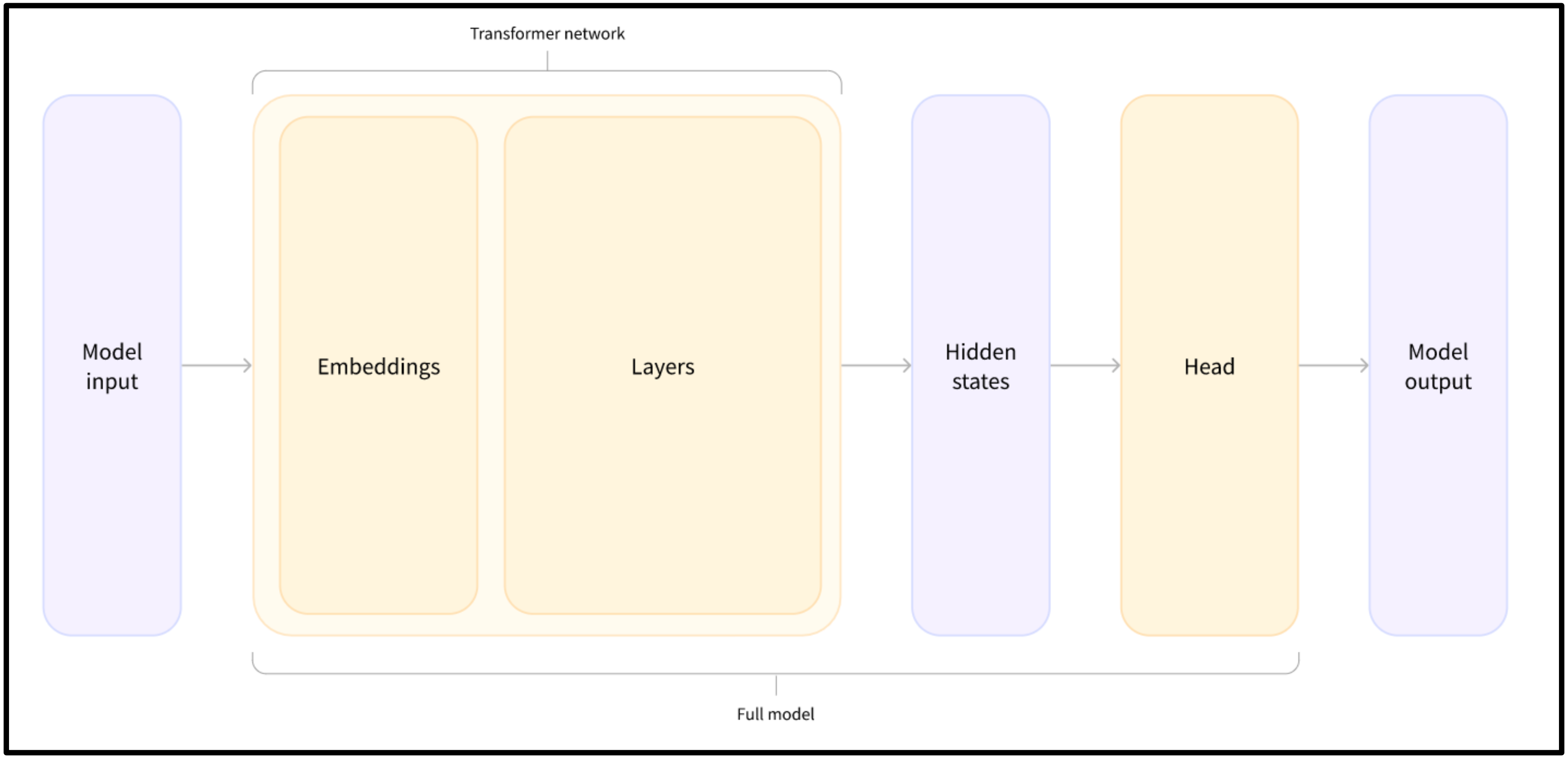
Text Processing Pipeline

General Pipeline

[Behind the pipeline - Hugging Face Course]



