

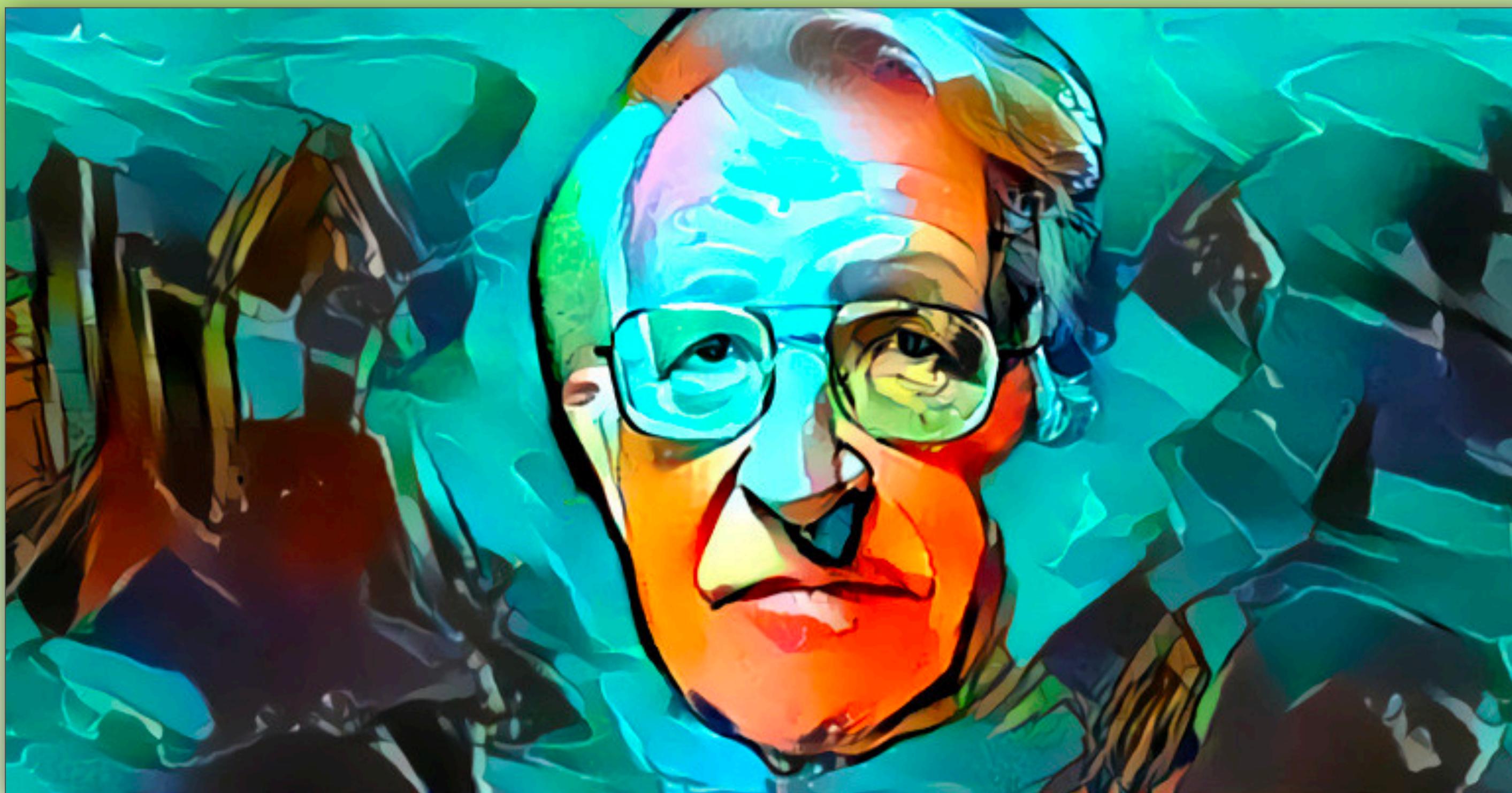
# spaCy tuTorial

Paco Nathan @pacoid



derwen.ai

<https://bit.ly/31wal51>



# part 01: parsing natural language

# Parsing Natural Language

NLP used to be mostly concerned about **transformational grammars**, linguistic theory by Chomsky, etc. Other former approaches such as **link grammars** have since been largely overlooked.

However, ML techniques allow much simpler approaches called *statistical parsing*. With the rise of Big Data, these techniques became even more effective – **Google won the NIST 2005 Machine Translation Evaluation** and remains a leader.

Probabilistic methods split texts into sentences, annotate words with part-of-speech, chunk noun phrases, resolve named entities, calculate similarity of documents, etc.

# The Era of Formal Grammars

**terminals**

{generate, hate, great, green, ideas, linguists}

**nonterminals**

{SENTENCE, NOUNPHRASE, VERBPHRASE, NOUN, VERB, ADJ}

**production rules**

$\text{SENTENCE} \rightarrow \text{NOUNPHRASE } \text{VERBPHRASE}$

$\text{NOUNPHRASE} \rightarrow \text{ADJ } \text{NOUNPHRASE}$

$\text{NOUNPHRASE} \rightarrow \text{NOUN}$

$\text{VERBPHRASE} \rightarrow \text{VERB } \text{NOUNPHRASE}$

$\text{VERBPHRASE} \rightarrow \text{VERB}$

$\text{NOUN} \rightarrow \text{ideas}$

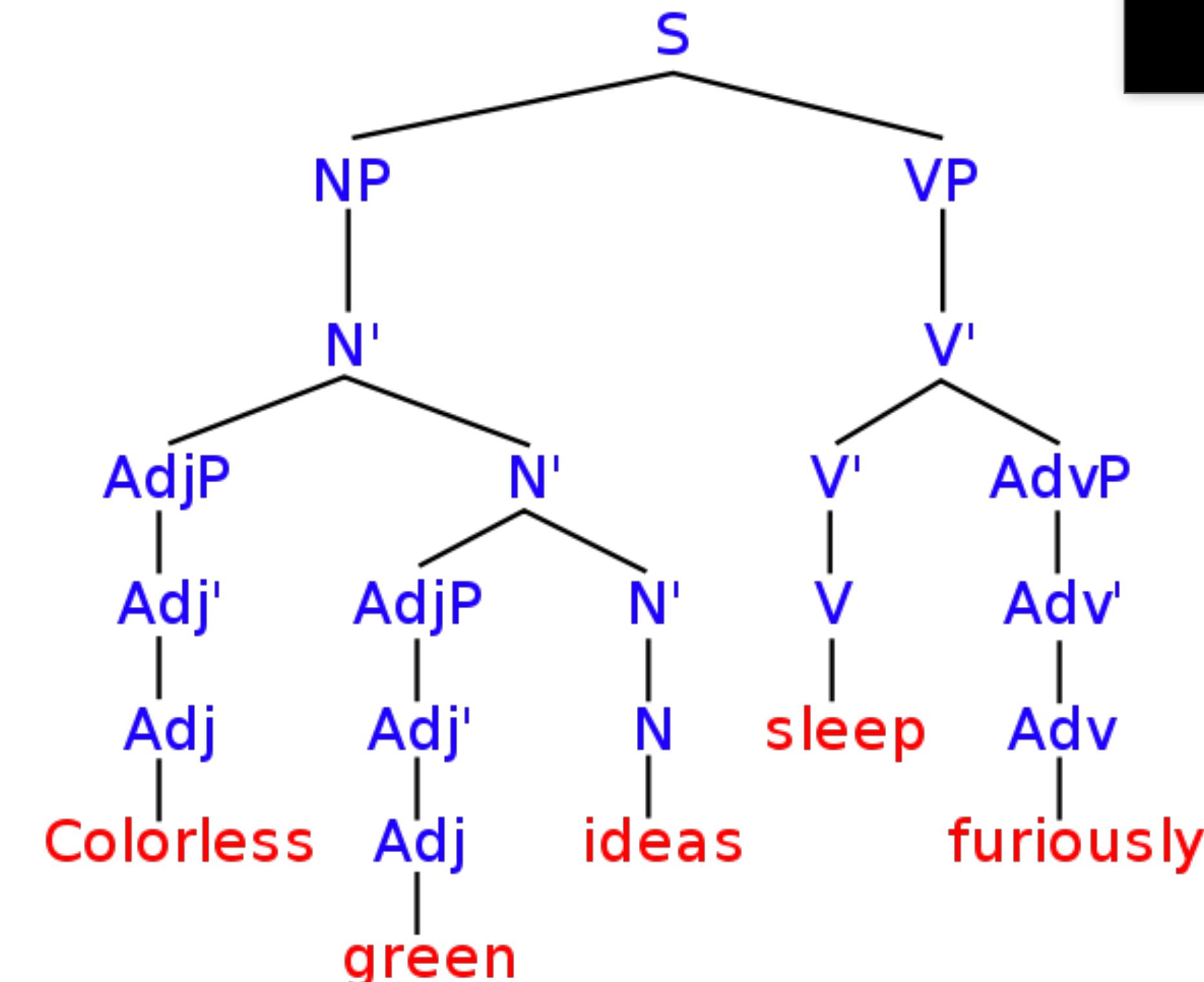
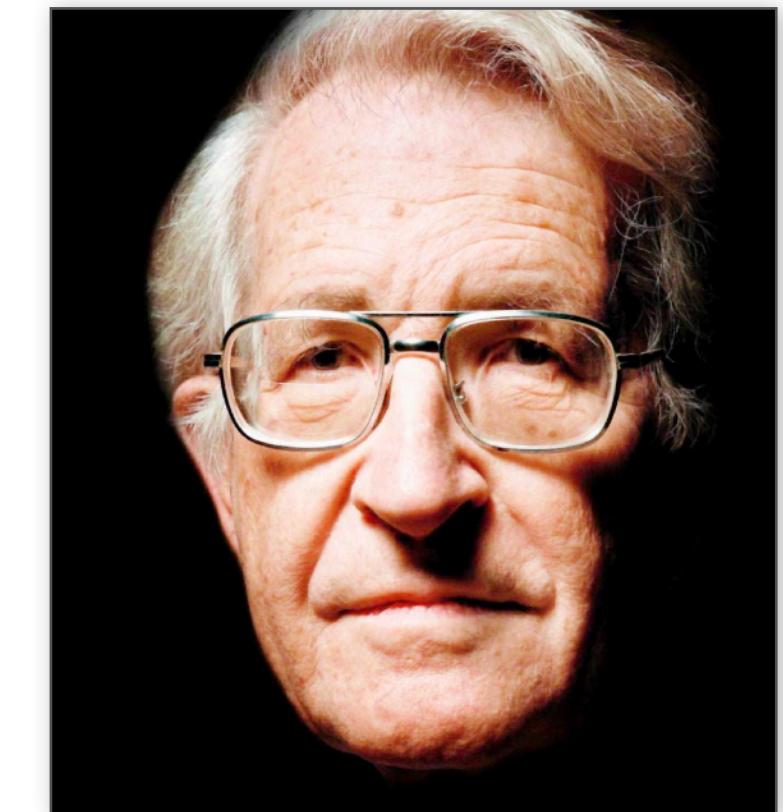
$\text{NOUN} \rightarrow \text{linguists}$

$\text{VERB} \rightarrow \text{generate}$

$\text{VERB} \rightarrow \text{hate}$

$\text{ADJ} \rightarrow \text{great}$

$\text{ADJ} \rightarrow \text{green}$



# Statistical Parsers

“Statistical Relational Learning”

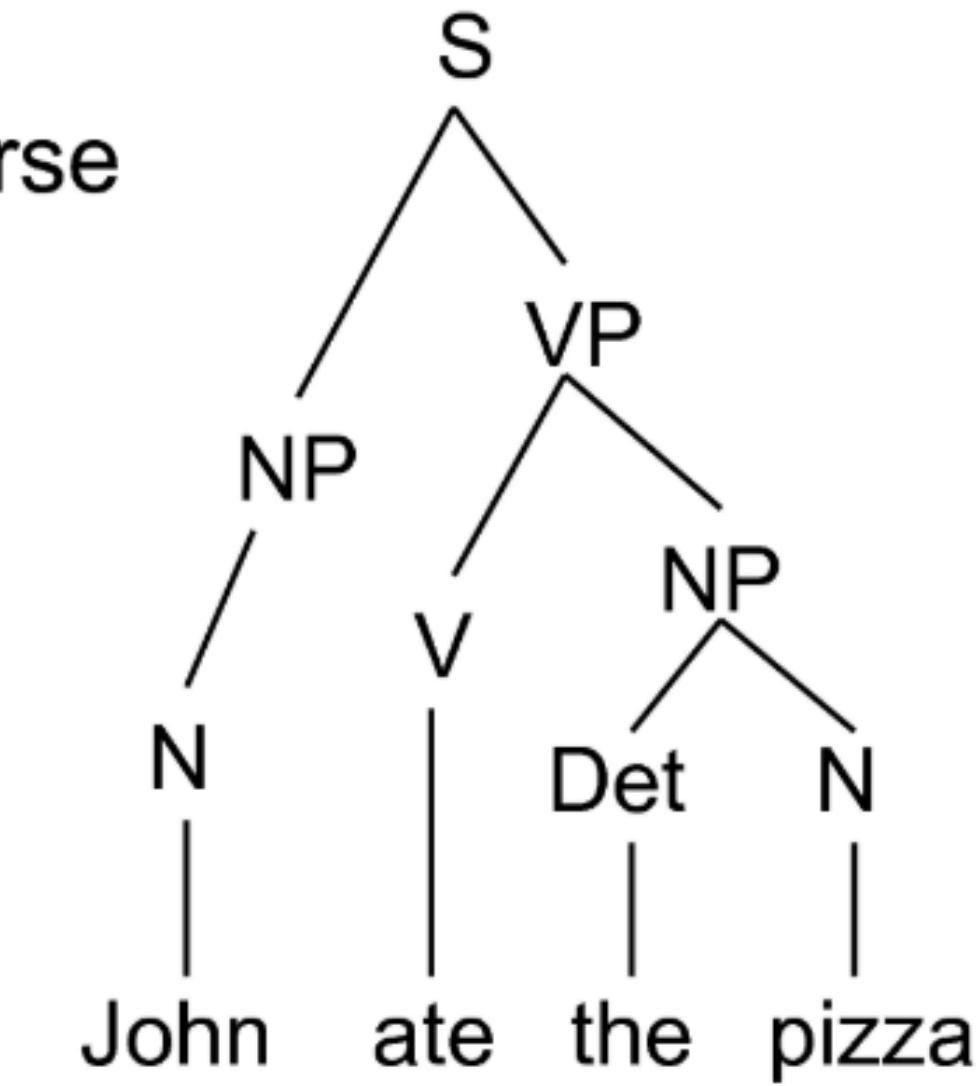
Pedro Domingos

see slides 96-97



## Statistical Parsing

- **Input:** Sentence
- **Output:** Most probable parse
- **PCFG:** Production rules with probabilities  
E.g.: 0.7  $NP \rightarrow N$   
0.3  $NP \rightarrow Det\ N$
- **WCFG:** Production rules with weights (equivalent)
- Chomsky normal form:  
 $A \rightarrow B\ C$  or  $A \rightarrow a$



# part 02: intro to spaCy

# Using Google Colab

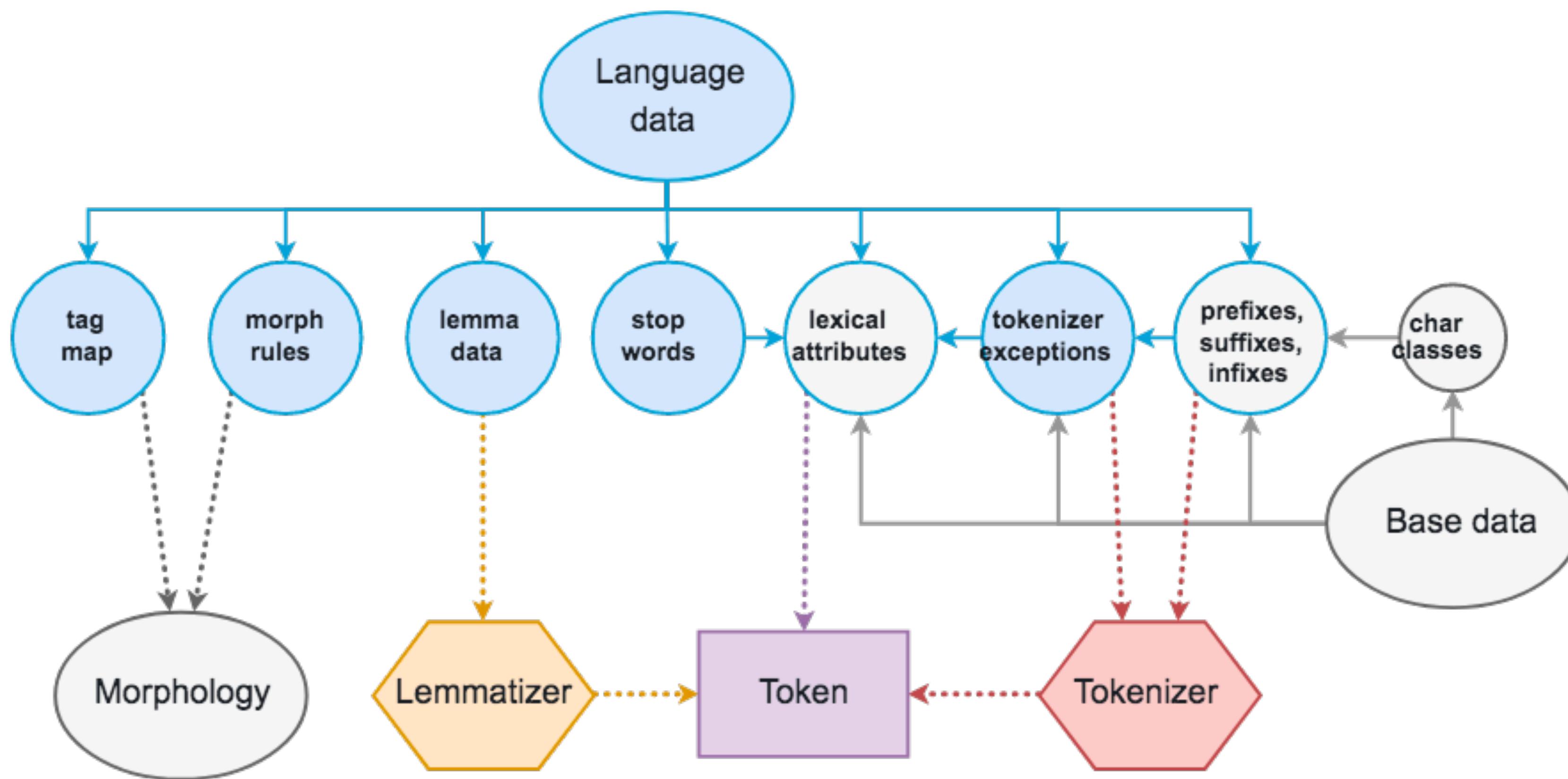
We'll be working with Jupyter notebooks, loaded from GitHub, then running inside **Google Colab**

Use this URL: [https://github.com/DerwenAI/spaCy\\_tutorial](https://github.com/DerwenAI/spaCy_tutorial)

The screenshot shows the Google Colab interface with the title "Overview of Collaboratory Features". The left sidebar contains a "Table of contents" with links to various sections: Cells, Code cells, Text cells, Adding and moving cells, Working with python, System aliases, Magics, Tab-completion and exploring code, Exception Formatting, Rich, interactive outputs, Integration with Drive, Commenting on a cell, and SECTION. The right panel is titled "Cells" and contains a detailed description of what a notebook is: "A notebook is a list of cells. Cells contain either explanatory text or executable code and its output. Click a cell to select it." It then describes "Code cells": "Below is a **code cell**. Once the toolbar button indicates CONNECTED, click in the cell to select it and execute the contents in the following ways:" followed by a bulleted list: "Click the **Play icon** in the left gutter of the cell; Type **Cmd/Ctrl+Enter** to run the cell in place; Type **Shift+Enter** to run the cell and move focus to the next cell (adding one if none exists); or Type **Alt+Enter** to run the cell and insert a new code cell immediately below it." It also notes that there are additional options for running some or all cells in the "Runtime" menu. Below this, a code cell displays the Python command `a = 10` and its output `10`. The "Text cells" section explains that this is a **text cell** and describes how to edit it using markdown syntax. It also mentions that LaTeX can be added using `\$` signs.

