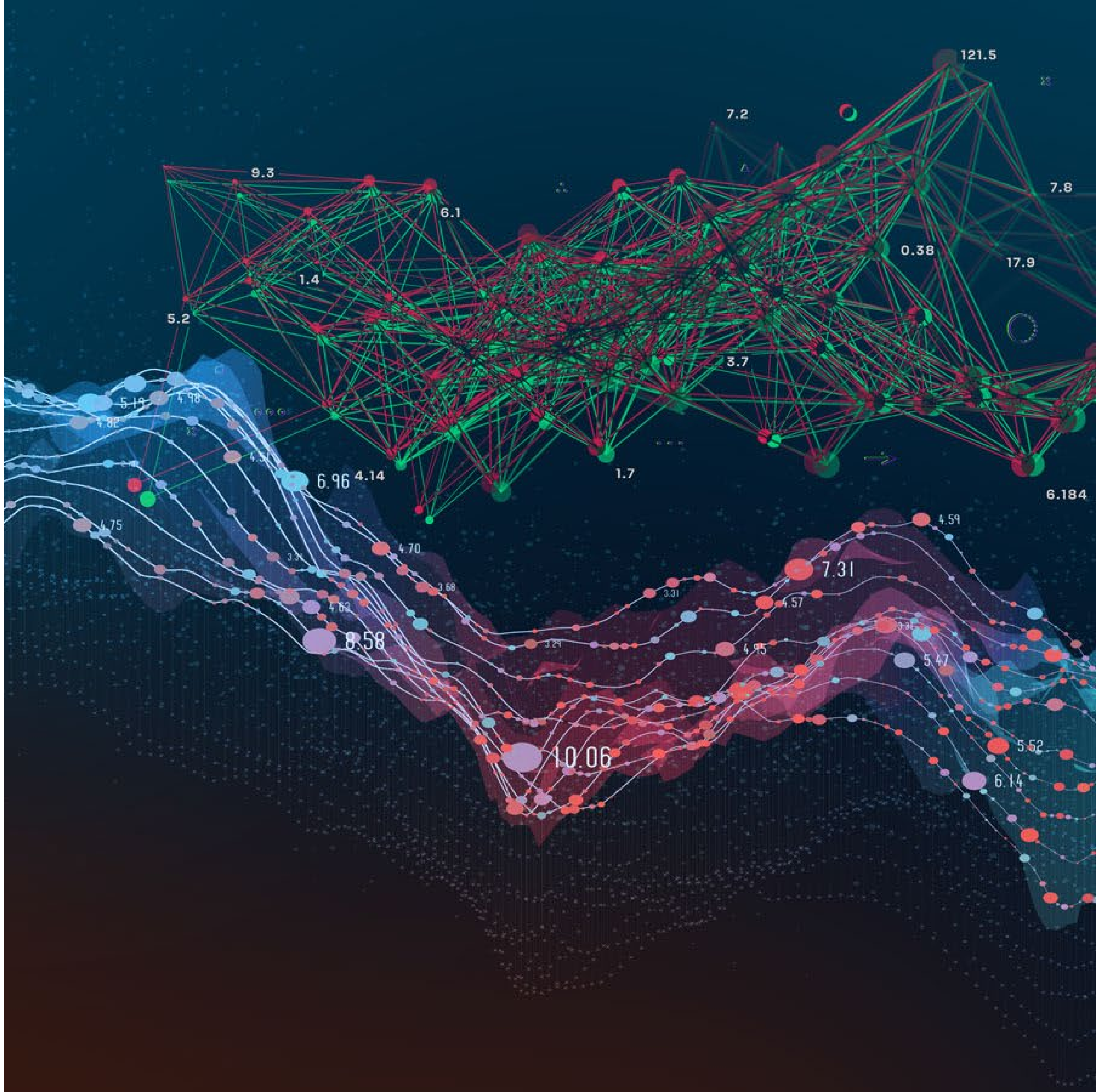
Pacific Northwest
NATIONAL LABORATORY

Text Analysis: Introduction to Advanced Language Modeling

Karl Pazdernik, Ph.D.
Pacific Northwest National Laboratory
North Carolina State University



Pacific Northwest NATIONAL LABORATORY

Welcome! Introductions

Karl Pazdernik

Data Scientist

PhD Statistics (ISU)

- Natural Language Processing
- Model Calibration and Uncertainty Quantification

Credit to PNNL colleagues for aiding in course content:

- Robin Cosbey
- Addie Kingsland
- Liz Cary
- Ryan Butner

Course Objectives

- Understand **core ideas** in Natural Language Processing (NLP)
  - Intricacies of textual data and NLP
  - Common NLP tasks
- Gain familiarity with common **NLP tools**
  - NLTK, Spacy, StanfordNLP, AllenNLP, HuggingFace
  - Be able to run code in practical examples
- Learn about **state-of-the-art**
  - What's easy/possible/impossible in NLP?
  - Common models/methods
  - Open problems/concerns

Agenda

Morning

- Text as a Modality
- Common NLP Tasks
- Ethics & Bias
- Preprocessing Basics

Afternoon

- Available Tools
- Text Representations
- Modeling
- Transformers

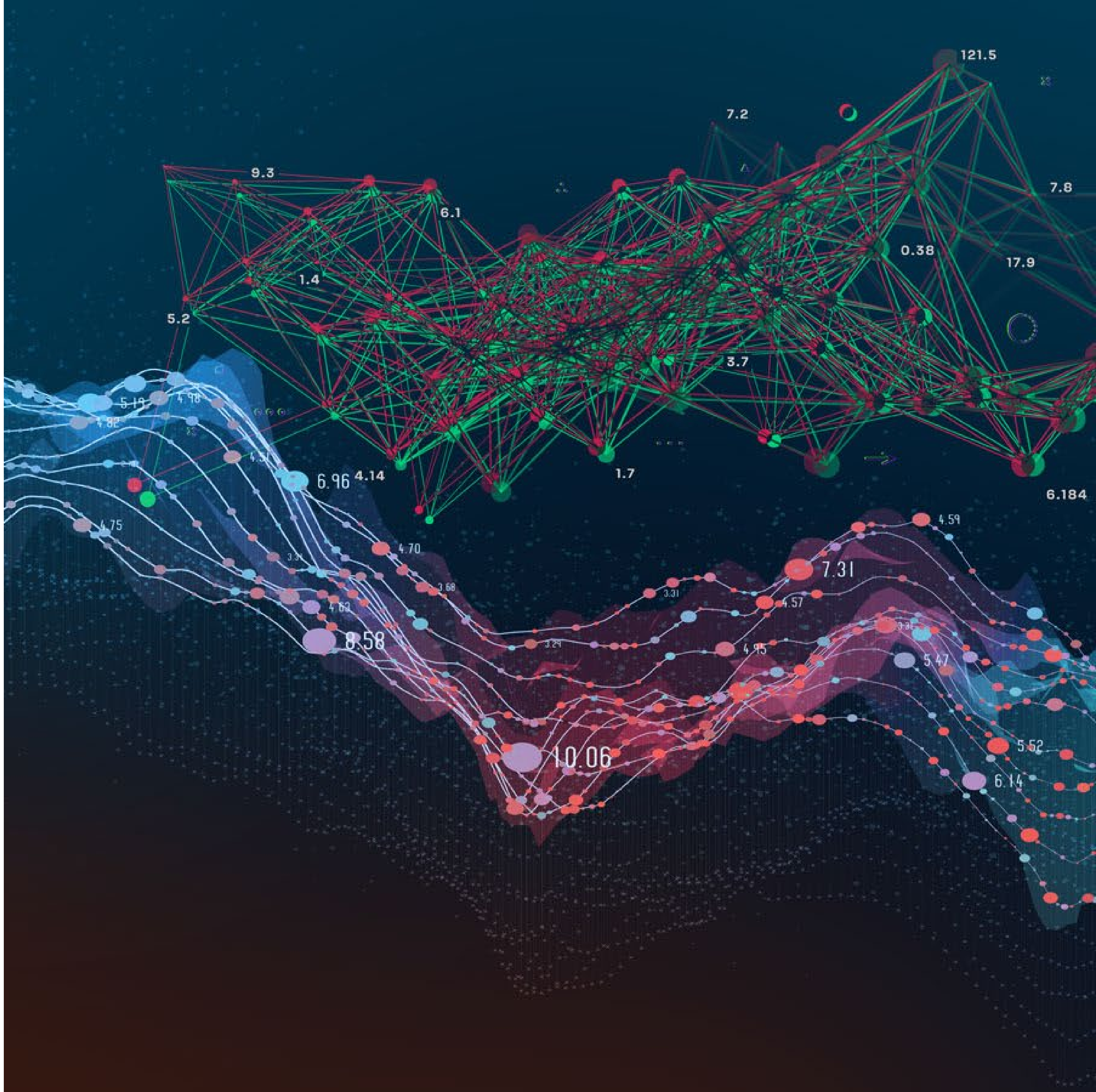
Quick Disclaimer

- This workshop incorporates many English language examples
  - Most work in NLP has traditionally focused on English to the exclusion of other languages

- Many aspects of NLP are ***highly language dependent***
  - Methods and techniques designed for English may not function properly
  - Most languages are "low resource", meaning there are no large stockpiles of well-annotated data that can be used for training