Zoomed-in View of Model



Hands-on Coding

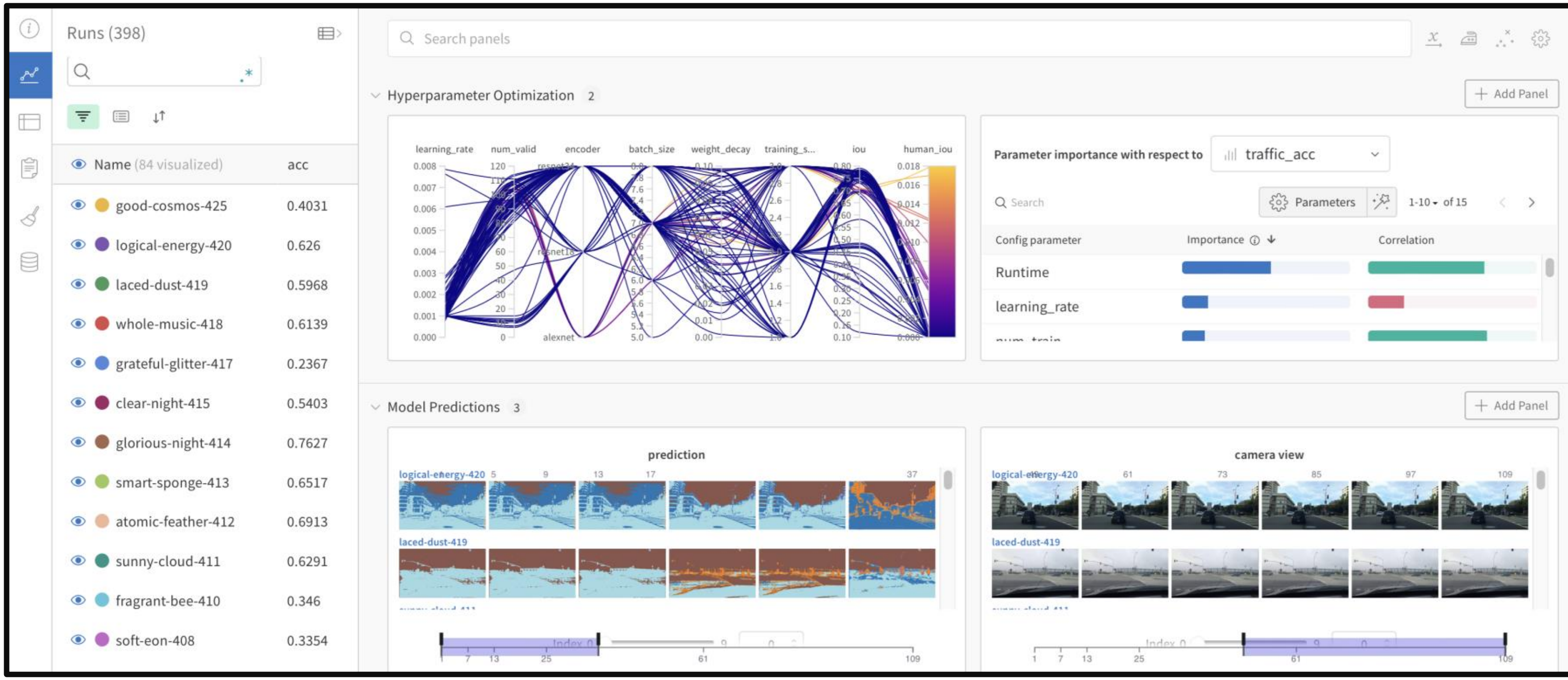
LESSON 5

Weights and Biases

UNITED STATES MILITARY ACADEMY
WEST POINT

Weights & Biases

Why? It automates your record keeping!