# Intro to spaCy

Let's get started with [spaCy](#)... open this [notebook](#) in [Google Colab](#) from GitHub then go to the section titled “Getting Started”



# Other Popular NLP Frameworks

**Stanford Core**

[nlp.stanford.edu/software/](http://nlp.stanford.edu/software/)

**Spark NLP**

[nlp.johnsnowlabs.com/](http://nlp.johnsnowlabs.com/)

**Flair NLP**

[research.zalando.com/welcome/mission/research-projects/flair-nlp/](http://research.zalando.com/welcome/mission/research-projects/flair-nlp/)

# Multiple Language Support

One frequently asked question is about *multi-language support* for NLP tools. Here are some links:

- <https://spacy.io/usage/adding-languages>
- <http://globalwordnet.org/wordnets-in-the-world/>
- <https://wordnet.princeton.edu/related-projects>
- <http://compling.hss.ntu.edu.sg/omw/>
- <https://nlp.stanford.edu/software/lex-parser.shtml>

# **part 03:**

## **acquiring text**

# Character Encoding

The example text that we've used is relatively “clean” ...  
that rarely happens in practice!

- “[Text vs. Data Instead of Unicode vs. 8-bit](#)”
- [ligatures](#) used in publishing add to the fun
- [codecs](#) become especially important for storing text in files
- See also about [Unicode equivalence](#) and how to [normalize](#) text in Python.

Open the `ex02.ipynb` notebook

# Character Encoding

Resolving character encoding issues is a hard problem, and will continue to be so – perhaps it's suitable as a use case for AI research...

For more details and further context about use cases in search, check the excellent article “[Character Filtering](#)” by Daniel Tunkelang, along with his entire series about [Query Understanding](#).



# Semantic Similarity

Semantic similarity between two texts can be measured using a variety of approaches. [Jaccard and Tanimoto](#) are two statistical measures.

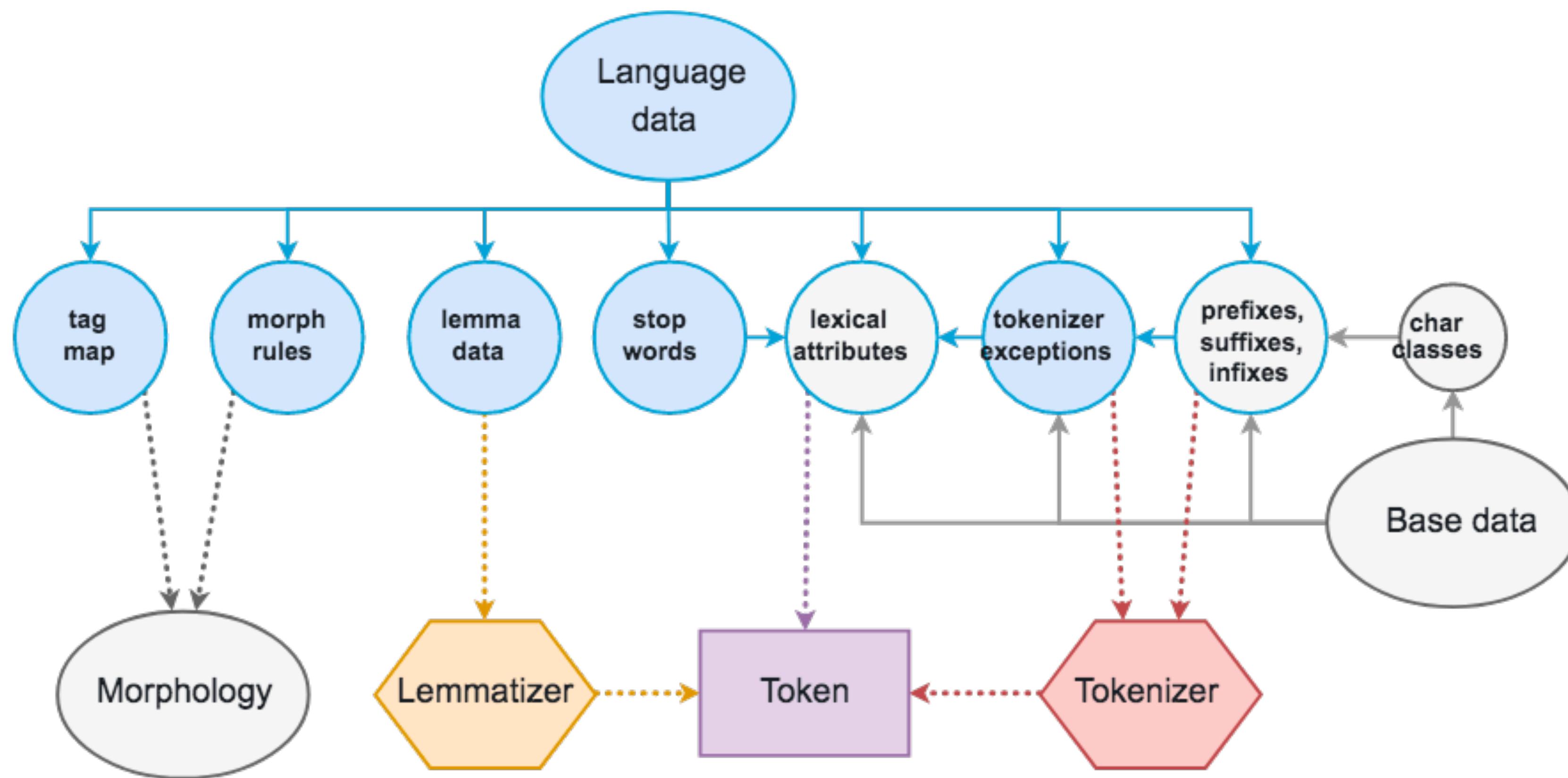
We can approximate a Jaccard similarity with [MinHash](#) – the [Probabilistic Data Structures in Python](#) tutorial has more detailed background about this approach.

That is mostly based on the [datasketch](#) library, building on work by [Philippe Flajolet](#), et al.



# spaCy

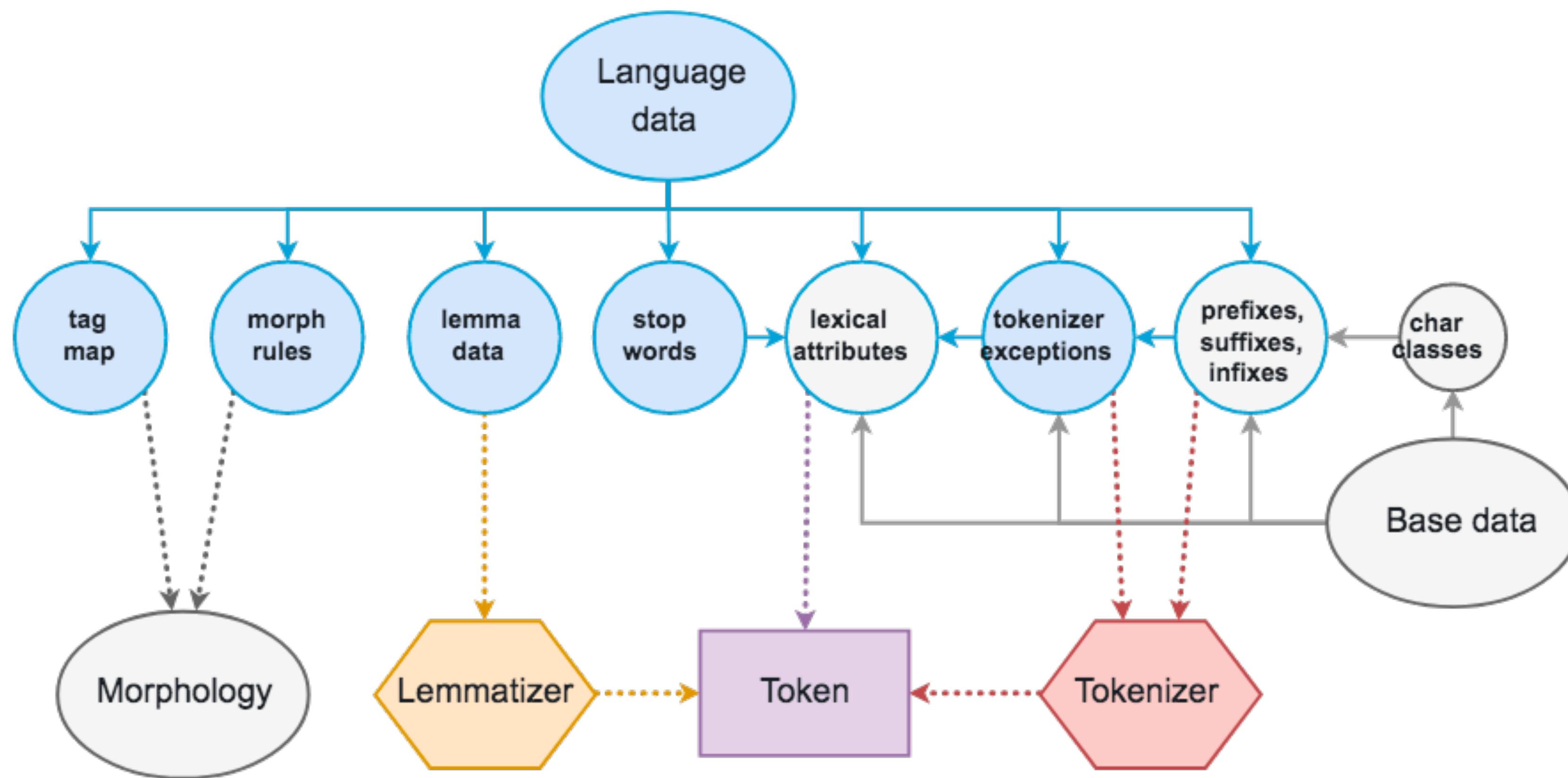
Let's explore more of [spaCy](#)... still using that same [notebook](#) in [Google Colab](#), go to the section titled “Acquiring Text”



# part 04: understanding text

# spaCy

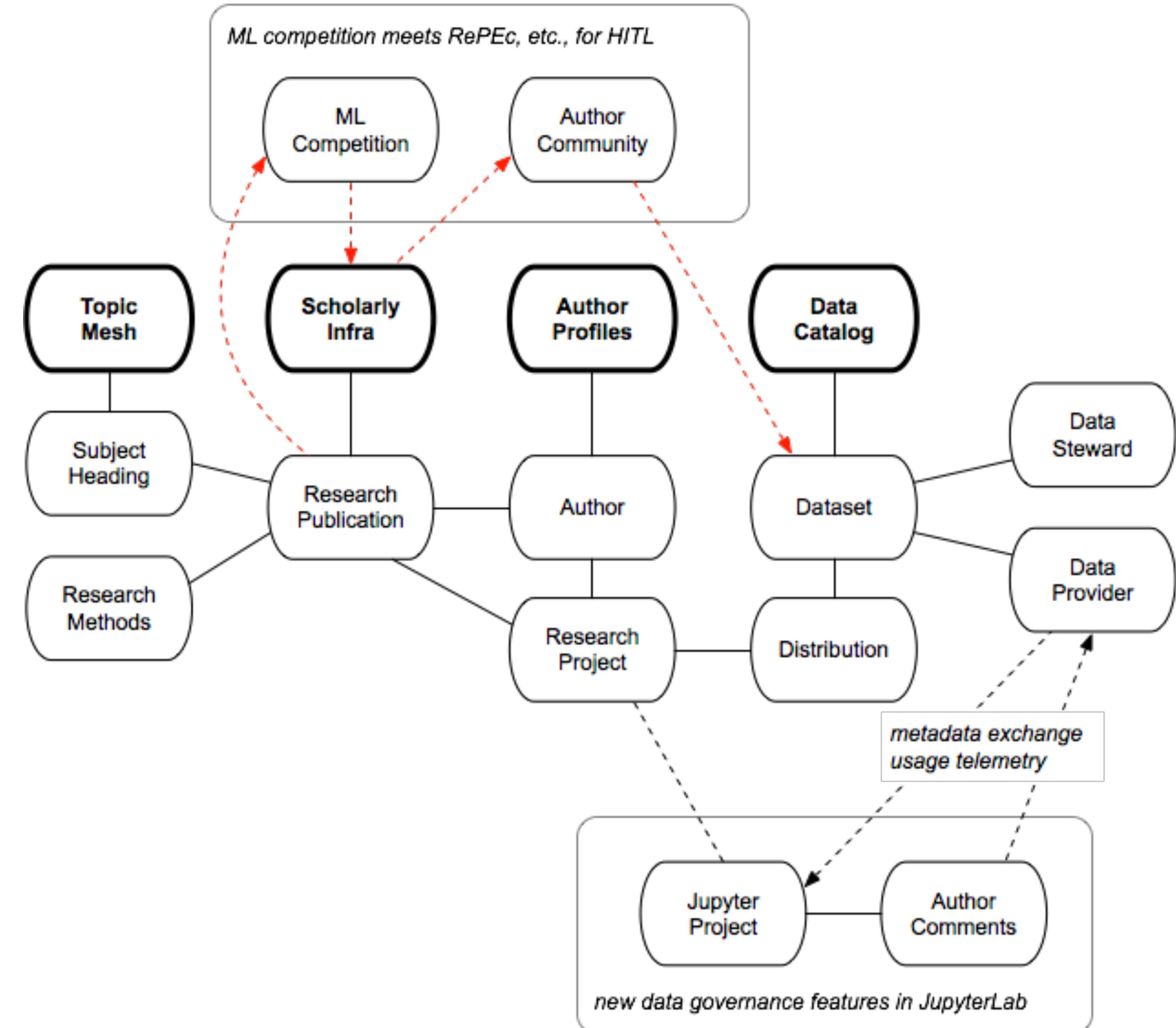
Let's explore more of [spaCy](#)... still using that same [notebook](#) in [Google Colab](#), go to the section titled “Natural Language Understanding”



# Knowledge Graph

the **vocabulary** used in the graph  
integrates W3C metadata standards:  
**DCAT**, **PAV**, **CITO**, **FaBiO**, **FOAF**,  
**DCMI**, etc.

initial problem of **entity linking**:  
leverage ML models to infer dataset  
attribution, i.e., link from research  
publications to a **known set of  
datasets**



# Serialized Formats: TTL, JSON-LD

rc.ttl

Raw

```
1 @base <https://github.com/Coleridge-Initiative/adrf-onto/wiki/Vocabulary> .  
2  
3 @prefix cito: <http://purl.org/spar/cito/> .  
4 @prefix dct: <http://purl.org/dc/terms/> .  
5 @prefix foaf: <http://xmlns.com/foaf/0.1/> .  
6 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
7 @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .  
8  
9 :publication-f909edfc5b2812f512fb  
10    rdf:type :ResearchPublication ;  
11    foaf:page "http://europepmc.org/articles/PMC5353819"^^xsd:anyURI ;  
12    dct:publisher "J Family Med Prim Care" ;  
13    dct:title "Subclinical hypothyroidism and the risk of hypercholesterolemia" ;  
14    dct:identifier "10.1370/afm.79" ;  
15    :openAccess "https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1466694/pdf/0020351.pdf"^^xsd:anyURI ;  
16    cito:citesAsDataSource :dataset-0a7b604ab2e52411d45a ;  
17 .  
18  
19 :dataset-0a7b604ab2e52411d45a  
20    rdf:type :Dataset ;  
21    foaf:page "https://wwwn.cdc.gov/nchs/nhanes/"^^xsd:anyURI ;  
22    dct:publisher "Centers for Disease Control and Prevention" ;  
23    dct:title "National Health and Nutrition Examination Survey" ;  
24    dct:alternative "NHANES" ;  
25    dct:alternative "NHANES I" ;  
26    dct:alternative "NHANES II" ;  
27    dct:alternative "NHANES III" ;  
28 .
```

# ML Competitions

- **RCC 2018-2019** finalists Allen AI, KAIST, GESIS, Paderborn
- participating teams contributed to an upcoming book:  
*Rich Search and Discovery for Research Datasets: Building the next generation of scholarly infrastructure*
- new **RC leaderboard** on GitHub: Rich Context KG produces an open corpus for the competition, using open access PDFs
- evaluation metric based on `Top5UptoD` relevance ranking
- initial baseline at >78% precision for entity linking of datasets

source	precision	repo	corpus	date	contact
LARC	78.36	<a href="#">link</a>	v0.1.5	2019-09-26	<a href="#">@philipskokoh</a>

# Related Conferences

**AKBC** (UC Irvine)

**Knowledge Graph Conference** (Columbia U, NYC)

**Connected Data London** (London)

**spaCy IRL** (Berlin)



# Extracting Text from PDF Files

In addition to using **Beautiful Soup** to extract text from HTML, another common need is to extract text from PDF files.