Pacific Northwest
NATIONAL LABORATORY

Text as a Modality

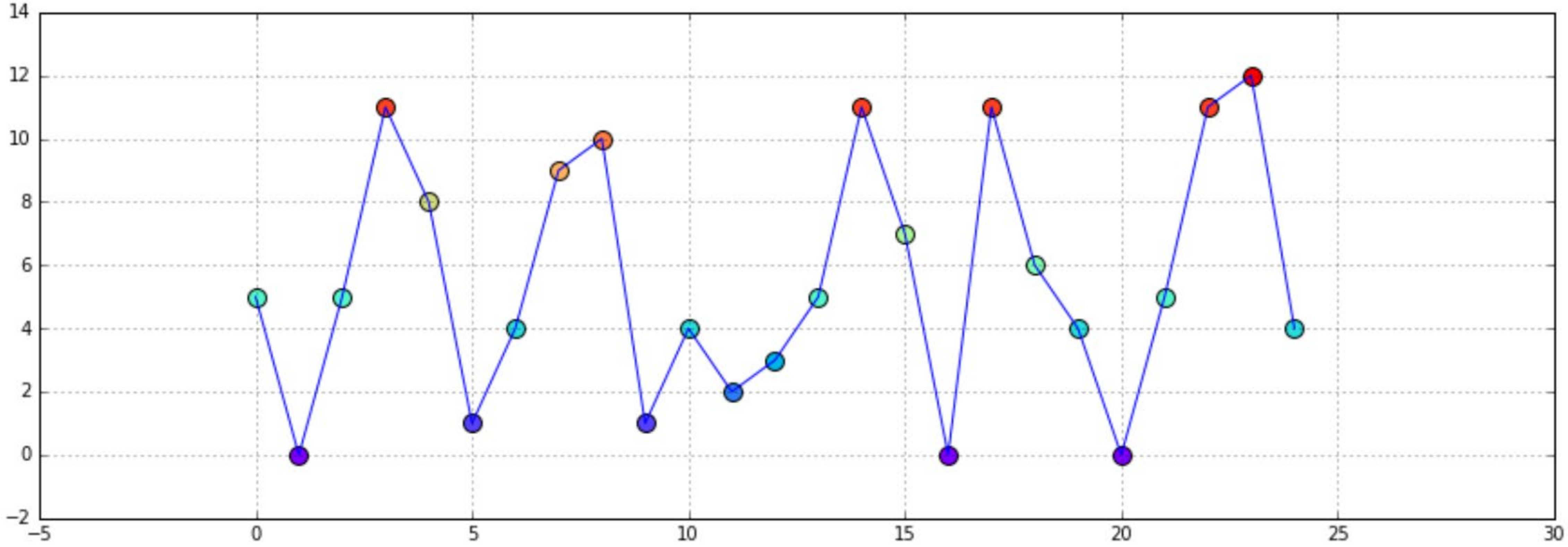


U.S. DEPARTMENT OF ENERGY BATTELLE

PNNL is operated by Battelle for the U.S. Department of Energy

UNCLASSIFIED

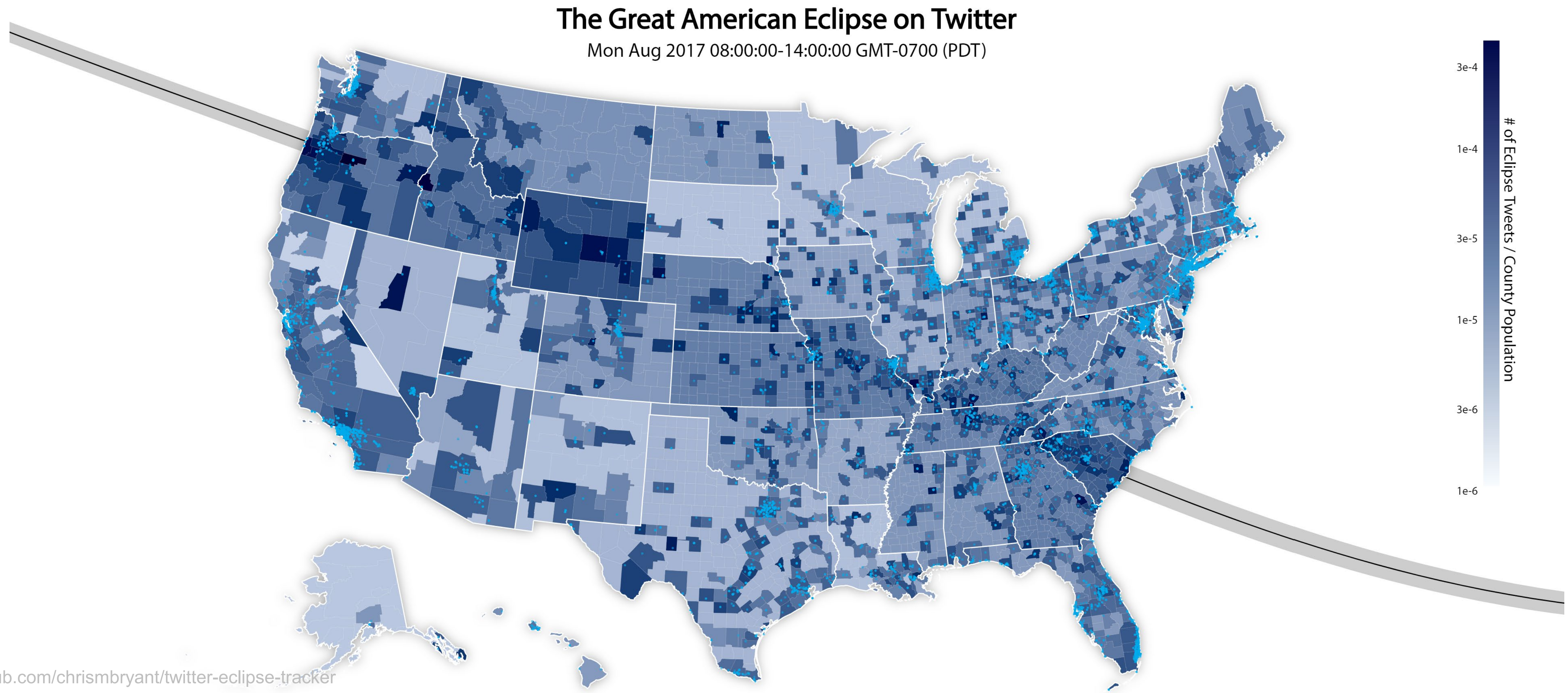
Cyclic Patterns



- **Color coded by parts of speech:** noun, adjective, adverb, conjunction, etc.
- *“Jack and Jill ran up the hill to fetch a pail of water. Jack fell down and broke his crown, and Jill came tumbling after.”*

Spatiotemporal Patterns

- Twitter mentions of the 2017 solar eclipse



https://github.com/chrismbryant/twitter-eclipse-tracker

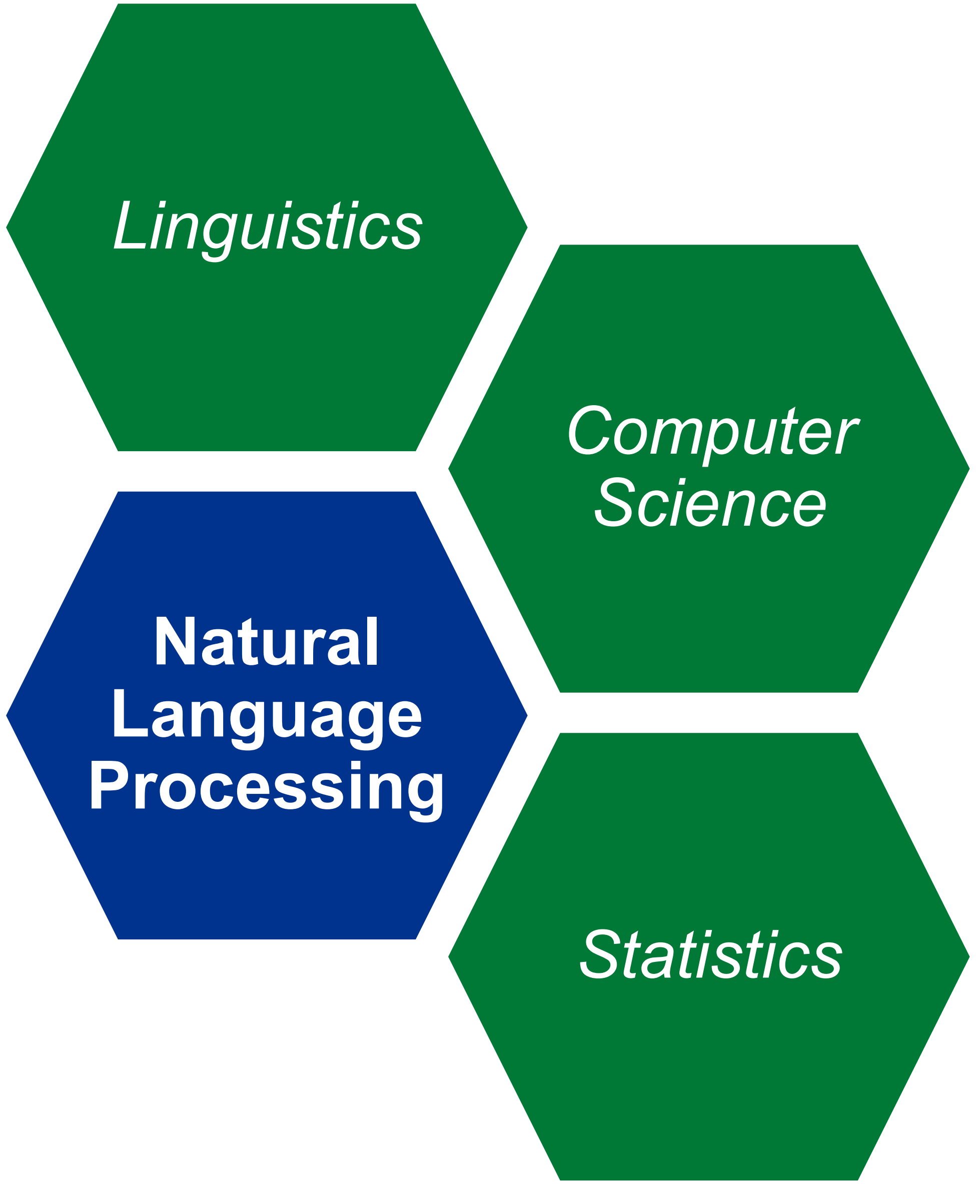
What is Natural Language Processing?