[W&B Docs | Weights & Biases Documentation (wandb.ai)](https://docs.wandb.ai)

[Intro_to_Weights_&_Biases.ipynb - Colaboratory (google.com)](https://colab.research.google.com)

MATHEMATICAL SCIENCES

Three calls to start logging!

1. wandb.init - Documentation
    1. `name` – a label for your run
    2. `project` – where you want to group your runs
    3. `entity` – username (important for collaboration)
    4. `tags`
    5. `notes`
    6. `config` – dictionary to store information about your run
2. wandb.config - Documentation
    a. dictionary of tracked items
    b. `argparse` compatible
3. wandb.log - Documentation – logging. This is the best part!

Weights & Biases

- Create a Weights & Biases account
- Create a project “`basic-demo`”
- In terminal:
  - `pip install wandb`
  - `wandb login` → use API key from “User Settings” to login

1. Quickstart Demo from W&B

- [Quickstart Colab Demo from W&B]()

2. Integrate with `Trainer` object from Hugging Face

- [Modified Colab demo from last lecture]()
- [Guide from W&B]()

Hands-on Coding

Fill in the blank...

Virtually ALL machine learning techniques require lots of ____________ to work well.

So... for a custom project, using custom data, how do we do that, exactly?

Labelling Data

- Many techniques – but I recommend: https://labelstud.io/ Label Studio
- Why?
  - Distributed labelling (more than 1 at a time)
  - Now hosted on ACI's WIRE https://icsarl.westpoint.edu/wire
  - Versatile
    - Text? Sure.
    - Image? Sure.
    - Audio? Sure.
  - Let's take it for a test drive!

Topic Modeling

- Topic modeling is an unsupervised machine learning technique to reduce dimensionality of your corpus.*
  - Start with 1000 unique documents
  - Group / Cluster ones that are similar
  - Think PCA for text data
- Two main techniques:
  - Lexical (LDA)
  - Transformer (BERTopic)
- Let's take this for a test drive on our course end feedback.

* An alternative to topic modeling is topic classification model (supervised)

Research Opportunities in D/Math!

Questions?

Additional Slides

Links

1. Old Demos
   a. Colab Notebooks:
      i. Fine-tuning a model in Google Colab [training]
      ii. Using a fine-tuned model in Google Colab

   b. Hugging Face: Hugging Face – The AI community building the future.

2. NLP
   a. **CS287: Deep Learning for NLP** is a course Jack took in grad school. It was only taught for one semester, but the lecture slides are all available online (for now) [link - scroll to "Lectures"]. There's also a helpful Supplemental Resources page for NLP.

   b. Hugging Face (particularly their Transformers library) is one of the most popular communities/platforms for NLP/AI/ML. They have great tutorials and readable documentation. Hugging Face models and datasets are designed to work well with and/or build on other popular NLP packages [nltk, AllenNLP, spacy].

3. Python
   a. If you want to using Python without downloading it – Google Colab is a free online version of a Jupyter notebook with incredible functionality.

   b. Jack uses VSCode as my IDE for Python.

   c. Most individuals use the Anaconda Distribution for Windows. You can find instructions here: [link]

   d. Jack paid ~$15 back in 2020 for a Udemy course called "Complete Python Bootcamp". [link – the video to install Python is available as a free preview Python Setup>Installing Python (Step by Step)]. The course lecture slides and materials are freely available. [slides] [github repo w/ Jupyter Notebooks].

   e. Google also has a free Python Course. [link]

…More Links

1. Sebastian Ruder [his site] [his Google Scholar] has a great blog that covers new ideas in NLP, NLP Progress, his highlights from NLP conferences…etc. In 2018 he gave a brief timeline of NLP. https://ruder.io/a-review-of-the-recent-history-of-nlp/.

2. If you want straightforward explanations of how the most used NLP models work, Jay Alammar has a blog with great visuals and explanations for word2vec (2013) and BERT (2018+) toward the bottom of his blog posts.