We'll show how to use **PDFx** by Chris Hager to handle this kind of work. Use the **notebook in Google Colab** to run an example.

Other popular libraries for PDF text extraction include:

- <https://pypi.org/project/PyPDF2/>
- <https://github.com/euske/pdfminer>
- <https://tika.apache.org/>

# part 05: examples

# Examples

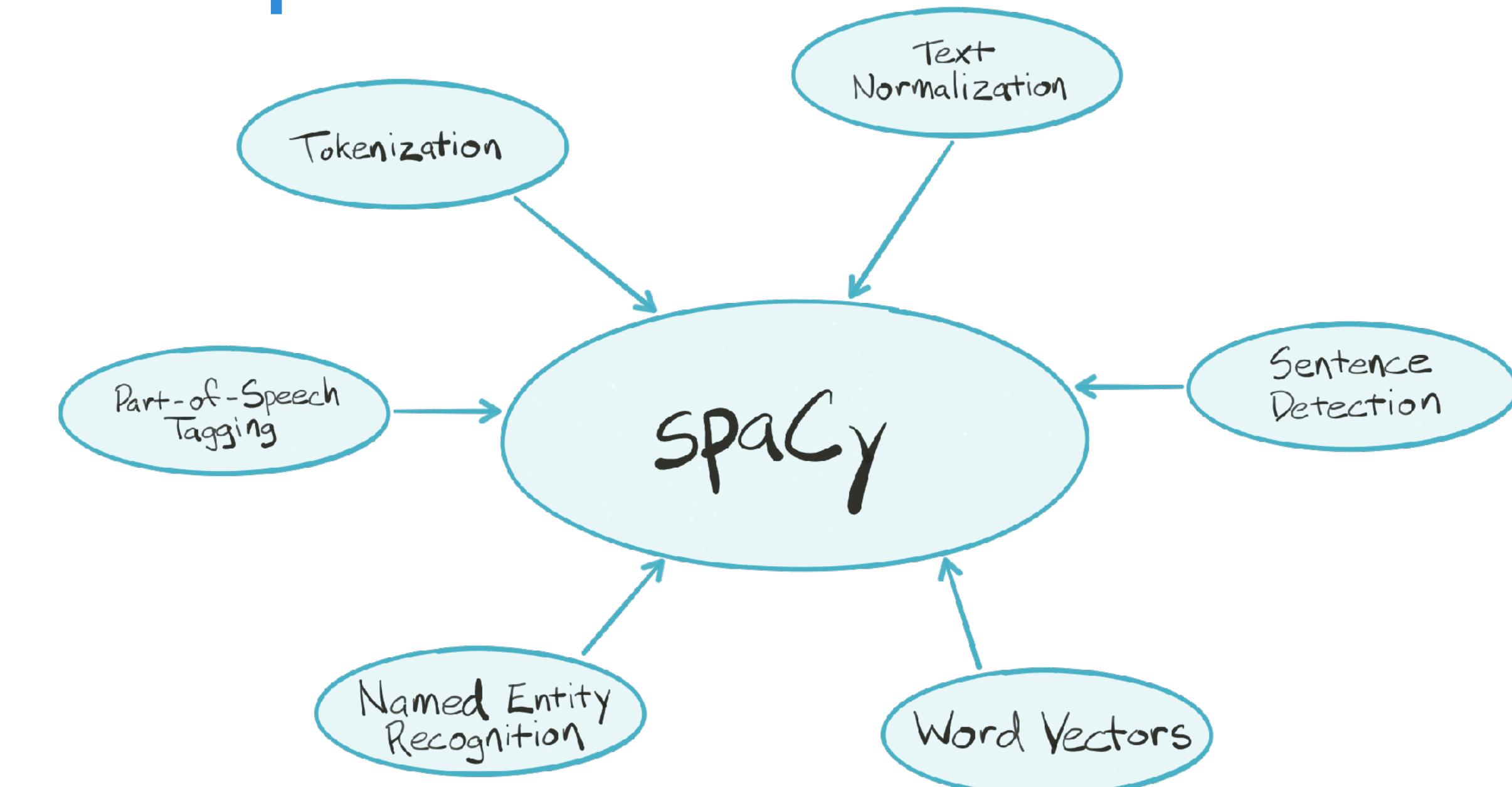
## “Modern NLP in Python”

Patrick Harrison, S&P Global Market Intelligence

- large corpus data mining
- LDA, Word2Vec, t-SNE, etc.
- knowledge graph construction

## “Maverick, Goose begin romantic relationship”

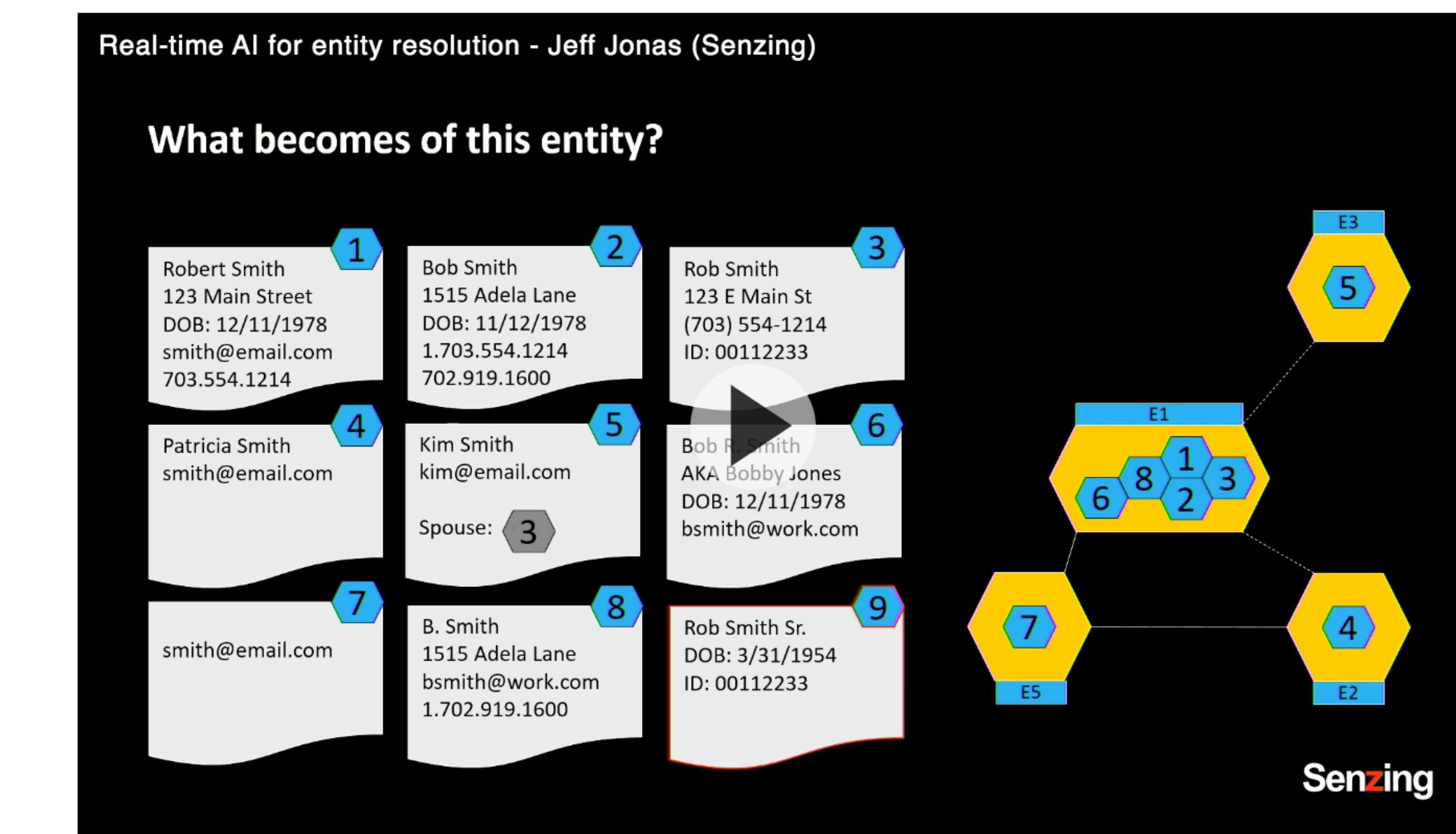
John Bohannon, Primer AI



# Examples

One of the leading commercial vendor for *named entity resolution* is from **Senzing**, led by Jeff Jonas

- “Real-time AI for entity resolution”
- “Real-time entity resolution made possible”



# Examples

**Long short-term memory** (LSTM) is an approach that allows recurrent neural networks to learn over many time steps. These can be used for time-series analysis, and also for learning sequences of data, such as in streams of voice or text.

Imagine feeding many scripts (semi-structured text) from a particular film genre through an LSTM, then generating new output.

For example that's what **Ross Goodwin** does...



# Using LSTM to generate film scripts



*Sunspring*  
[youtu.be/LY7x2lhqjmc](https://youtu.be/LY7x2lhqjmc)

by **Benjamin AI**

*It's No Game*  
[goo.gl/Waw5Px](http://goo.gl/Waw5Px)



# Examples

## Botnik Studios with Voicebox, etc.

The screenshot shows a Medium article page. At the top right are icons for search, bookmark, notifications, and profile. Below that, it says "Recommended by Quincy Larson, deb siegel, and 15 others". The author's profile picture and name, "Elle O'Brien", are shown, followed by her title, "Computational scientist, software developer, science writer", and the date, "Aug 6 · 6 min read". The main title of the article is "Romance Novels, Generated by Artificial Intelligence". The first paragraph discusses the author's fascination with romance novels and their interest in generating them using neural networks.

I've always been fascinated with romance novels—the kind they sell at the drugstore for a couple of dollars, usually with some attractive, soft-lit couples on the cover. So when I started futzing around with text-generating neural networks a few weeks ago, I developed an urgent curiosity to discover what artificial intelligence could contribute to the ever-popular genre. Maybe one day there will be entire books written by computers. For now, let's start with titles.

The screenshot shows the Voicebox application interface. At the top right is a "Predictive Writer" button. The main area displays a grid of words from a "Cinnamon Pancake Recipes" source. The word "at" is highlighted in yellow, indicating it is the predicted next word. The grid is organized into columns and rows, with numbers 1 through 9 indicating word positions. The bottom of the screen shows keyboard shortcuts for navigating the grid.

Source: Cinnamon Pancake Recipes				Shuffle ✖	
in	1	for	2	at	3
rolls	4	the	5	until	6
8	7	on	8	1	9
buns	ALT+0	loaves	ALT+1	60	ALT+2

# part 06: transformers

# Embedded Language Models

**ELMo**: Deep Contextualized Word Representations

**BERT**: State-of-the-Art Pre-training for Natural Language Processing

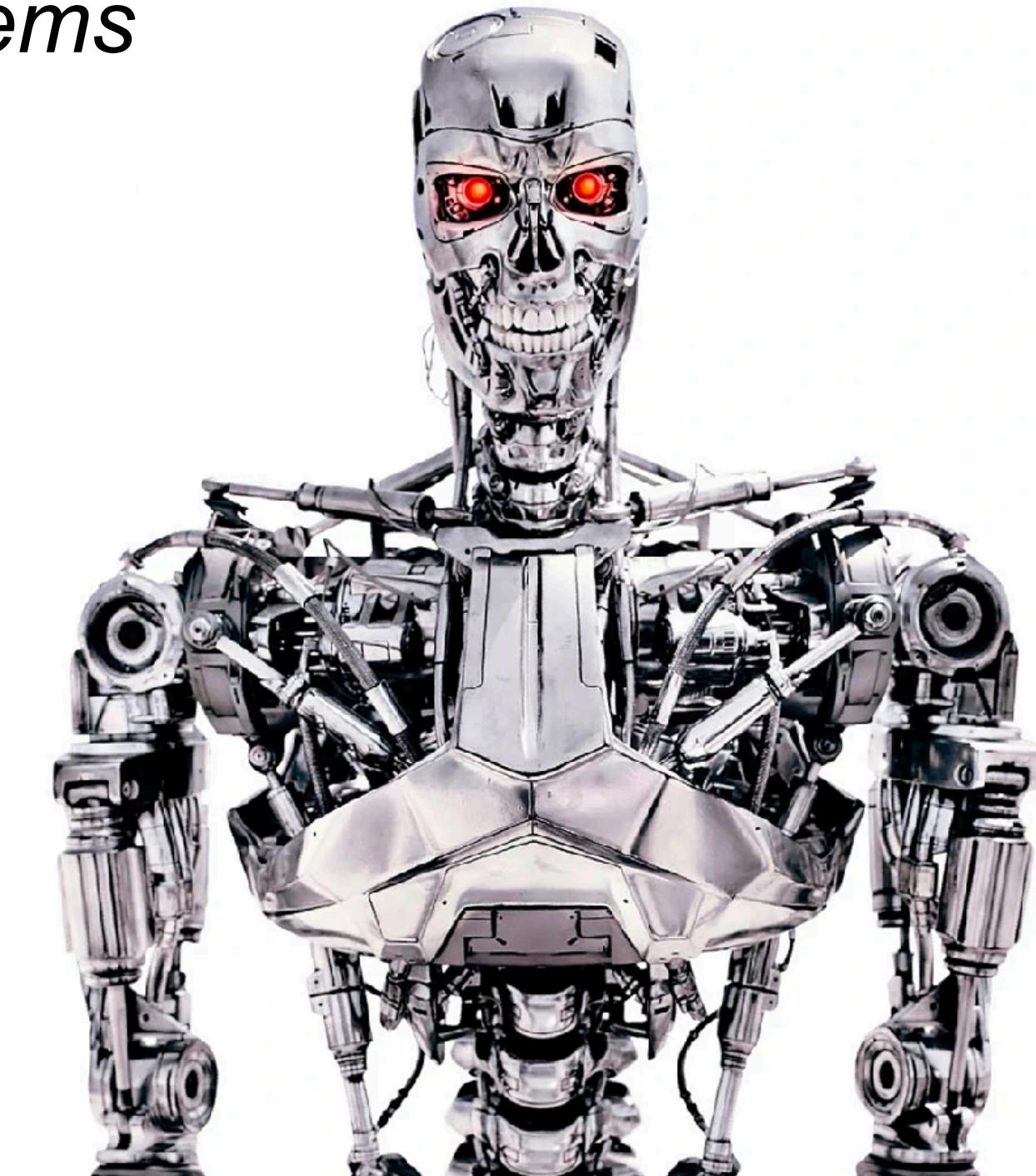
**ERNIE**: Enhanced Representation through kNowledge IntEgration

**XLNet**: Generalized Autoregressive Pretraining for Language Understanding

**GPT-2**: Language Models are Unsupervised Multitask Learners

**RoBERTa**: An optimized method for pretraining self-supervised NLP systems

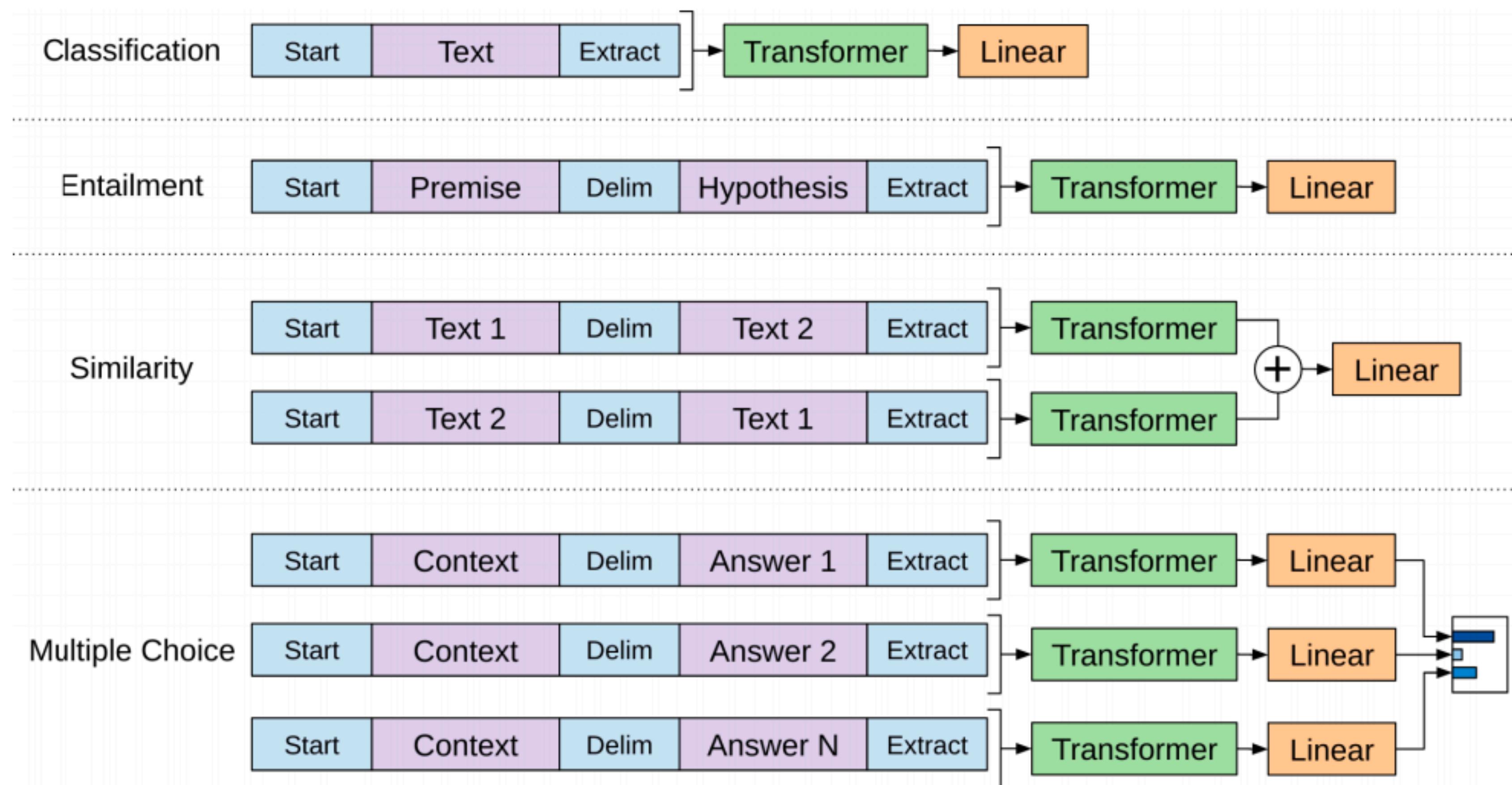
**DistilBERT**: Knowledge distillation during the pre-training phase