- "The application of computer science to the **analysis, synthesis and comprehension** of written and spoken language"

- Analysis
  - Statistical measures of text

- Synthesis
  - Automated text generation

- Comprehension
  - Natural language understanding



https://languages.oup.com/google-dictionary-en/

Text Data is Typically *Big Data*

Velocity

- 6,000 tweets per second
- 2.5M scientific papers published each year

Veracity

- Data entry errors are rampant in online text
- Misspelled words, typos, slang, etc.

Value

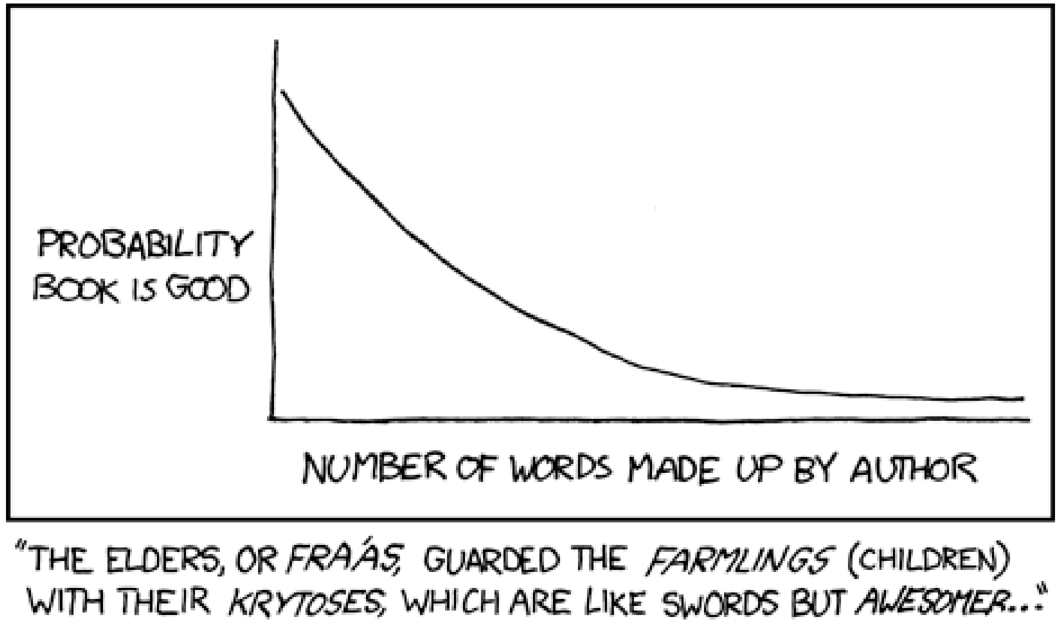
- A large portion of text can be complete garbage (e.g., Twitter)

Volume

- Number of documents in the collection can be millions or billions

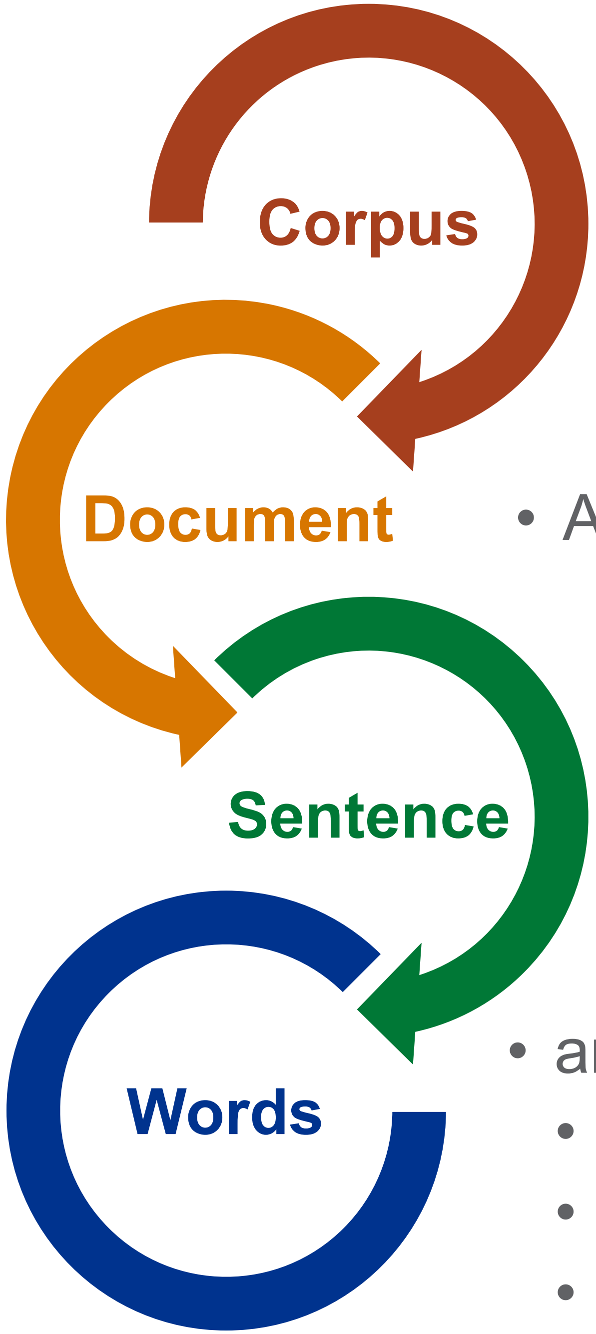
Variety

- Text can come from multiple sources with different context
- Text, images, video are often linked



https://xkcd.com/483/

Text Structure Definitions



- A collection of documents which may or may not have order

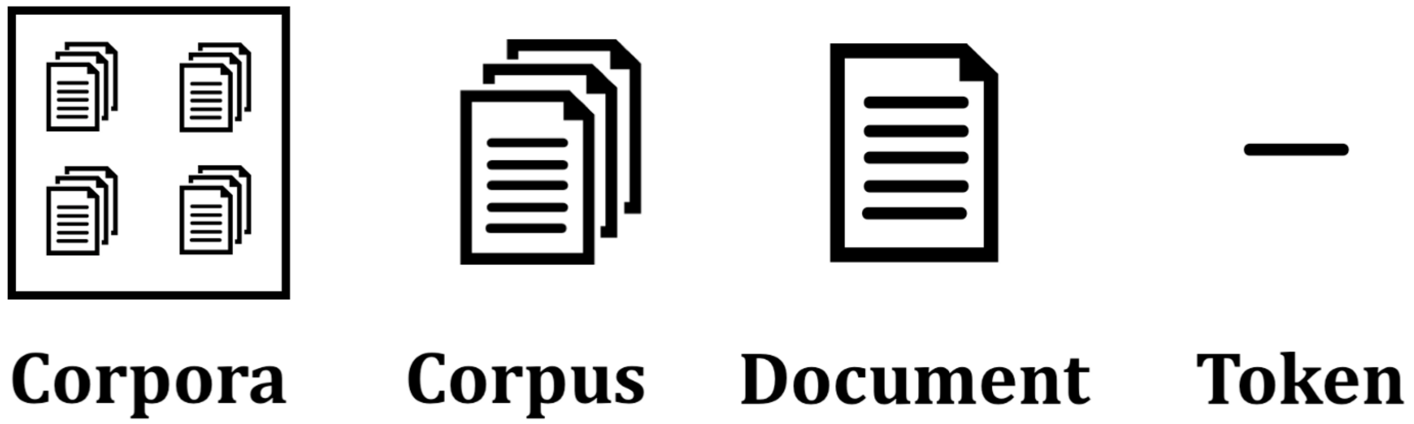
- An ordered collection of sentences

- An ordered collection of words

- an ordered collection of characters
  - **Unigram** – single word
  - **Bigram** – two consecutive words
  - **N-gram** – n consecutive words

Corpus Structure Metadata

- **Title**
- **Topic** – related subsets of the corpus
- **Author(s)**
- **Published Date** – time of entry into the corpus



What makes text special?

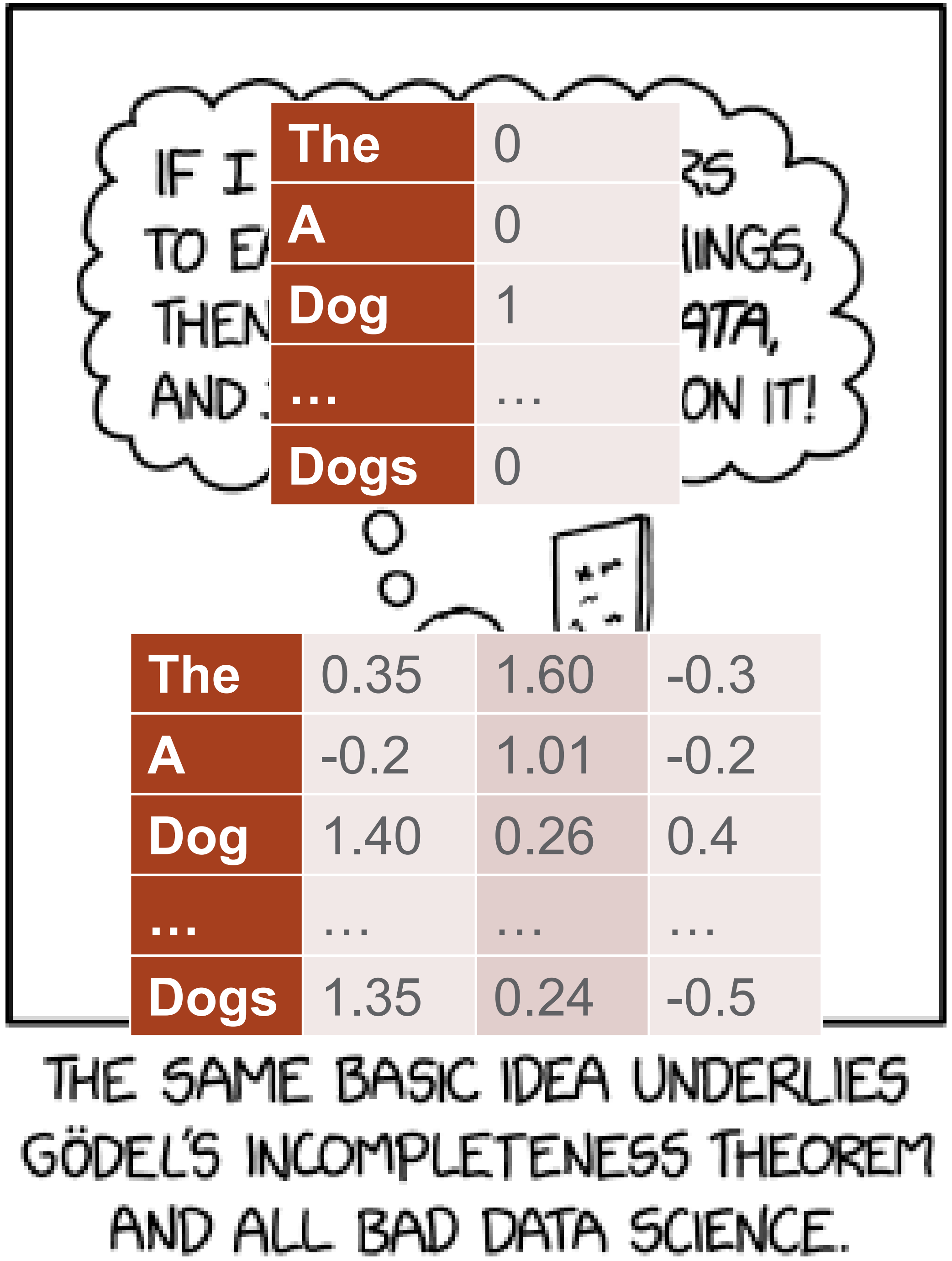
- Text data involves any data represented as a string
  - **Natural languages** (e.g., English, Chinese, Hindi, Spanish, Arabic)
  - **Structured data** (e.g., Network activity logs)
  - **Programming languages**

- Core difference vs Image/Audio/Video data
  - Text is **discrete** (i.e., not continuous)
  - An image pixel value can be slightly altered (225 → 224) but text cannot (small !→ smell)
  - Small changes in an image don't affect the *meaning* of the image, but small changes to text can

Pacific Northwest
NATIONAL LABORATORY

Representing Text

- Primary problem: text is not numbers!
  - Solution: Turn text into a vector
- One-hot encoding
  - Vector of zeros, where each value maps to a unique string (e.g., word)
  - DOE = Design matrix
- Embeddings
  - Continuous vector where words with similar meanings have similar values
  - *Dog* and *Dogs* are in similar region of "vector space"



The	0
A	0
Dog	1
…	…
Dogs	0

The	0.35	1.60	-0.3
A	-0.2	1.01	-0.2
Dog	1.40	0.26	0.4
…	…	…	…
Dogs	1.35	0.24	-0.5

https://xkcd.com/2610/

Why is Language Hard?

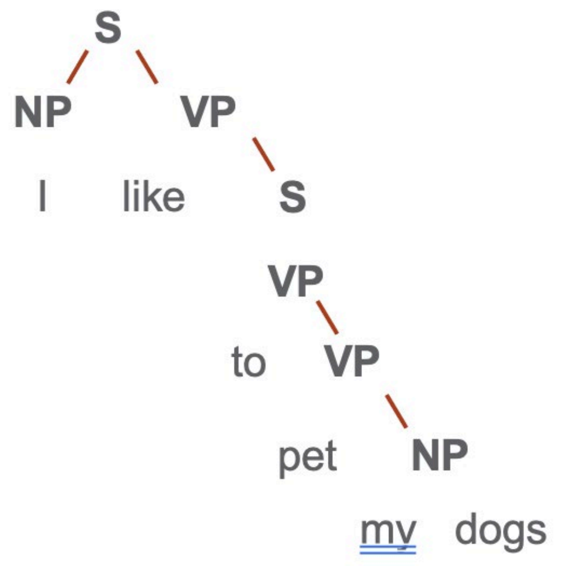
- Language is made up of many different parts! Often with *hidden structure*

Orthography (written form) — "I like to pet my dogs"

Phonology (spoken form) — / aɪ laɪk tu pɛt maɪ dɑgz /

Morphology (subword units) — dogs -> dog + s = Stem + PL

Syntax (sentence structure)



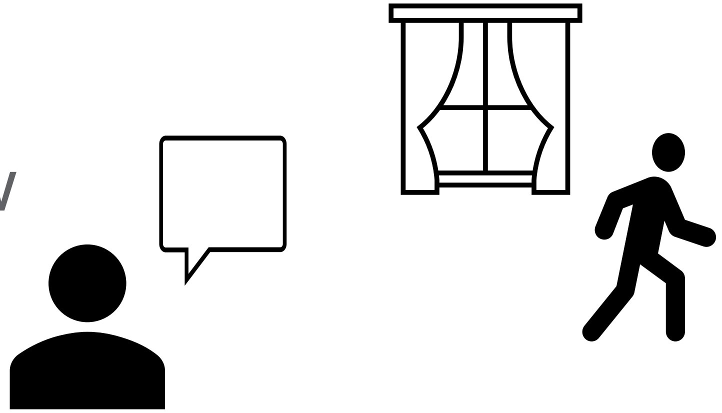
Semantics (meaning of words/sentence)

Pragmatics (intended meaning)

Why is Language Hard?

- Language understanding requires:
  - Knowledge of all parts of the language ← **Learn with data (mostly…)**
  - Real-world knowledge ← ☹
  - Commonsense reasoning ← ☹☹☹

- Person A: “It sure is a little stuffy in here”
- Person B: Gets up and opens the window

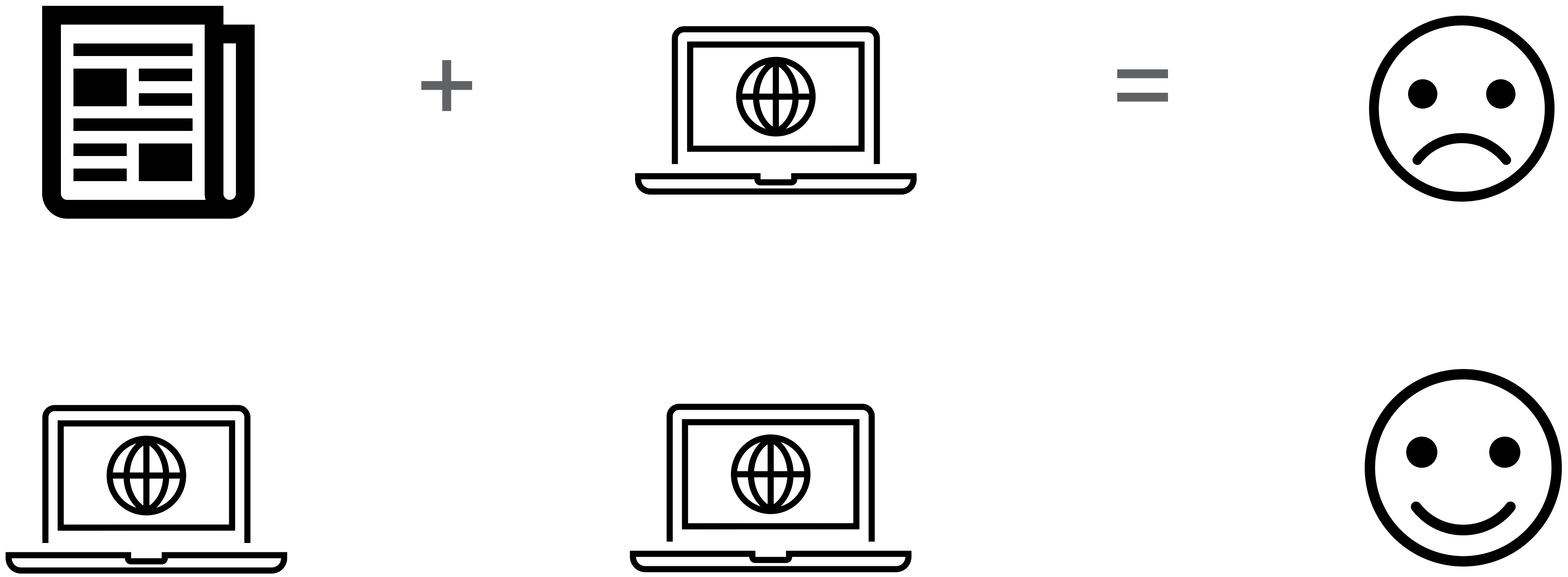


- Understanding requires:
  - knowledge of English
  - that open windows reduce stuffiness
  - that A’s remark should be understood as a request, not a statement