# Transformers

“The Illustrated BERT, ELMo, and co.  
(How NLP Cracked Transfer Learning)”

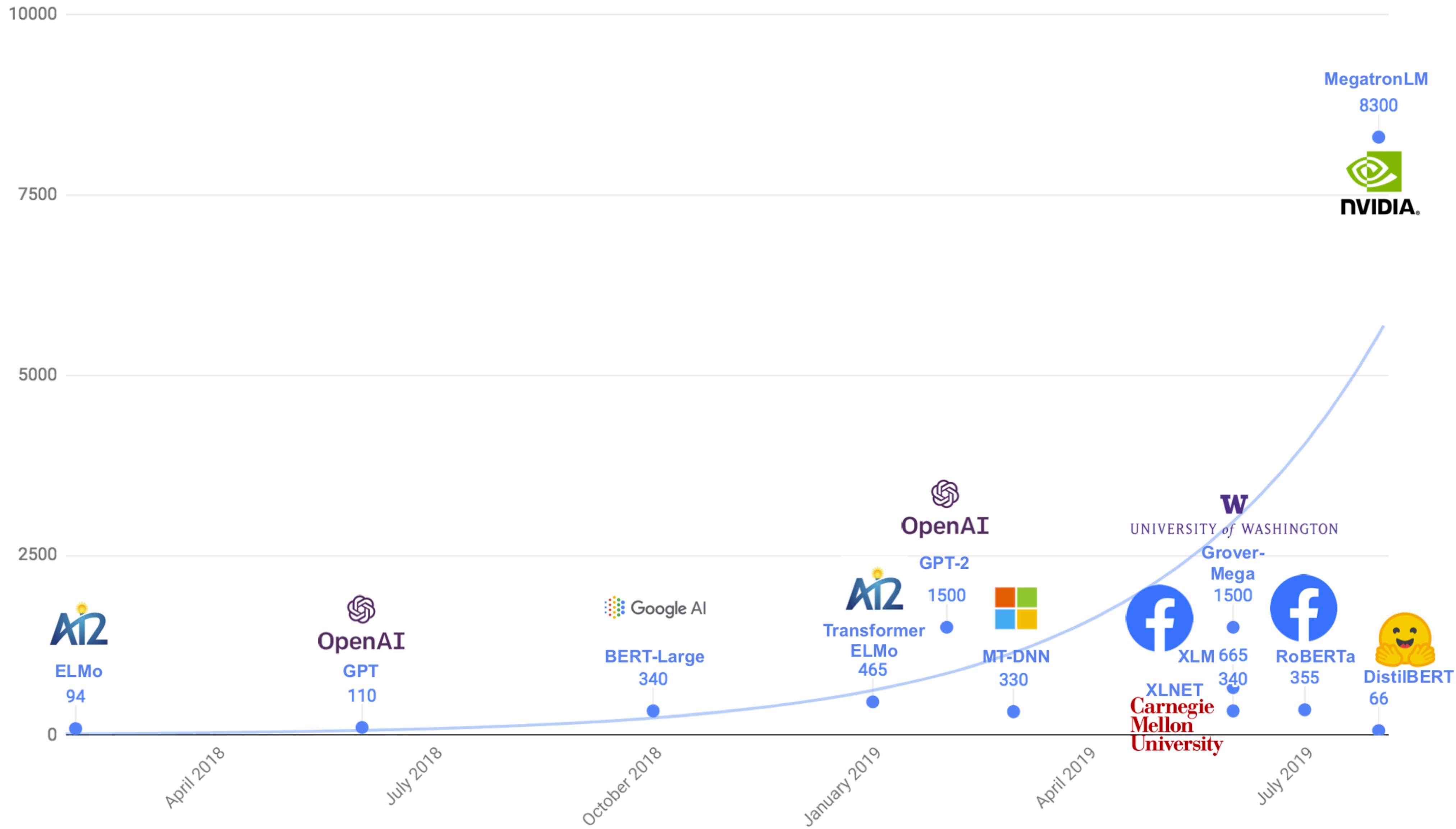
Jay Alammar



# Transformers

word embeddings	Word2Vec, GloVe, etc.	<ul style="list-style-type: none"><li>use a vector (a list of numbers) to represent words in a way that captures semantic or meaning-related relationships</li><li>pre-trained on vast amounts of text data</li><li>creates a kind of linear algebra for computing semantics</li></ul>
language modeling	ELMo	<ul style="list-style-type: none"><li>contextualized word-embeddings</li><li>gained its language understanding from being trained to predict the next word in a sequence of words</li><li>a model can learn from vast amounts of text data without needing labels</li></ul>
transfer learning	ULM-FiT	<ul style="list-style-type: none"><li>introduced a process to effectively fine-tune that language model for various tasks</li><li>NLP finally had a way to do transfer learning probably as well as Computer Vision could</li></ul>
masked language model	BERT	<ul style="list-style-type: none"><li>a transformer-based model where the language model looks both forward and backwards</li><li>openAI transformer had provided a fine-tunable pre-trained model, but it wasn't bi-directional like ELMo</li></ul>
unsupervised multitask learners	GPT-2	<ul style="list-style-type: none"><li>both unsupervised learning and supervised learning</li><li>subword representation, obtained by Byte Pair Encoding (BPE)</li></ul>
model compression	distilBERT	<ul style="list-style-type: none"><li>a compression technique in which a small model is trained to reproduce the behavior of a larger model</li><li>about 50% as many parameters as BERT, retaining 95% of its performances on the language understanding benchmark GLUE</li></ul>

# Transformers – growth of parameter size



# Distillation

“DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter”

“Hugging Face Implements SOTA Transformer Architectures for PyTorch and TensorFlow 2.0”

“EIE: Efficient Inference Engine on Compressed Deep Neural Network”

<https://huggingface.co/transformers/>

*“...it is possible to reduce the size of a BERT model by 40%, while retaining 97% of its language understanding capabilities and being 60% faster...”*

# Leaderboards

- **GLUE leaderboard**
- **NLP-Progress**
- **Papers With Code state-of-the-art (SOTA)**

# spaCy transformers

“spaCy meets Transformers: Fine-tune BERT, XLNet and GPT-2”

Matthew Honnibal, Ines Montani

- [notebook](#) (load in Google Colab)

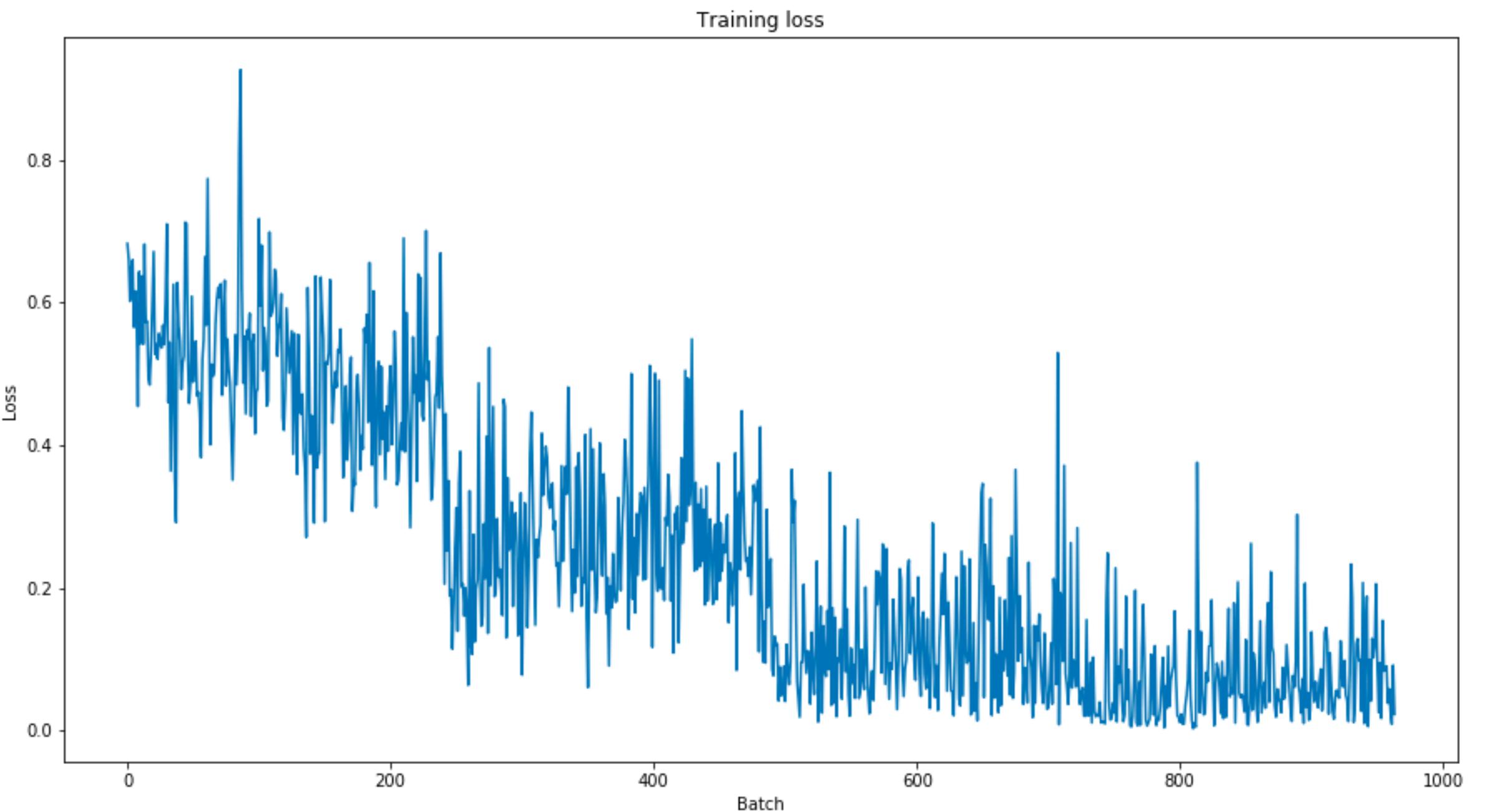


# BERT Fine-Tuning

“BERT Fine-Tuning Tutorial with PyTorch”

Chris McCormick

- [notebook](#) (load in Google Colab)
- [download](#) (add through your Google Drive)
- [leaderboard](#) (compare results)



# part 07: ethics and compliance

# AIF360: Fairness and Bias

## AI Fairness 360 Open Source Toolkit

- “OSCON, 2019: Removing Unfair Bias”

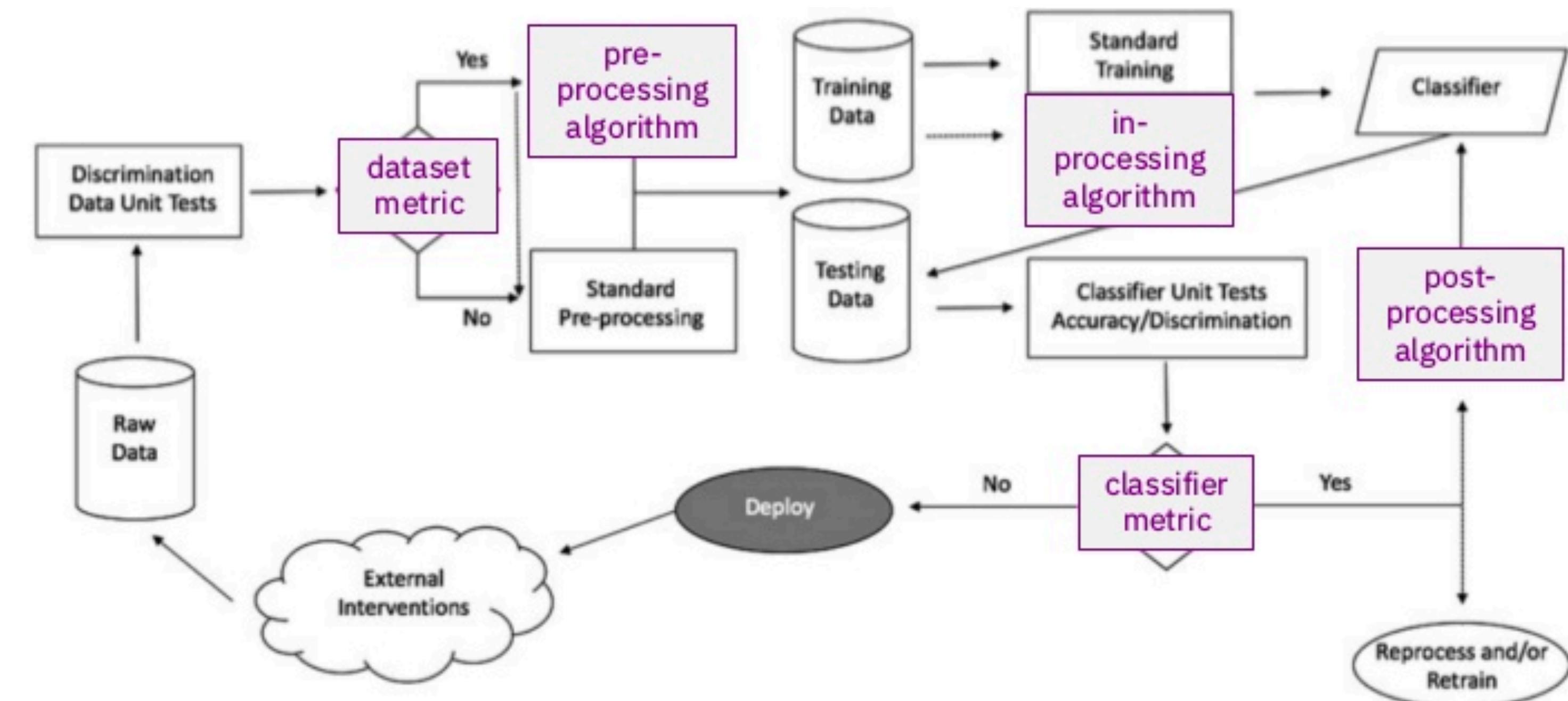
[part 1](#), [part 2](#)

Ana Echeverri, Trisha Mahoney

- “**AIF360 - Trusted and Fair AI**”

Animesh Singh

- [online demo](#)



# AllenNLP Interpret

## AllenNLP Interpret: A Framework for Explaining Predictions of NLP Models

- **interactive demo**: masked language model (BERT)
- **paper**
- **demo code**

### Masked Language Modeling

Enter text with one or more "[MASK]" tokens and BERT will generate the most likely token to substitute for each "[MASK]".

Sentence:

We're off to see the Wizard,  
the Wonderful Wizard of  
[MASK].

Mask 1 Predictions:

88.3%	Oz
2.6%	America
1.3%	London
1.0%	California
0.7%	England

### HotFlip Attack

HotFlip flips words in the input to change the model's prediction. We iteratively flip the input word with the highest gradient until the prediction changes.