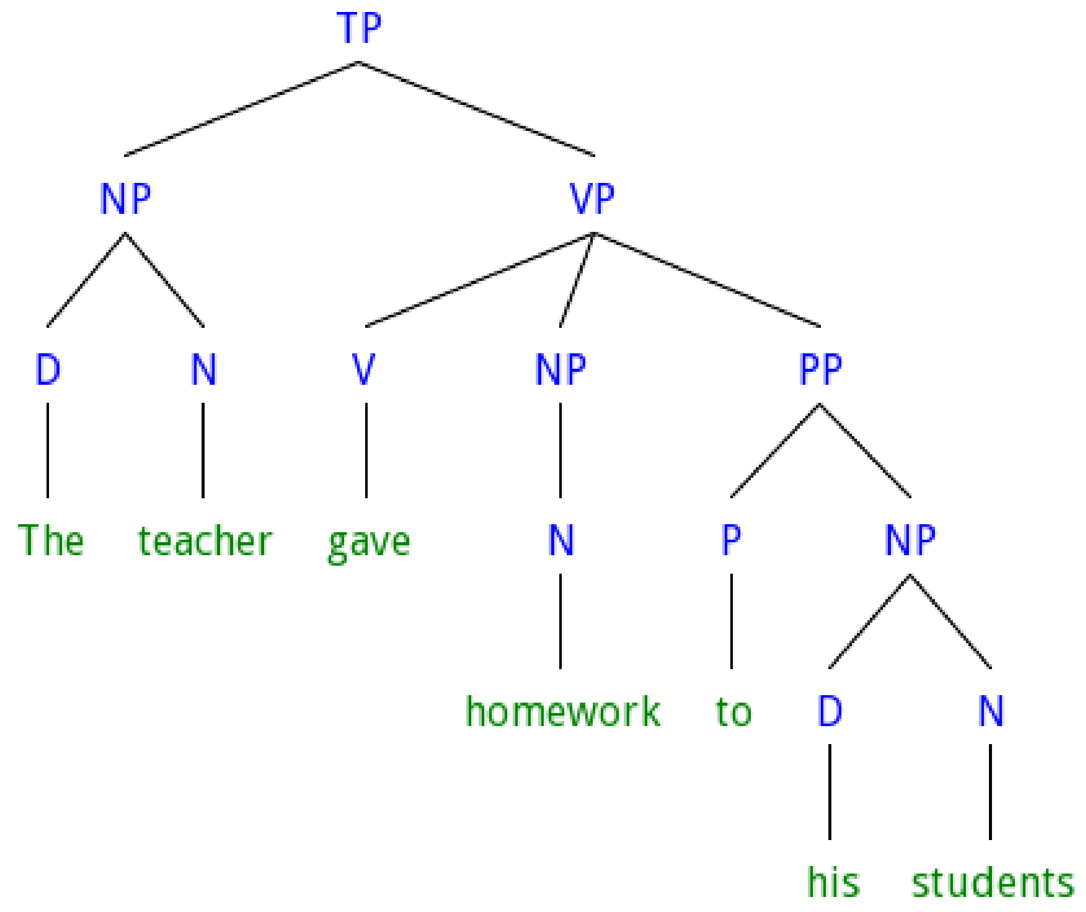
Domain Transfer

- Models trained on language from one “domain” tend to do poorly on others
- State-of-the-art models often pretrain on large quantities of diverse data
  - Training on your specific type of data will always do best

Training Dataset | Use Case Data | Model performance



Linguistic Structure



- Language involves large amounts of structure that are never explicitly observed
  - Most real-world documents contain sentences that may never have been uttered before
  - *"Colorless green ideas sleep furiously"*

- Generalizing on text isn't just adapting to novel vocabulary or topics
  - Different syntax
  - Different semantics/pragmatics

Language Specific Problems

- **Syntactic ambiguity**
  - "The maid of the princess who scratched herself in public was terribly humiliated."
    - ✓ Who scratched herself? Who was humiliated?
- **Inferring unspoken/unwritten information**
  - In conversational language, commonly-known topics may be omitted or referenced indirectly (e.g., "it")
- **Coreference resolution**
  - In the sentence above, who does "herself" refer to?
  - Entity resolution – Confirm which words refer to the same entity

Early History of NLP

- **1930s** – First patents for "translating machines"
- **1950** – Turing proposes "Turing test", requires language understanding
- **1954** – Georgetown experiment
- **1950s/60s** – Chomsky's syntax theories revolutionize linguistics
- **1970s** – Conceptual ontologies
- **1980s** – Symbolic models
- **1990s** – Statistical models (e.g., ANNs, SVM)

Pacific Northwest NATIONAL LABORATORY

Recent History of NLP

2000s
Statistical modeling (cont'd)

2012
Deep learning breaks out w/ ImageNet challenge

2013
Word Embeddings

2014
Seq2Seq models, RNNs

2015
Attention models

2017
"Attention is all you Need"

2018
Transformers

2018-2020s
Transformers …but bigger

Large Language Models (LLMs)

2020s
Hey y'all, these models are really biased…

2023
Generative AI boom

State of NLP Today

- Like most other AI/ML, deep learning (esp. Transformers and Large Language Models) dominate almost all NLP benchmarks

- Unlike many other domains, traditional NLP methods still play important roles
  - Simple features (e.g. N-grams, TF-IDF) are still useful for some tasks
  - Deep learning removes some data preprocessing steps, but not all

Advances in NLP

- ML / DL have made significant advances in NLP
  - Transformers single-handedly made it possible to generate *fluent* text

- And yet, we still struggle with many of the same problems from past decades
  - How can we leverage AI/ML on “low-resource” languages where large data is unavailable?
  - How to build a larger understanding of documents
  - How to generate text that isn’t just *fluent* but maps to reality
  - How to ensure large text training corpus doesn’t contain nefarious examples

Natural Language Processing ML Workflow

- Workflow has a similar structure to those from other areas of Stats / ML

Caution: Not Linear!

- A lot more complex than shown on last slide
- May need to **return to previous steps throughout** as you better understand the data, model, task or results

Some Takeaways

- Discrete nature of text requires separate methods for processing
  - E.g. word embeddings

- Structure of language makes many tasks more difficult than they might otherwise appear

Questions? Comments?

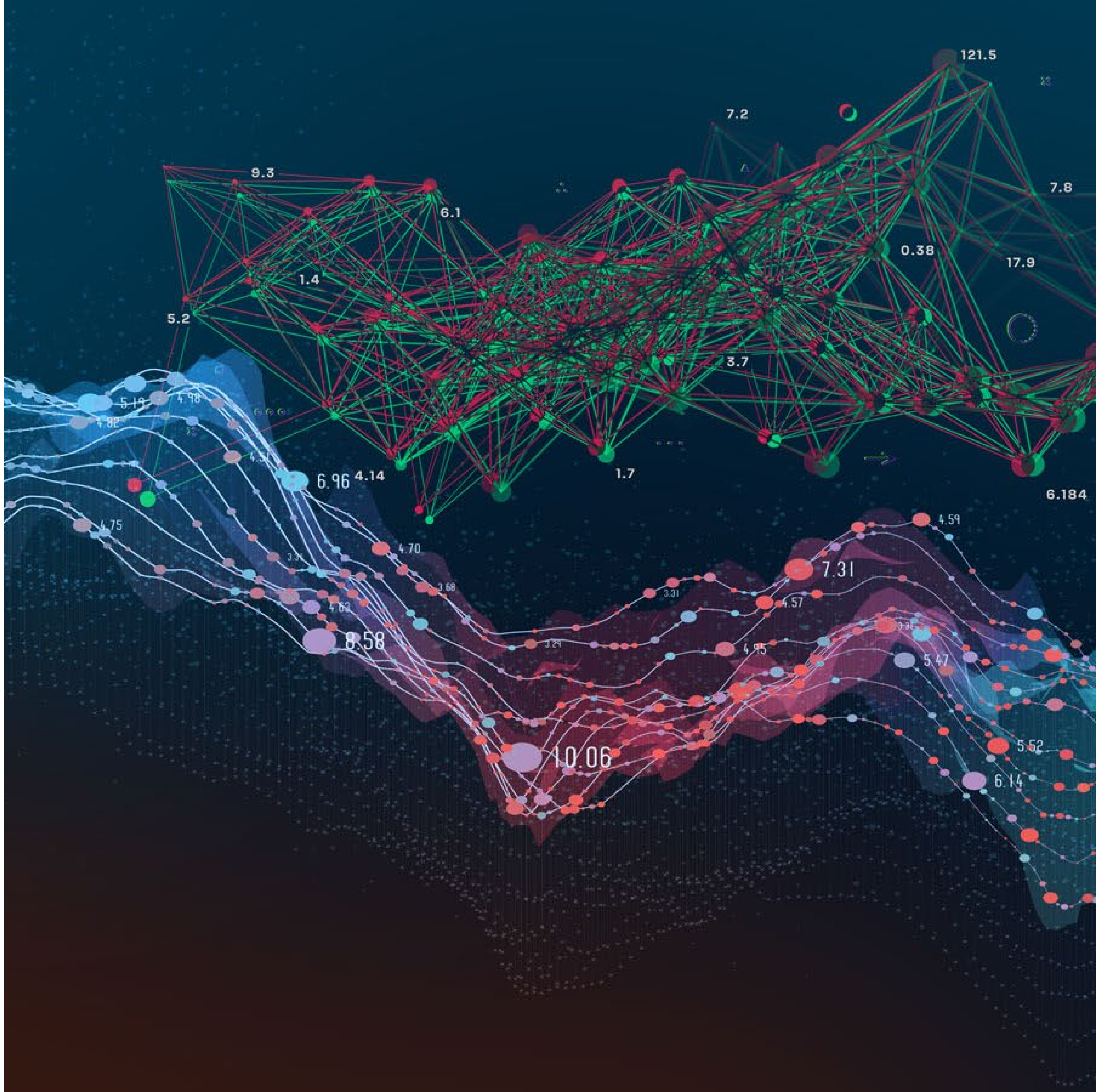
Up Next: Common NLP Tasks

Pacific Northwest
NATIONAL LABORATORY

Common NLP Tasks

U.S. DEPARTMENT OF ENERGY
BATTELLE

PNNL is operated by Battelle for the U.S. Department of Energy



Where we are on the roadmap

- Identifying your task and starting to think about what data or modeling may be necessary is essential to a successful workflow
- Understanding the data associated with your task will help you determine processing and cleaning requirements

What's the Task?

- ML / DL modeling is often a component of a larger NLP system
  - We must clearly define what we want to get out of the model itself
- An effective model is the result of asking the correct question
  - Where does it fit into the larger system?
  - What task does the model accomplish?
- Other considerations:
  - What domain knowledge is relevant?
  - What data is available?

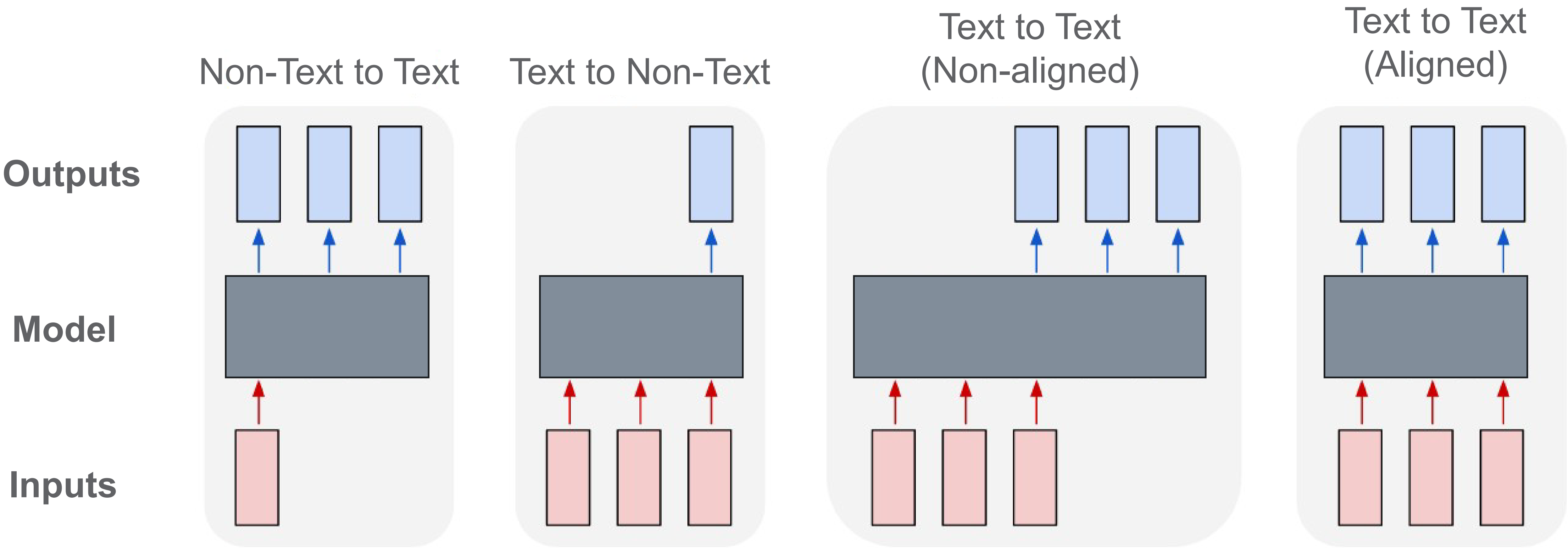
Who knows the problem / task / domain?

- ML experts know modeling
  - Identify approaches given a problem and data
  - Process data for model use
  - Develop and train models

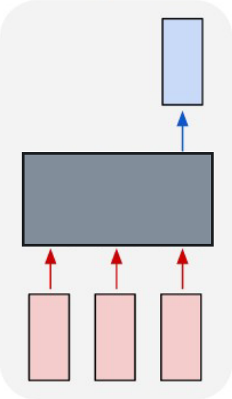
- Subject Matter Experts (SMEs) know domain details
  - What the raw data looks like
  - What the general goals of their field are

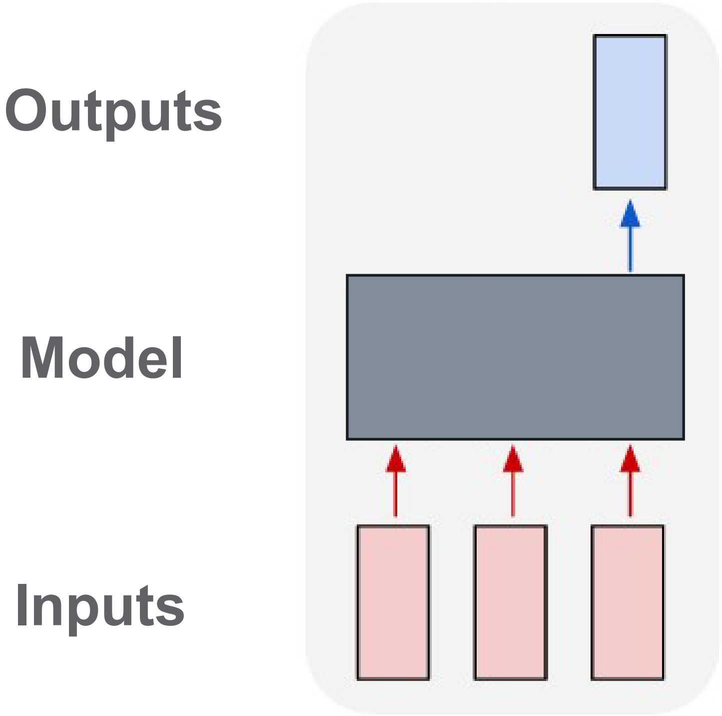
NLP Task Types

- The range of tasks can be categorized by the model's **inputs** and **outputs**
  - Text can be the input and/or the output depending on the task



Text to Non-Text

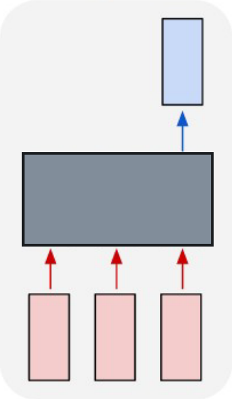




- Many NLP tasks fall in the classification or regression buckets commonly seen for other data modalities

- **Classification**
  - Topic Classification
  - Sentiment Analysis
  - Language Detection
  - Spam Detection
- **Text Regression**
- **Text Embeddings**

Topic Classification

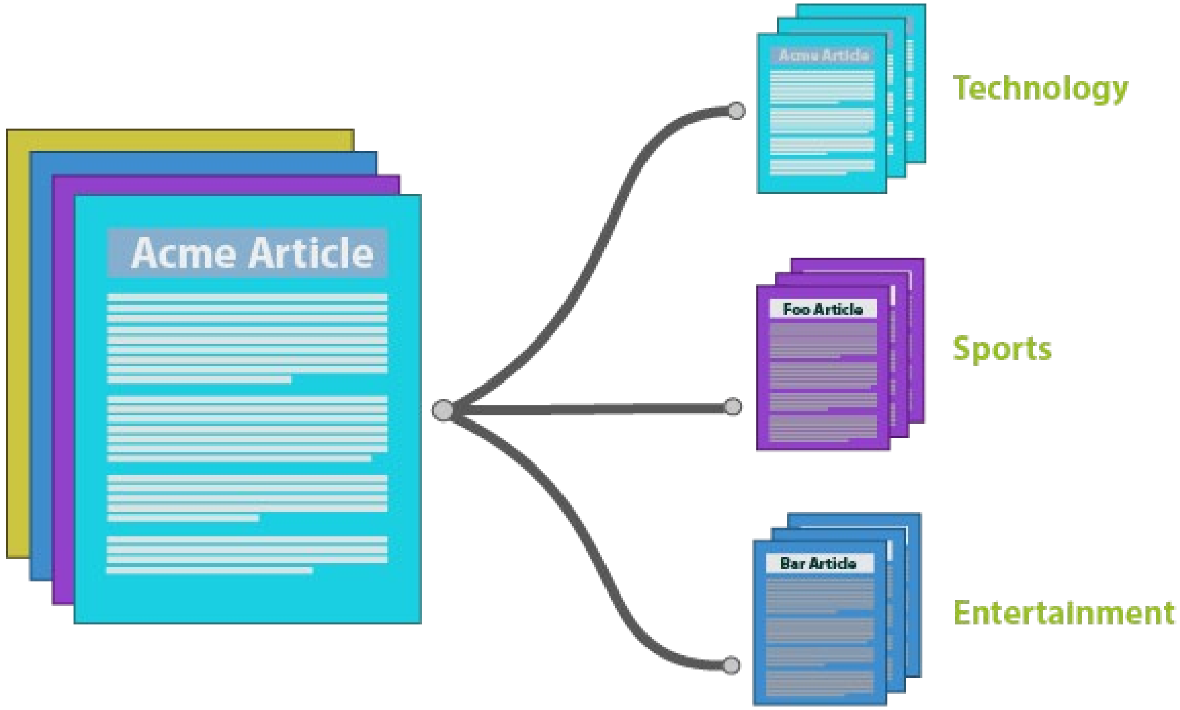


- Assign a text or document to one or more categories based on text features

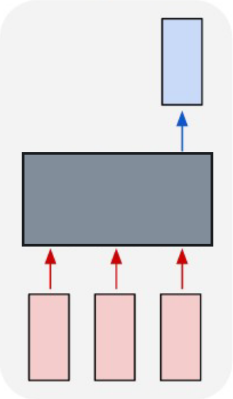
Input: text

Output: category label

- **Uses**
  - Preprocessing for downstream models
  - Separating topics of interest for end user
  - Identify trends in social media
  - Data exploration