Original Input: [CLS] We ' re off to see the Wizard , the Wonderful Wizard of [MASK] . [SEP]

Flipped Input: [CLS] We ' re off to see the Wizard , the Wonderful fortune of [MASK] . [SEP]

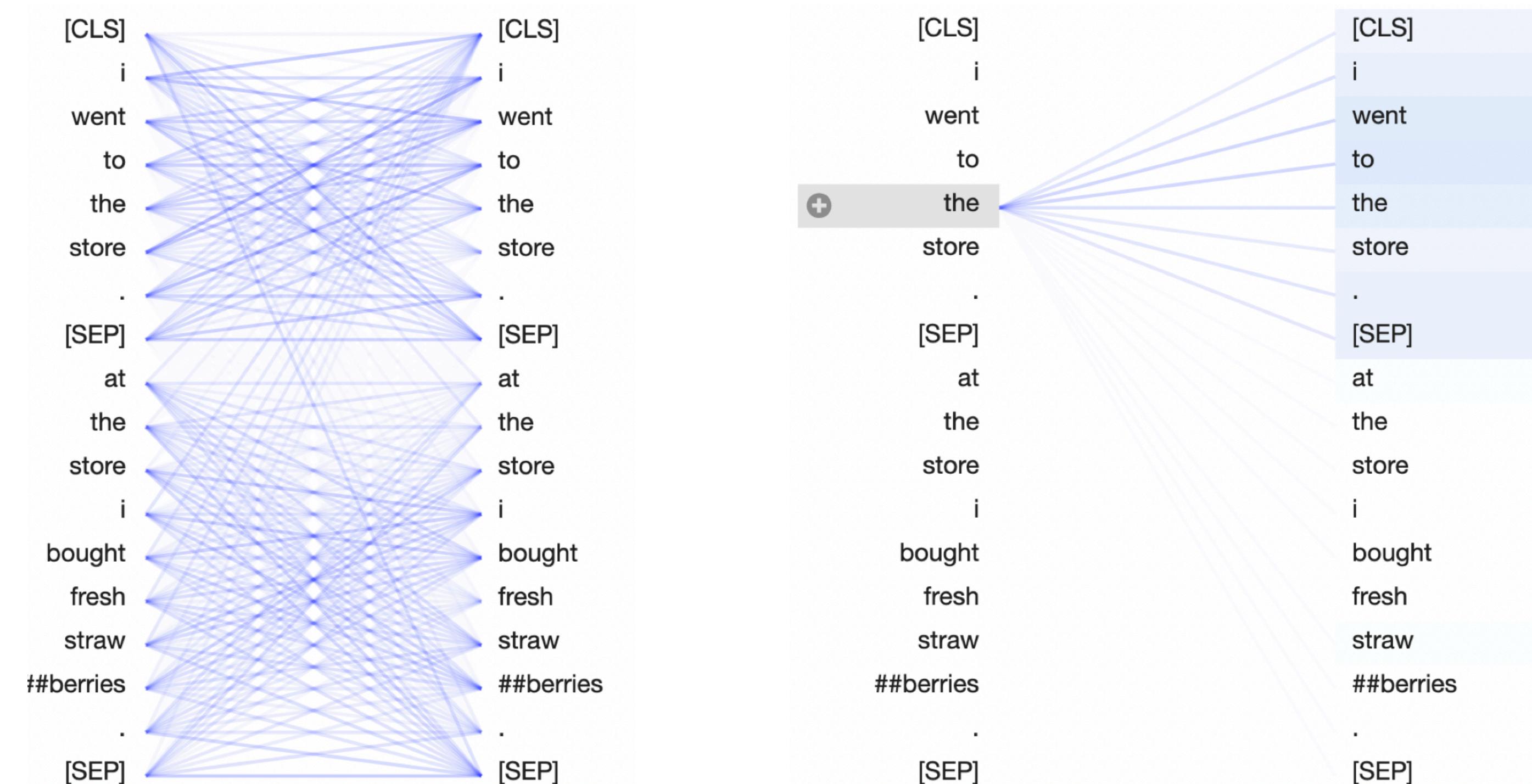
Prediction changed to: London

# Visualizing BERT

BertViz, Jesse Vig

Visualize BERT's self-attention layers on text classification tasks

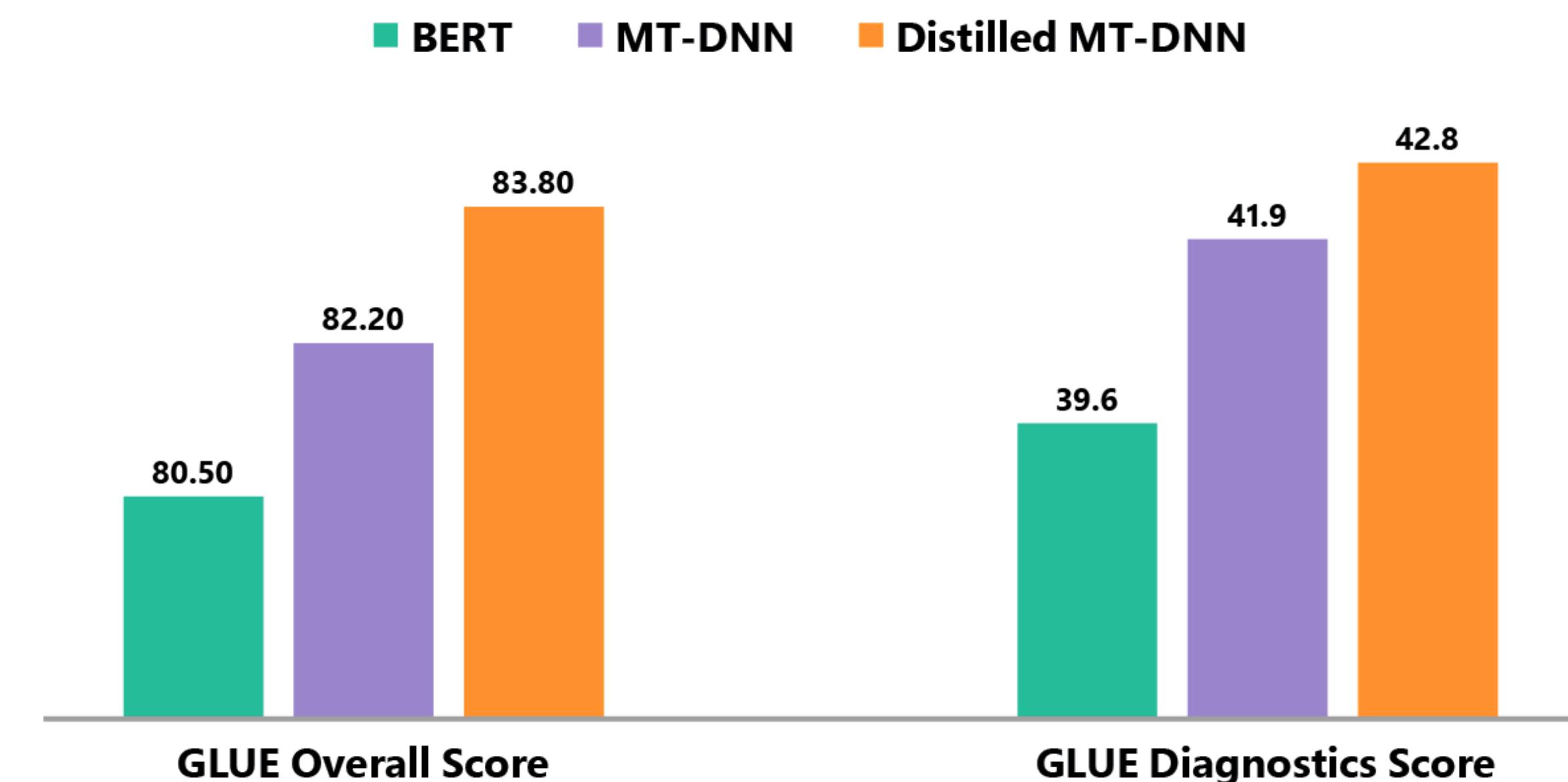
AI Explainability 360 Open Source Toolkit



# MT-DNN: Explainability for BERT

## Multi-Task Deep Neural Networks for Natural Language Understanding

- “Robust Language Representation Learning via Multi-task Knowledge Distillation”
- “Improving Multi-Task Deep Neural Networks via Knowledge Distillation for Natural Language Understanding”
- code repo



# part 08: hardware

# Hardware in perspective

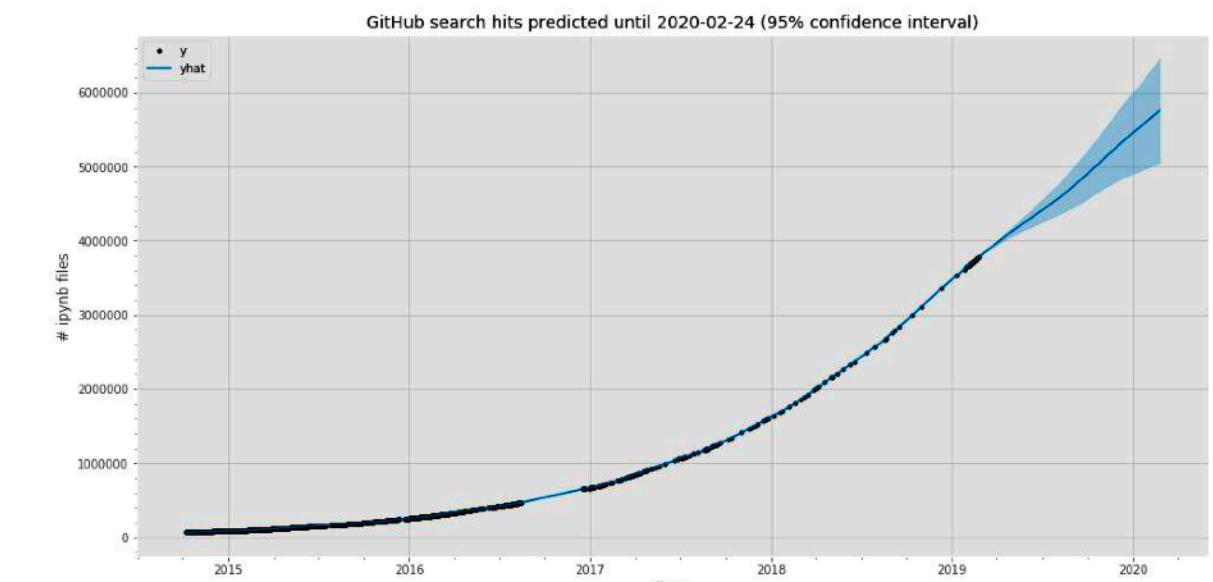
An **emerging trend** disrupts the past 15-20 years of software engineering practice:

**hardware > software > process**

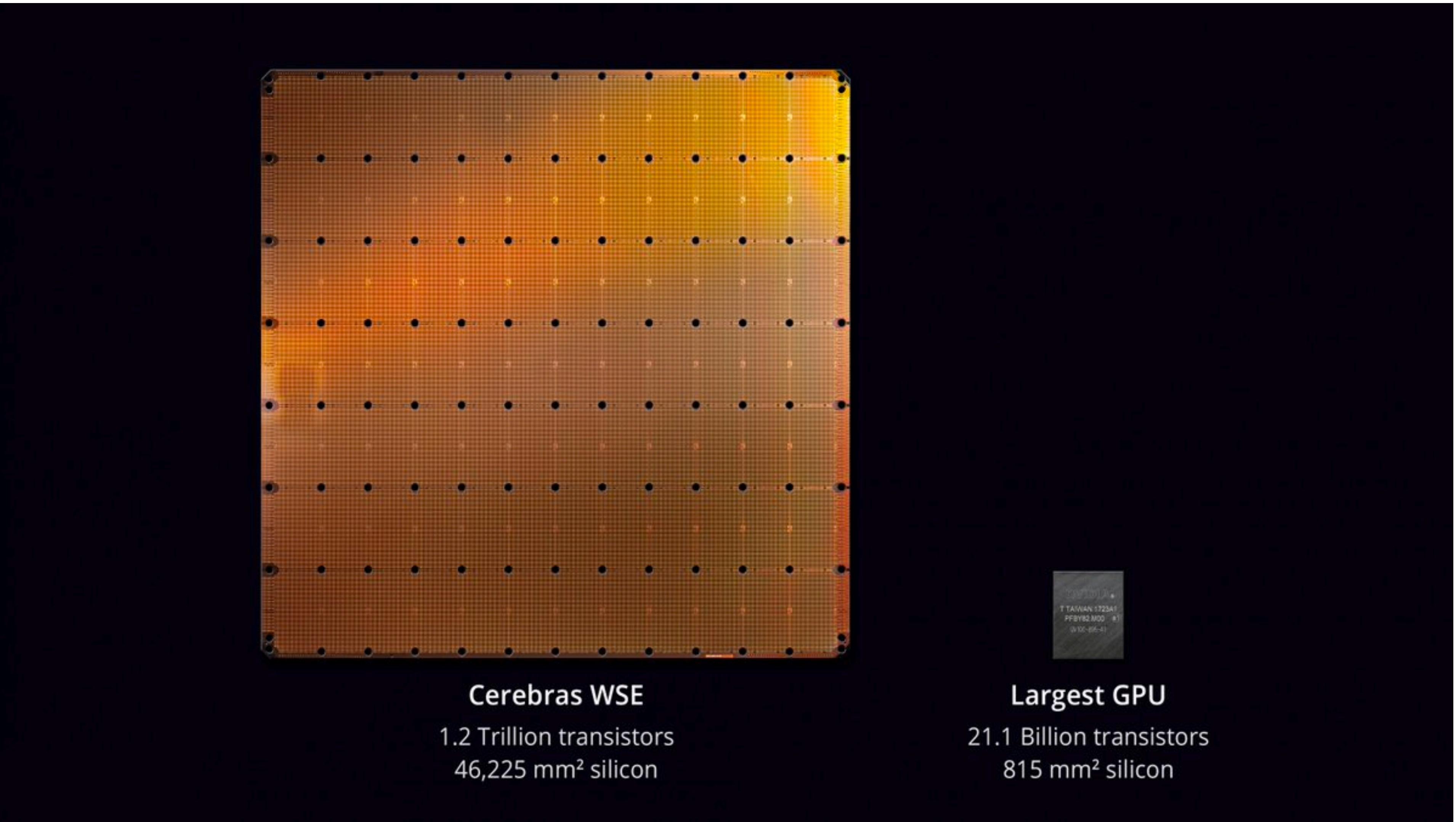
Hardware is now evolving more rapidly than software, which is evolving more rapidly than effective process

**Moore's Law** is all but dead, although ironically many inefficiencies had been based on it

**Project Jupyter**, **Apache Arrow**, **NumPyWren** and the related **Ray** are emblematic for data infrastructure transformation in enterprise



# ASICs the size of iPads, for ML

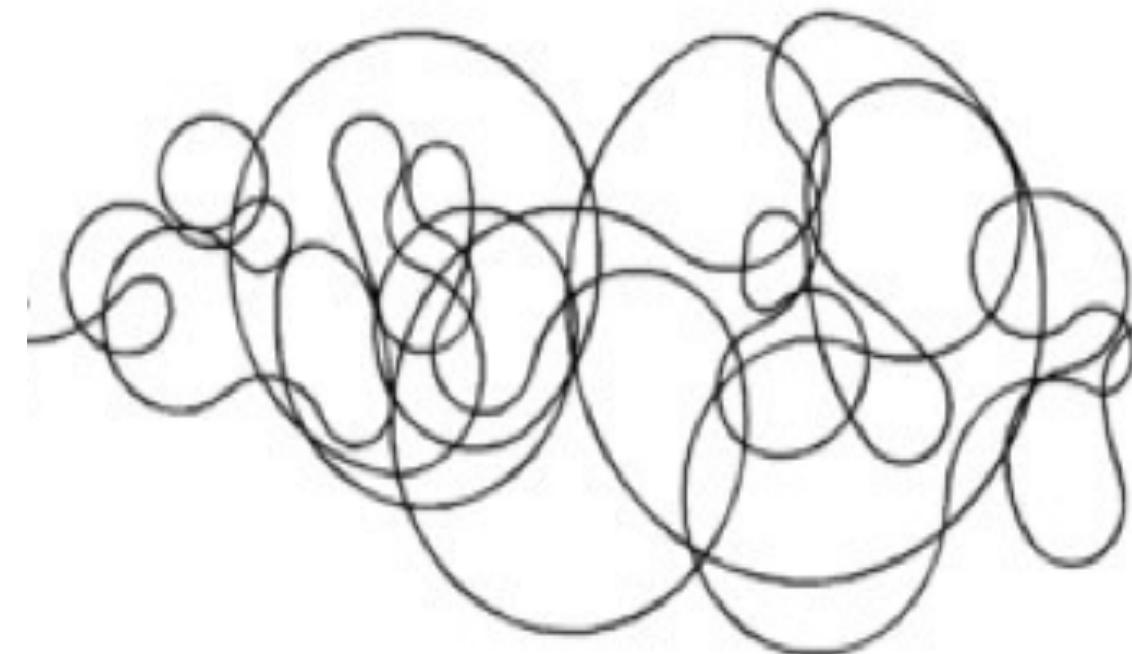


# Cluster topologies, by generation

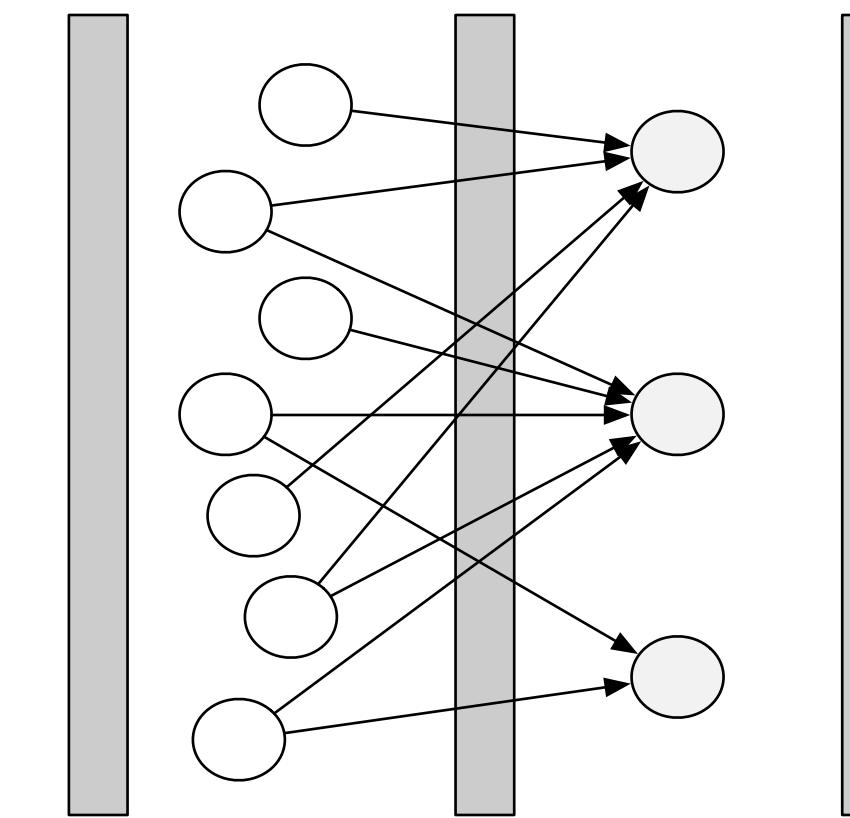
1. MPI was too complex for reuse and scale of communities across both academia and industry (since 1992)
2. Batch, i.e., Hadoop, Spark, and their communities scaled (2006-2016); however their topologies were often too overdetermined for the more valuable use cases in ML – meanwhile, hardware assumptions changed significantly
3. More flexible compute topologies became prioritized:
  - “**Taming Latency Variability and Scaling Deep Learning**”  
**Jeff Dean** Google (2013-11-10) *pre-Tensorflow*
  - UC Berkeley RISE Lab: **Ray** and related work, now turning out to be even more general purpose than TF



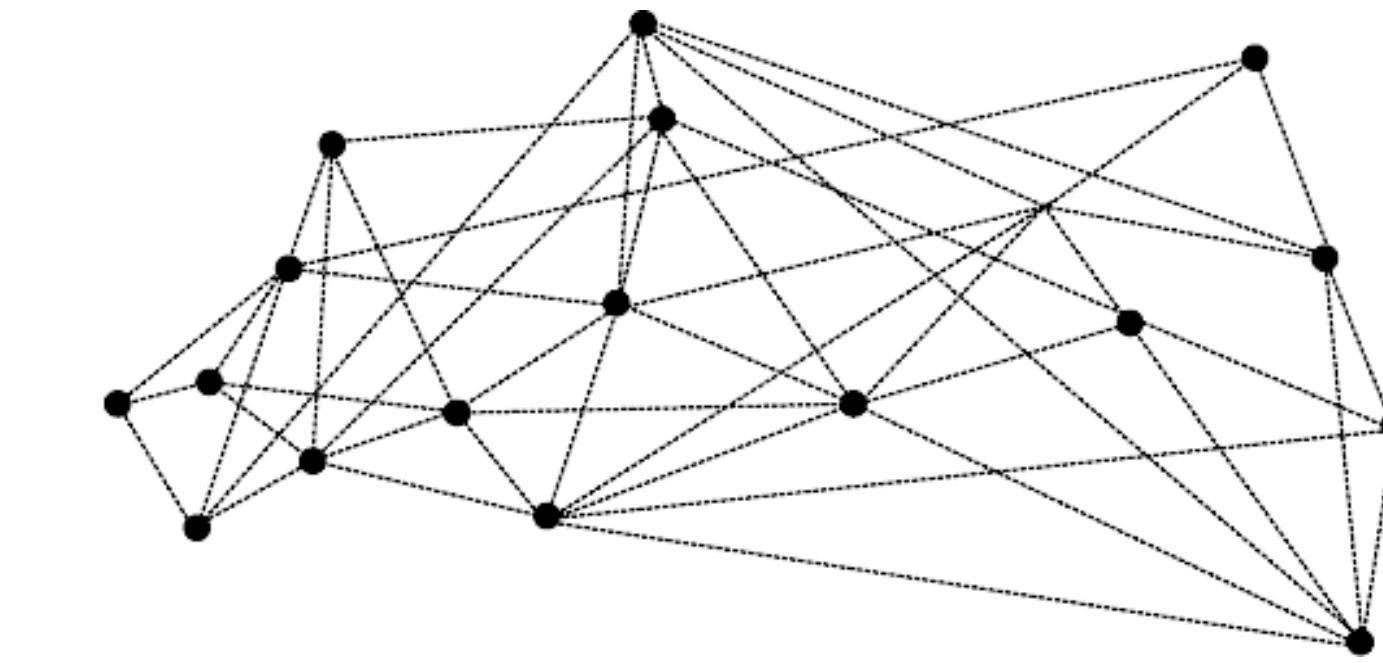
# Cluster topologies, by generation



1990s



mid-2000s



current

Opinion: one problem with software/hardware interface for distributed systems is that it's taken *decades* to prioritize the need for handling **graphs/tensors** directly in popular, accessible open source libraries, without having some **commercial database vendor** intermediate.