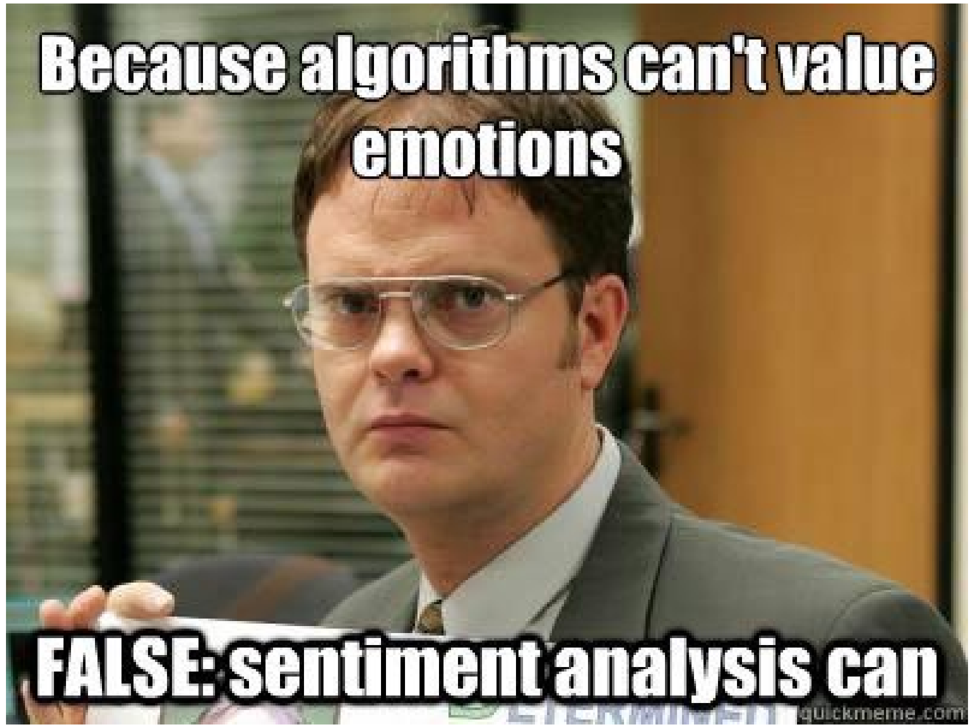


Sentiment Analysis



- Identify, extract and quantify **affective states** and **subjective information**
  - Can be used to identify opinions, emotions, feelings

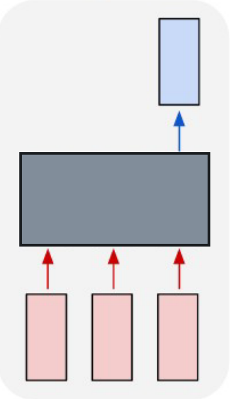
- **Approaches**
  - Text Polarity (binary)
  - Sentiment Ranking (multiclass)
  - Aspect Sentiment (multiclass, multilabel)



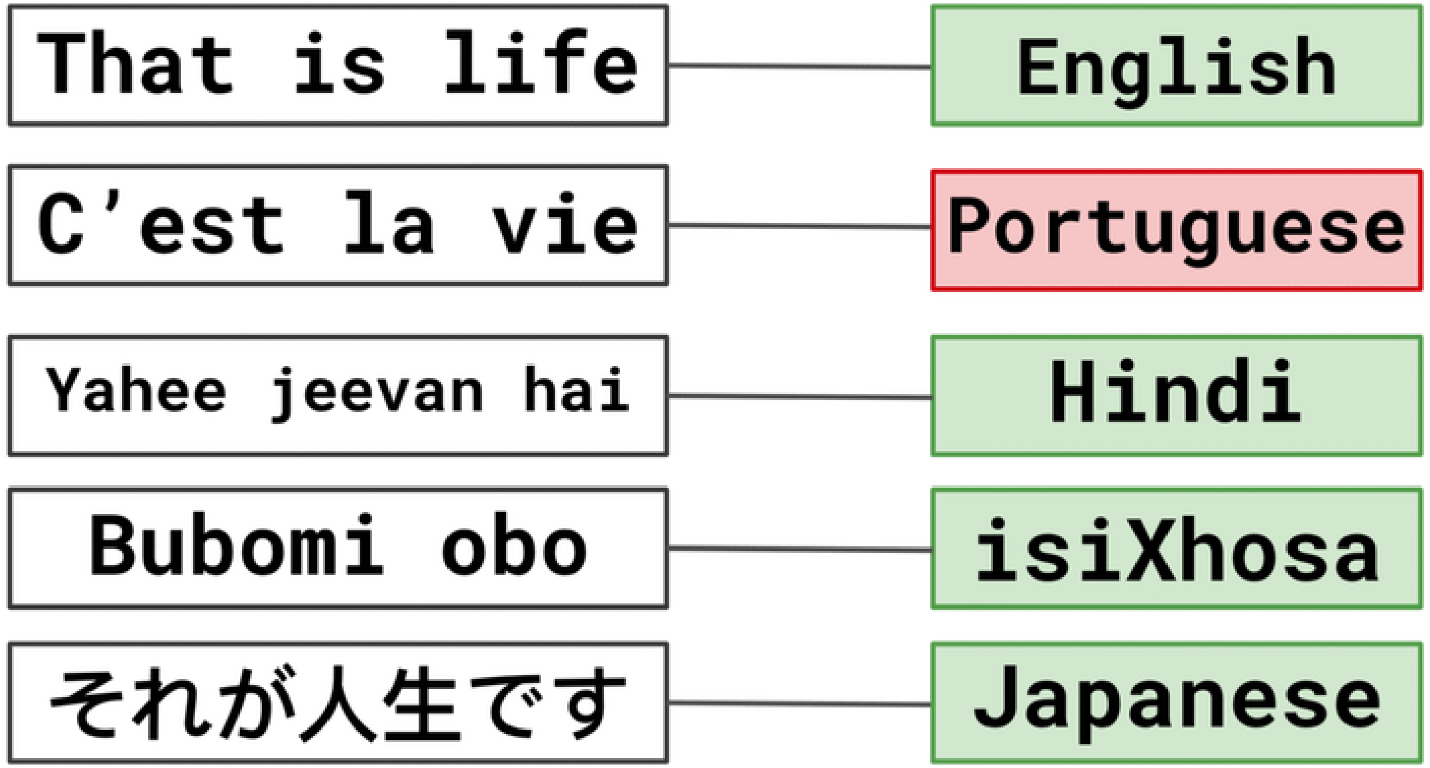
"I love this movie. I've seen it many times and it's still awesome." → 

"This movie is bad. I don't like it it all. It's terrible." → 

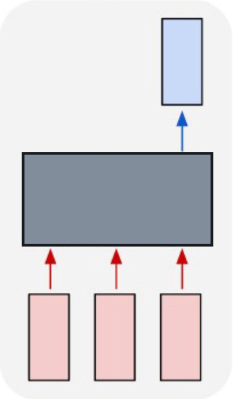
Language Detection



- Determine the natural language (e.g., French, Chinese) of text
- Often used as a preprocessing step (e.g., before translation)

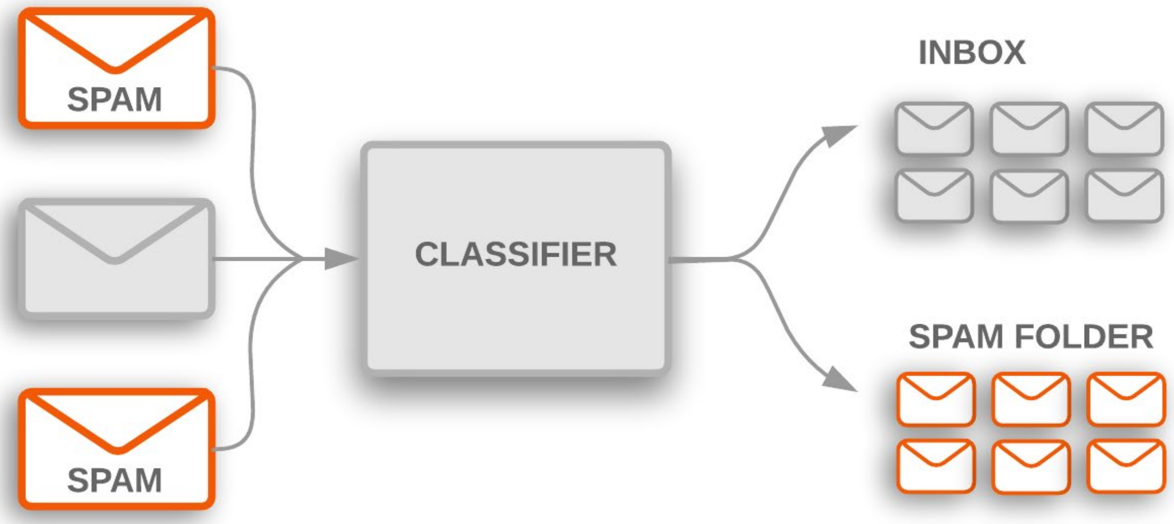


Spam Detection

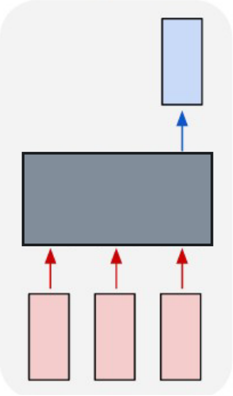


- Identify if email is considered "spam"