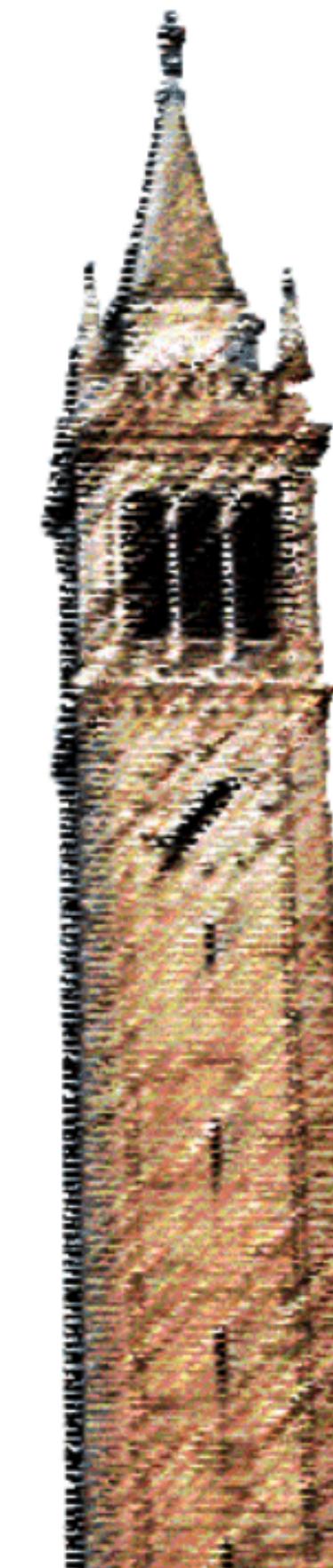
# Evolution of cloud patterns

UC Berkeley published a [2009 report](#) about early use cases for cloud computing, which foresaw the shape of industry deployments over much of the next decade, and led directly to [Apache Mesos](#) and [Apache Spark](#)

It's fascinating to study the **contrasts** between that 2009 report and its 2019 follow-up.

(minor footnote: [vimeo.com/3616394](https://vimeo.com/3616394))

Above the Clouds: A Berkeley View of Cloud Computing



*Michael Armbrust  
Armando Fox  
Rean Griffith  
Anthony D. Joseph  
Randy H. Katz  
Andrew Konwinski  
Gunho Lee  
David A. Patterson  
Ariel Rabkin  
Ion Stoica  
Matei Zaharia*

Electrical Engineering and Computer Sciences  
University of California at Berkeley

Technical Report No. UCB/EECS-2009-28  
<http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html>

February 10, 2009

2009

# Evolution of cloud patterns

Early patterns of cloud use mirrored the **virtualization** familiar to enterprise firms in their existing on-prem architectures, mostly as a convenience – *for migration*

More **contemporary patterns** force restructuring – *for efficiency and security* – **decoupling computation and storage**

Cloud Programming Simplified: A Berkeley View on  
Serverless Computing



Eric Jonas  
Johann Schleier-Smith  
Vikram Sreekanti  
Chia-Che Tsai  
Anurag Khandelwal  
Qifan Pu  
Vaishaal Shankar  
Joao Menezes Carreira  
Karl Krauth  
Neeraja Yadwadkar  
Joseph Gonzalez  
Raluca Ada Popa  
Ion Stoica  
David A. Patterson

Electrical Engineering and Computer Sciences  
University of California at Berkeley

Technical Report No. UCB/EECS-2019-3  
<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-3.html>

February 10, 2019

# Key takeaways from “A Berkeley View”

- physics + cloud economics imply less “framework” layers
- service offerings **migrate** up the tech stack
- issues with **data gravity**, increased **vendor lock-in**

# Impact of edge devices and low-power ML

- **Episode 3, Domino:** massive latent hardware  
(2018-11-07)
- **Alasdair Allan, Babilim Light Industries:**  
**The intelligent edge and the demise of big data?**  
(2019-05-02)
- **Pete Warden, Google:**  
**TensorFlow.js on low-power devices**  
(2018-10-10)

# Hardware Implications

processor redesign:

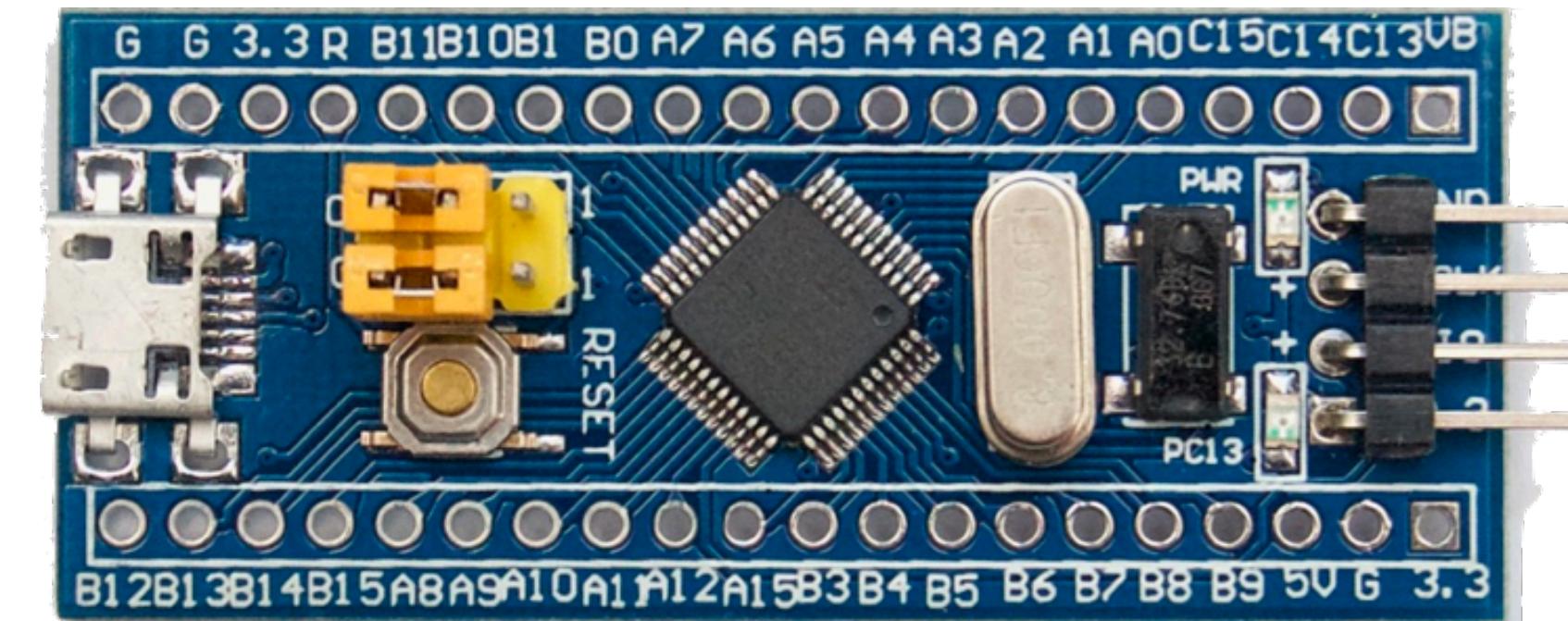
- $10^2$  more compute units
- $10^5$  faster memory
- $10^3$  faster compute

model compression:

- $10^2$  more compute
- $10^4$  less energy

edge inference:

- multi-modal perception (e.g., CV + speech)
- no battery, PV on ambient light
- low price per unit



# part 09: energy and policy concerns

# Problem: AI-based Climate Change

The current process of training **deep learning** models generates exceptionally large quantities of atmospheric carbon and heat...

arXiv.org > cs > arXiv:1906.02243

Computer Science > Computation and Language

## Energy and Policy Considerations for Deep Learning in NLP

Emma Strubell, Ananya Ganesh, Andrew McCallum

(Submitted on 5 Jun 2019)

Recent progress in hardware and methodology for training neural networks has ushered in a new generation of large neural models. These models have obtained notable gains in accuracy across many NLP tasks. However, these accuracy improvements come at a significant cost. These models are costly to train and develop, both financially, due to the cost of hardware and electricity or cloud compute time, and environmentally, due to the carbon footprint required to fuel modern tensor processing hardware. In this paper we bring attention of NLP researchers by quantifying the approximate financial and environmental costs of training a variety of neural network models for NLP. Based on these findings, we propose actionable recommendations to reduce costs and environmental impact in NLP research and practice.

Comments: In the 57th Annual Meeting of the Association for Computational Linguistics (ACL). Florence, Italy. July 2019

Subjects: Computation and Language (cs.CL)

Cite as: arXiv:1906.02243 [cs.CL]

(or arXiv:1906.02243v1 [cs.CL] for this version)

# Carbon Emissions

[arxiv.org/abs/1906.02243](https://arxiv.org/abs/1906.02243)

Consumption	CO <sub>2</sub> e (lbs)
Air travel, 1 passenger, NY↔SF	1984
Human life, avg, 1 year	11,023
American life, avg, 1 year	36,156
Car, avg incl. fuel, 1 lifetime	126,000
Training one model (GPU)	
NLP pipeline (parsing, SRL)	39
w/ tuning & experimentation	78,468
Transformer (big)	192
w/ neural architecture search	626,155

Table 1: Estimated CO<sub>2</sub> emissions from training common NLP models, compared to familiar consumption.<sup>1</sup>

Consumer	Renew.	Gas	Coal	Nuc.
China	22%	3%	65%	4%
Germany	40%	7%	38%	13%
United States	17%	35%	27%	19%
Amazon-AWS	17%	24%	30%	26%
Google	56%	14%	15%	10%
Microsoft	32%	23%	31%	10%

Table 2: Percent energy sourced from: Renewable (e.g. hydro, solar, wind), natural gas, coal and nuclear for the top 3 cloud compute providers (Cook et al., 2017), compared to the United States,<sup>4</sup> China<sup>5</sup> and Germany (Burger, 2019).

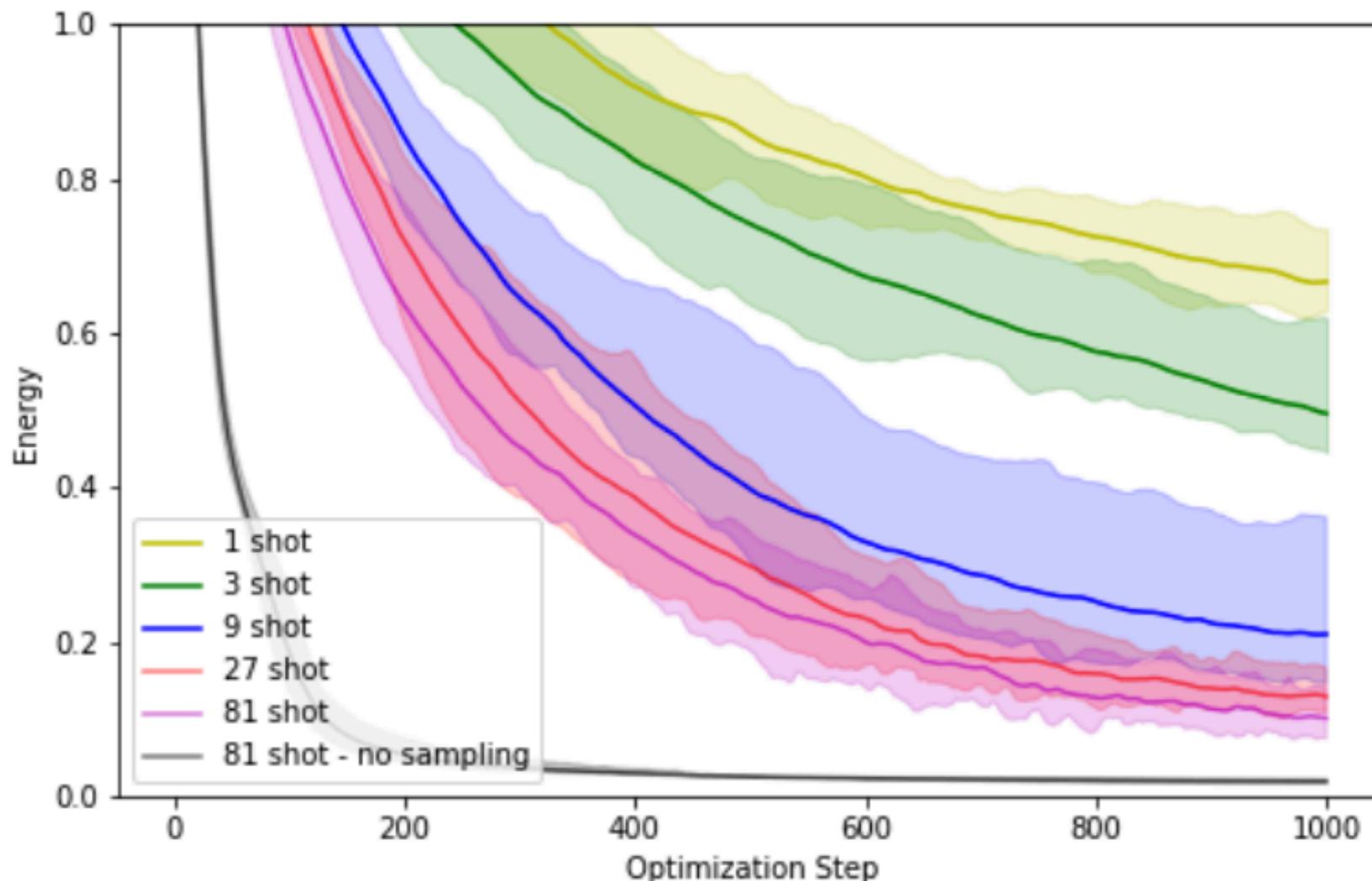
NLP model emits ~5x car (incl. fuel)

# But the Algorithms...

However, most of the computation required to train deep learning models relies on variants of **stochastic gradient descent**.

Fortunately these algorithms can be performed with *superlinear* (read: nearly exponential) performance increase on quantum vectorization devices.

<https://arxiv.org/abs/1910.01155>



---

**Algorithm 3**  $n$ -shot QSGD with Hamiltonian sampling

---

```
1: Given  $U(\theta)$  satisfying a  $K$ -term parameter shift rule, with  $\theta \in \mathbb{R}^d$ , and  $\mathcal{L}(\theta) = \langle H \rangle_\theta$  where  $H = \sum_{j=1}^M H_j$ 
2: Set initial circuit parameters  $\theta^{(0)}$  and learning rate  $\alpha$ 
3:  $t \leftarrow 0$ 
4: while  $t < T$  do
5:   for all  $1 \leq i \leq d$  do
6:     Sample  $j \in \{1, \dots, M\}$  with  $\text{prob}(j) = 1/M$ 
7:     for all  $1 \leq k \leq K$  do
8:       Evaluate  $\tilde{h}_j^{(n)}(\theta_{k,i}^{(t)})$ 
9:     end for
10:     $g_i(\theta^{(t)}) \leftarrow \sum_{k=1}^K M \gamma_{k,i} \tilde{h}_j^{(n)}(\theta_{k,i}^{(t)})$ 
11:     $\theta_i^{(t+1)} \leftarrow \theta_i^{(t)} - \alpha g_i(\theta^{(t)})$ 
12:  end for
13:   $t \leftarrow t + 1$ 
14: end while
```

▷ Iterate through optimization steps  
▷ Iterate through circuit parameters  
▷ Sample a Hamiltonian term  
▷ Iterate through parameter shift terms  
▷ Construct estimator for  $\langle h_j \rangle_{\theta_{k,i}^{(t)}}$  via  $n$  measurements  
▷ Construct estimator for  $\partial \langle H \rangle_{\theta^{(t)}} / \partial \theta_i$   
▷ Update parameters

# Something something quantum

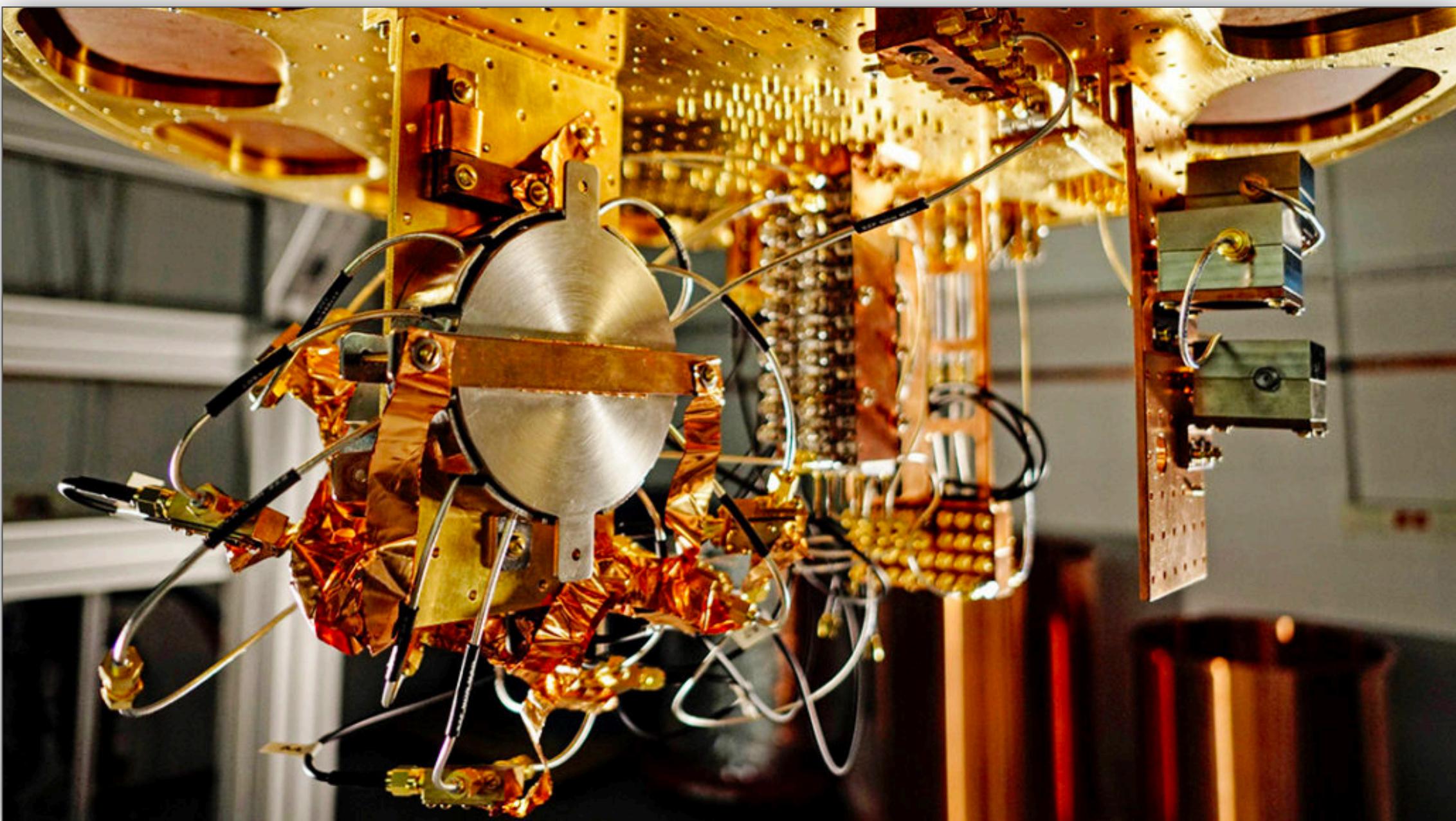
However, **quantum devices** tend to require:

- lots of power
- extreme cooling

Which brings us back to the general problem of cloud computing, etc.

**Google: “quantum supremacy”**

**IBM: “quantum advantage”**

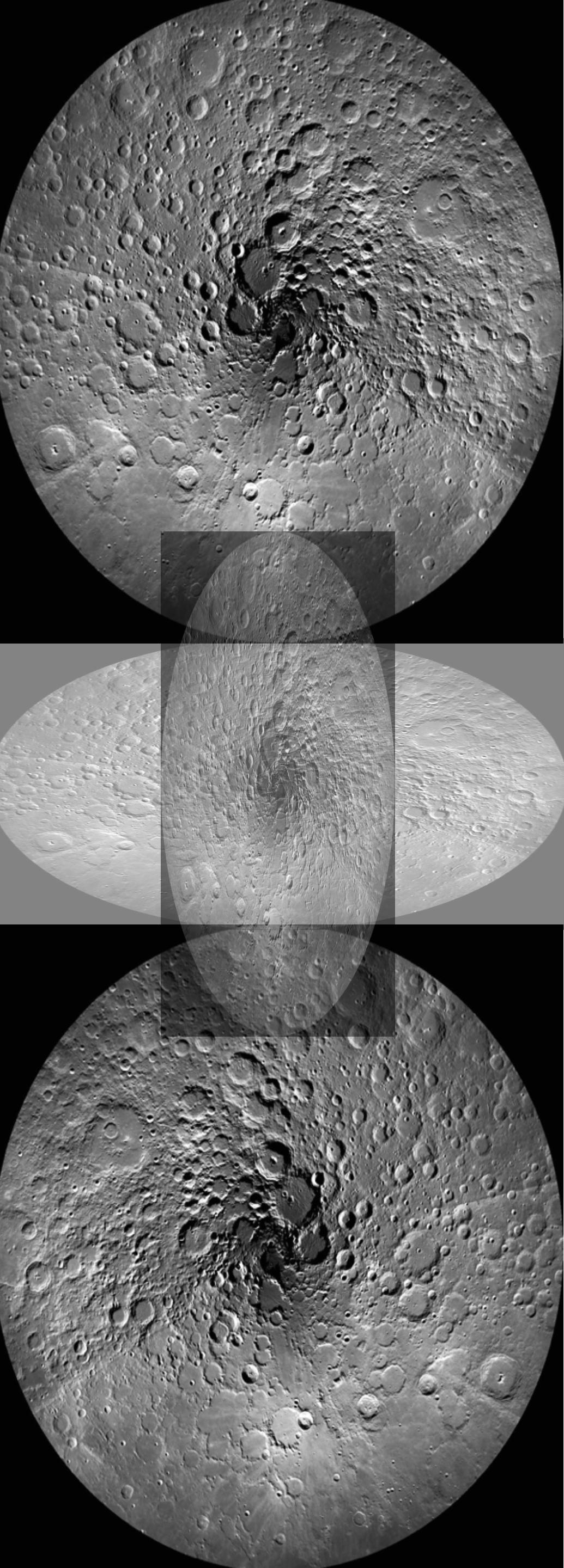


# Eus Keus?

Hence, we might consider migrating those extreme cloud computing needs to **The Moon** ?!?

Craters in the South Polar region offer the coldest points in our solar system (inside craters) plus perpetual sunlight (craters rides) for solar power.

Then run heat pumps out under the surface regolith.



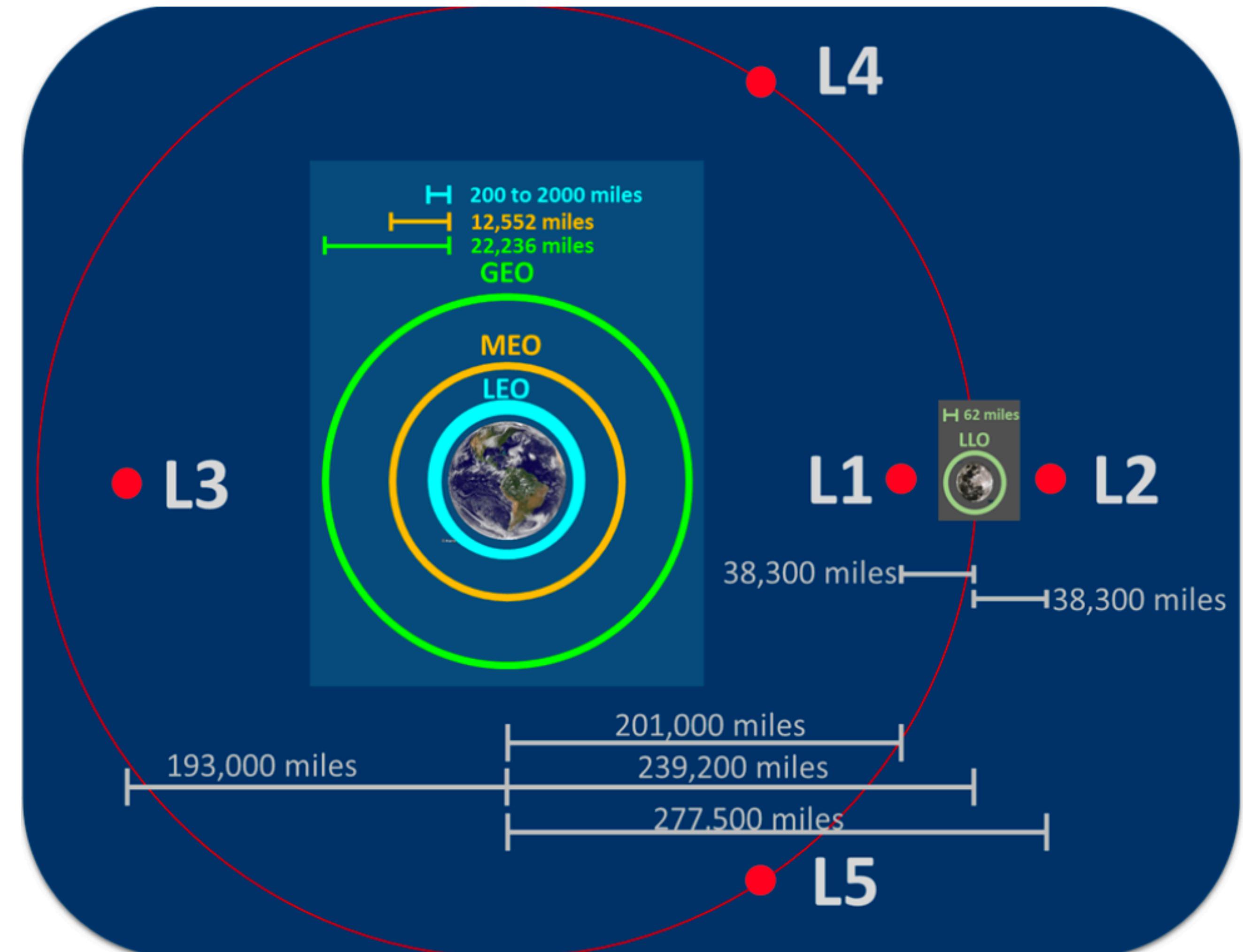
# Cislunar System

*Big Data to the rescue...*

*Train ginormous ML models in quantum datacenters located on The Moon using batch operations, serialize them, then transmit back to the Earth – to support transfer learning use cases.*

*We can place routers on a supply station at the L1 point.*

*In any case, we'll need to fly a soletta out to Lagrange points soon enough.*



# Because Fashion Choices

Two words: “moon busses”

We'd get to wear incredibly cool outfits and hair styles.

Plus we would spend much time Underground, which should be rather familiar to Londoners.



# Balancing the Budget

A former CTO of NASA (and devout rocket scientist) estimates the costs for establishing a permanent Lunar Base to be < \$15 billion.

Frankly, the fabrication plants built by Intel recently in the US and Israel cost approximately that.

Home > Semiconductors

## Intel Considers \$11 Billion Fab in Israel

by Anton Shilov on January 29, 2019 4:00 PM EST

Posted in Semiconductors, CPUs, Intel, SoC

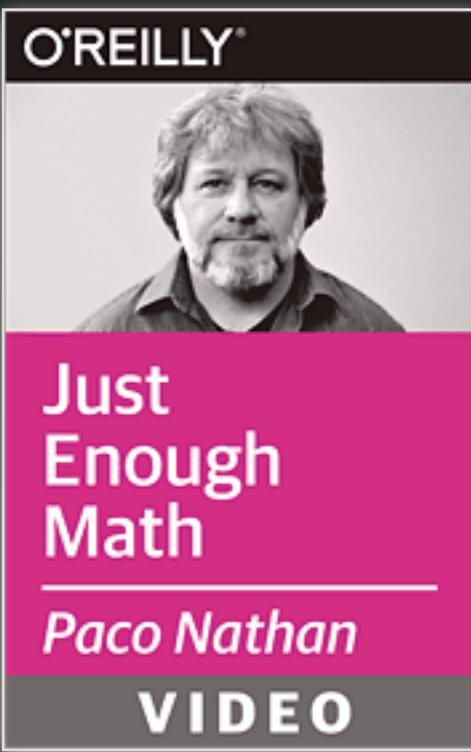
37 Comments

+ Add A Comment

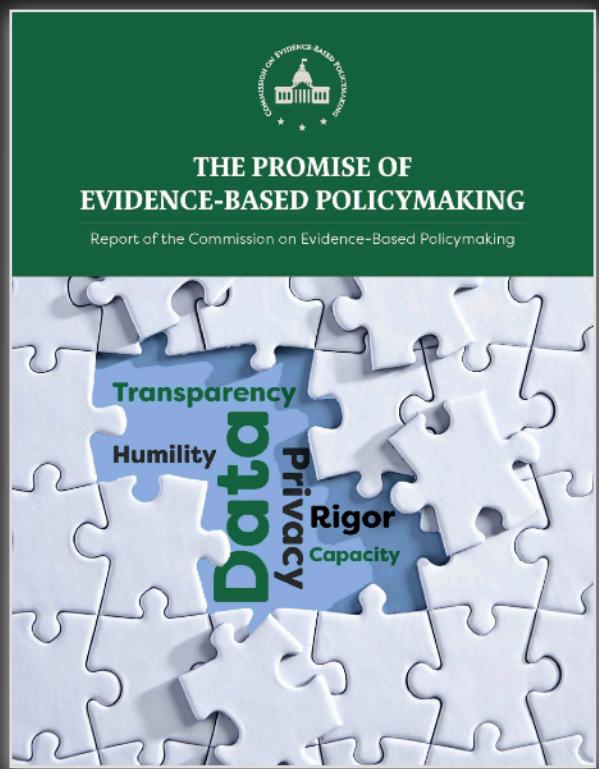


# publications, interviews, conference summaries...

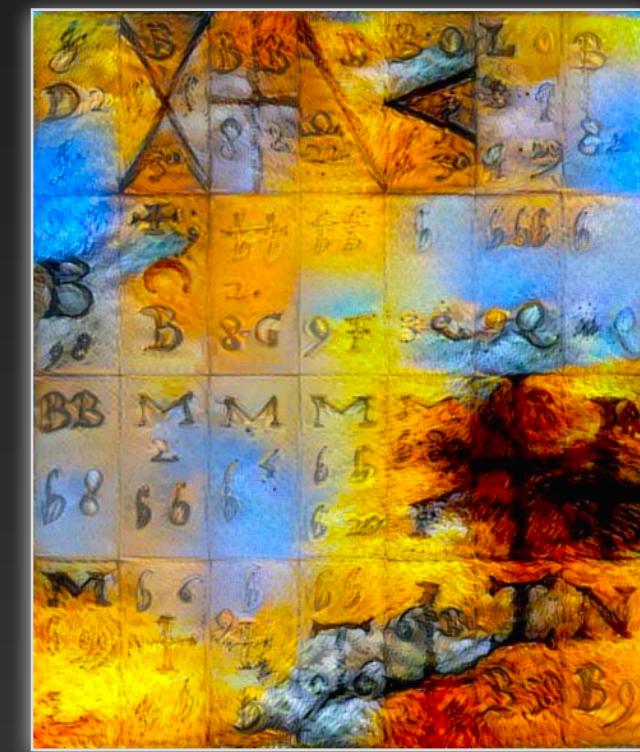
[@pacoid](https://derwen.ai/paco)



Just Enough Math



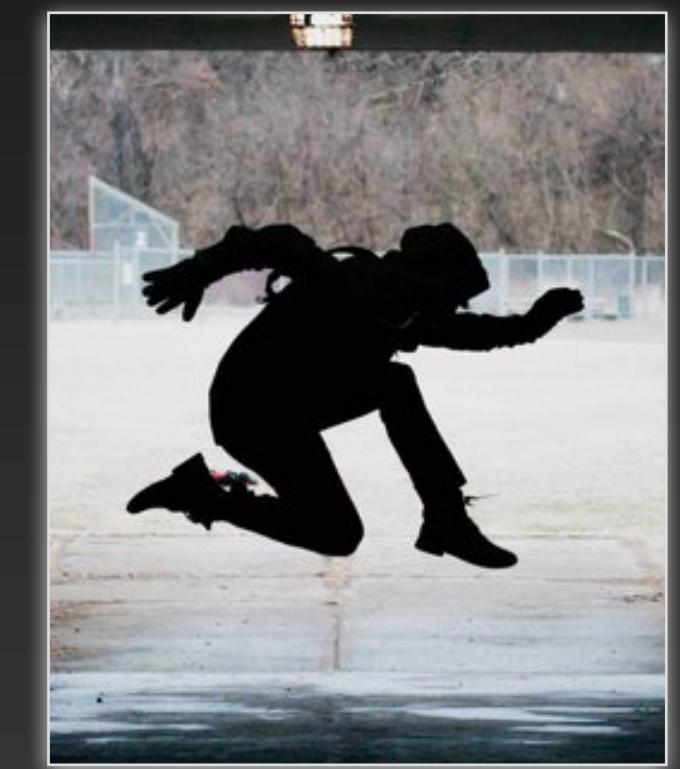
Rich Context



Hylbert-Speys



Rev conf



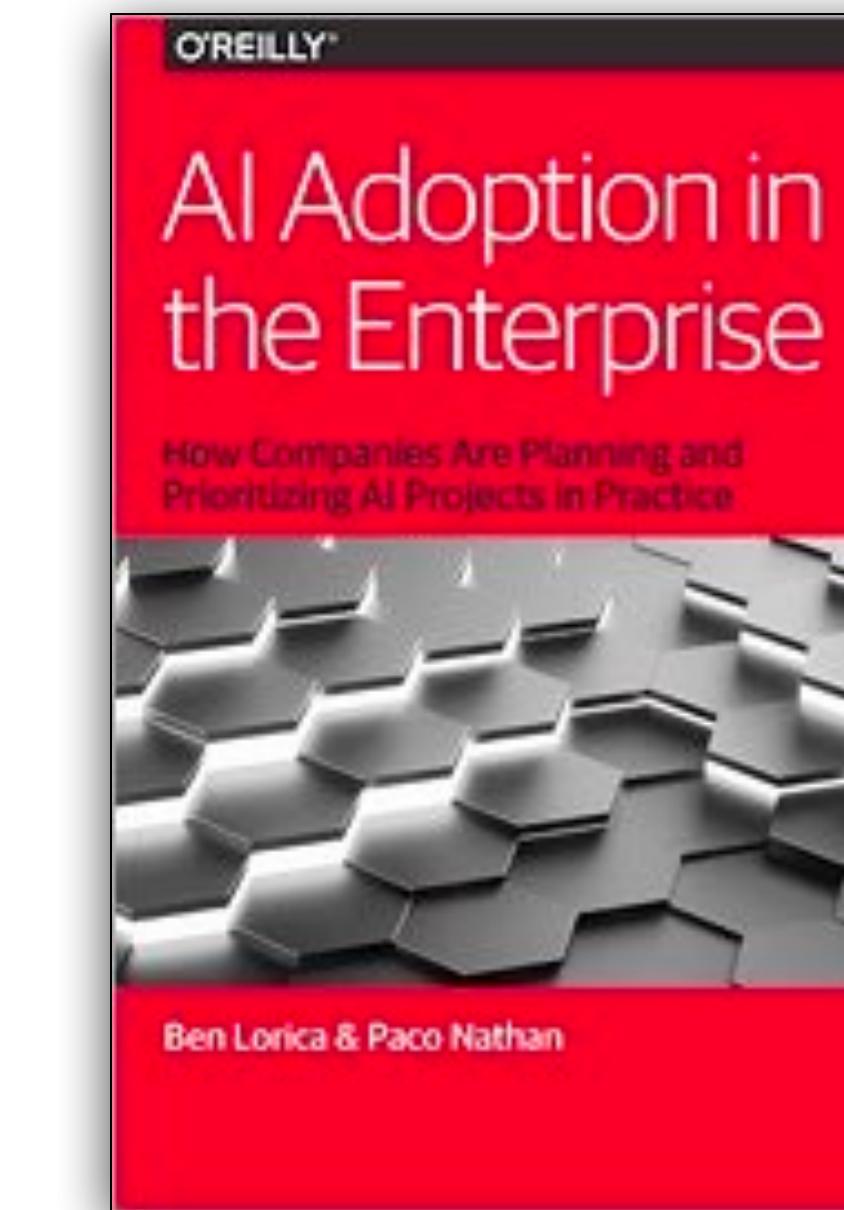
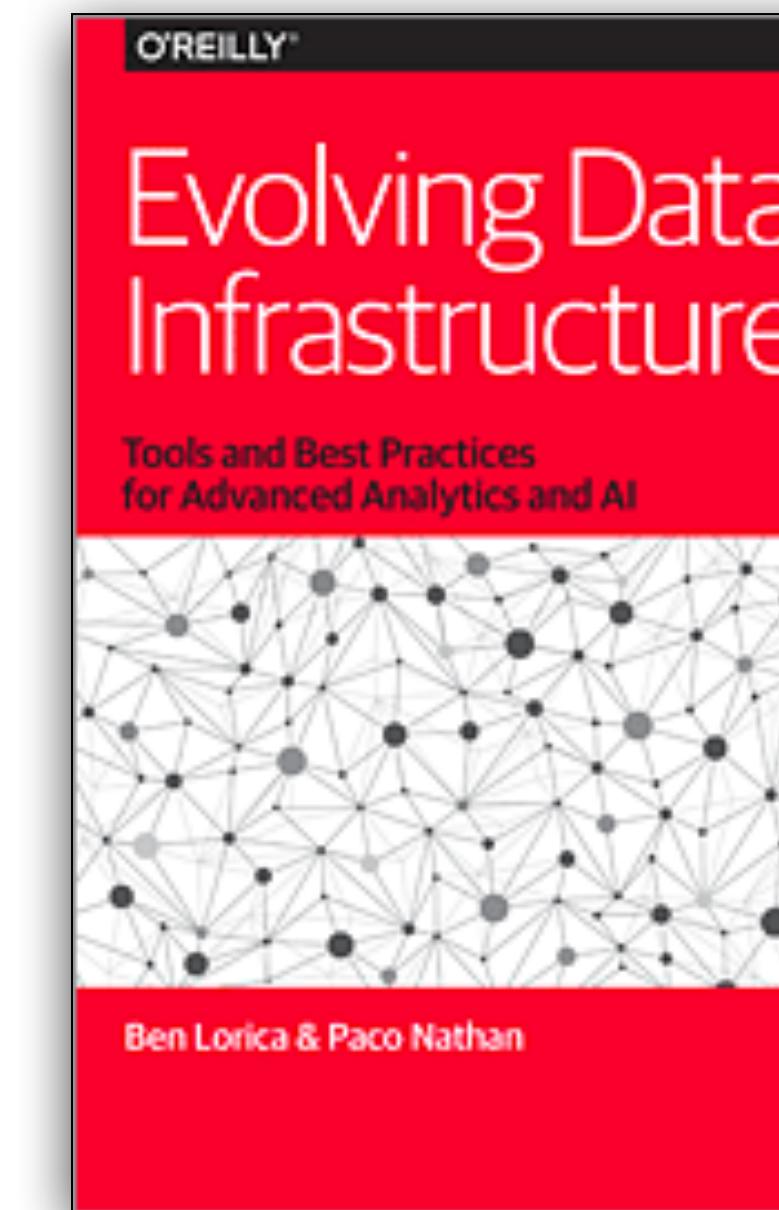
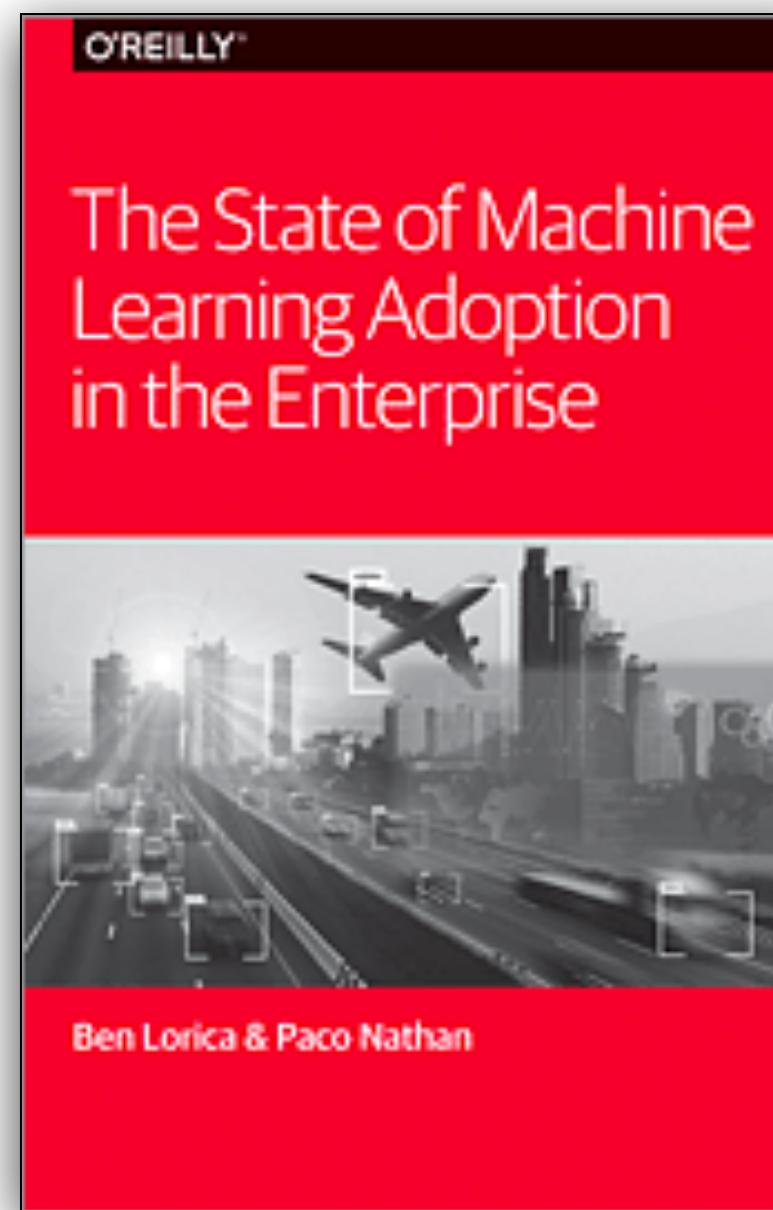
Themes + Confs  
per Pacoid



[derwen.ai](https://derwen.ai)

# Industry surveys for AI and Cloud adoption

- “Three surveys of AI adoption reveal key advice from more mature practices”  
**Ben Lorica, Paco Nathan O'Reilly Media (2019-02-20)**
- Episode 7, *Domino*: surveying “ABC” adoption in enterprise  
(2019-03-03)

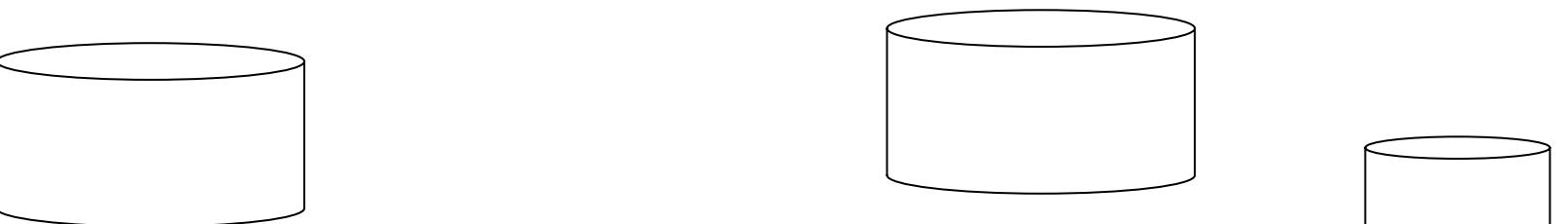
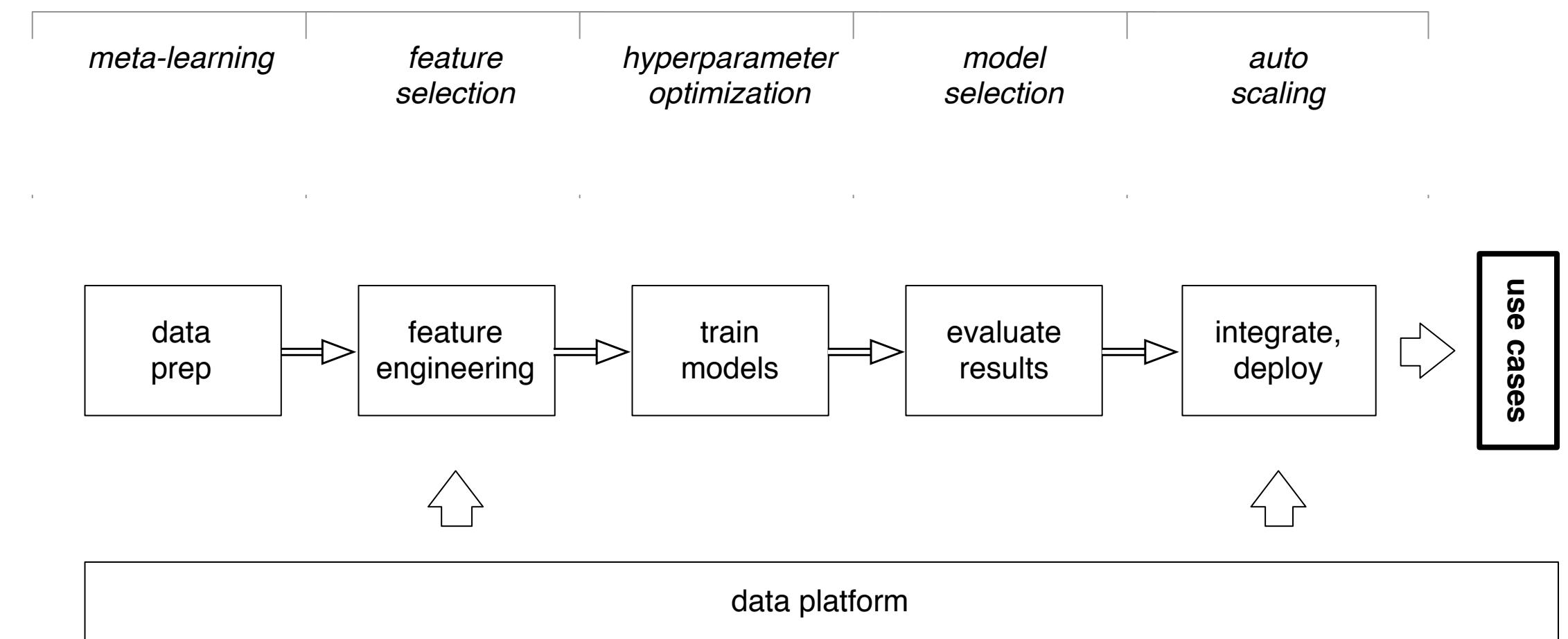


# Unpacking AutoML



[derwen.ai/s/yvkg](https://derwen.ai/s/yvkg)

*we noted an uptick in adoption for  
a third aspect, co-evolving along  
with DG and MLOps*

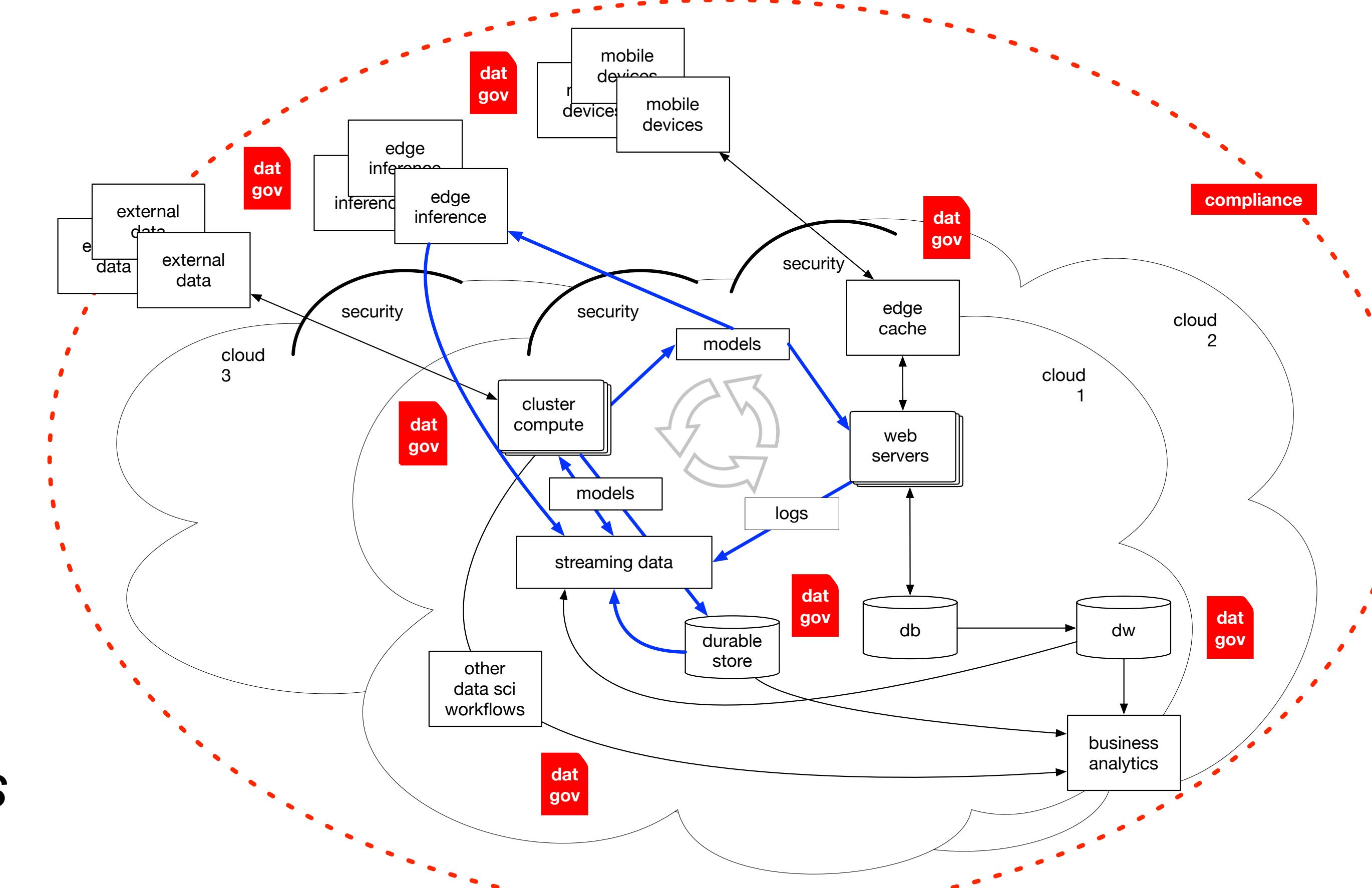


# Overview of Data Governance



[derwen.ai/s/6fqt](https://derwen.ai/s/6fqt)

*we noted a resurgence in data governance – this report examines key themes, vendors, issues, etc.*



# Active Learning as a data strategy



[derwen.ai/s/d8b7](https://derwen.ai/s/d8b7)

*teams of people + machines,  
leveraging the complementary  
strengths of both*