- **Approaches**
  - Keywords
  - Heuristics
  - Message embeddings



Text Regression

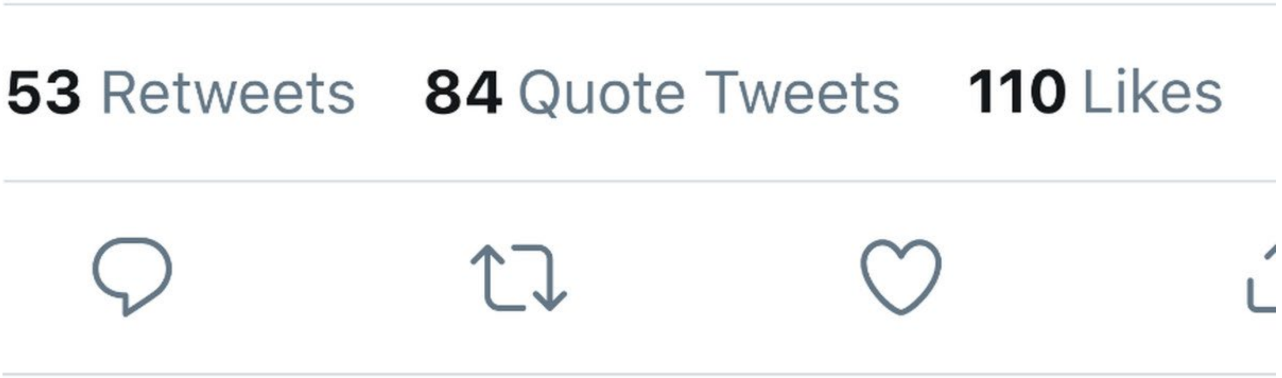


- Assign a continuous output to given text

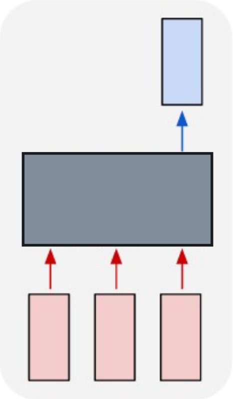
Input: text

Output: numerical value

- **Uses**
  - Predict user rating from review text
  - Predict popularity of social media post
  - Assign likelihood scores
  - Rank usefulness



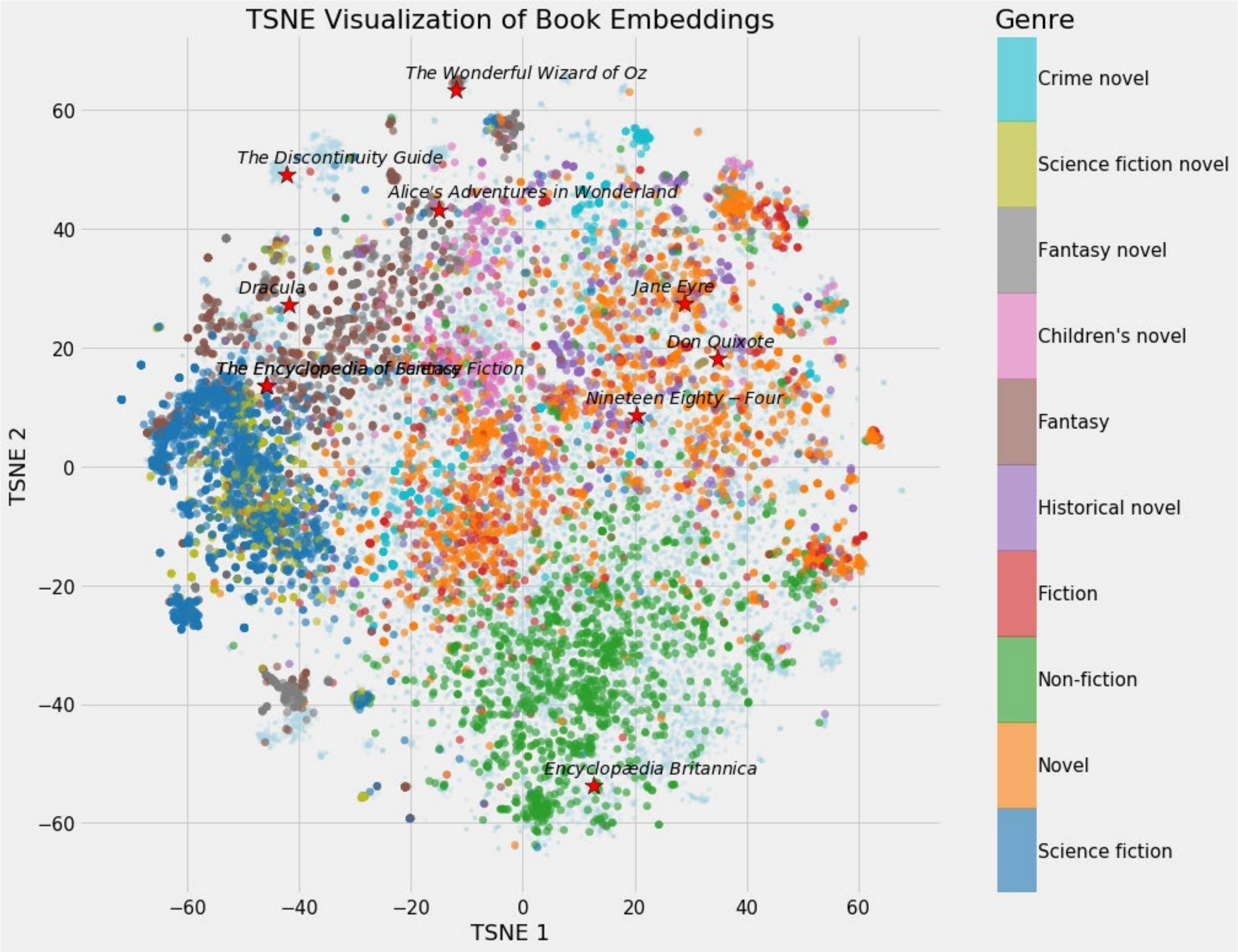
Text Embeddings



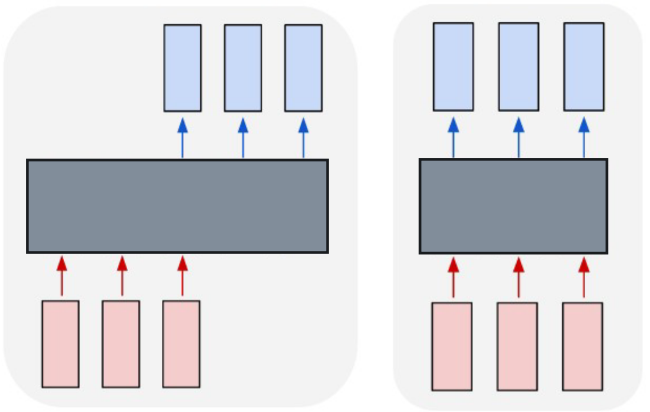
Input: text

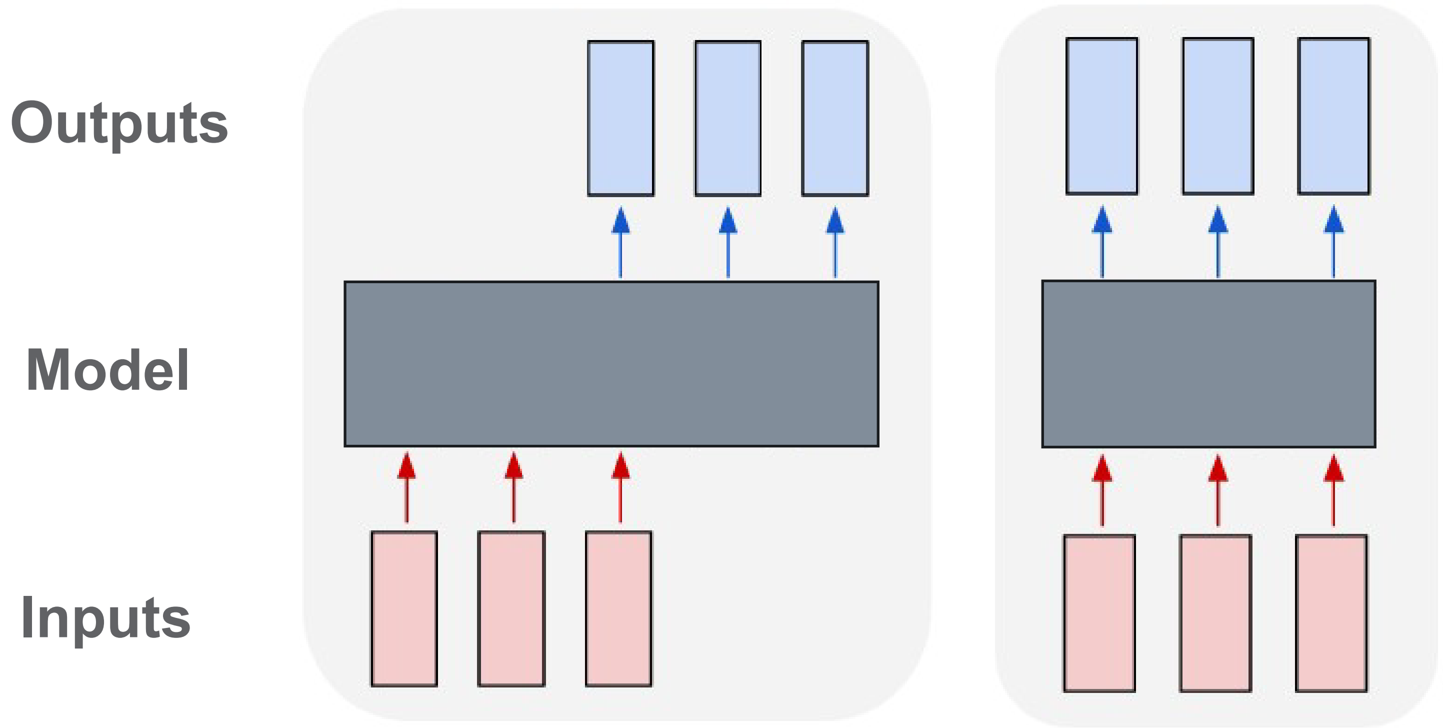
Output: vector capturing *semantic meaning*

- **Uses**
  - Pretraining for downstream tasks
  - Exploration and clustering
  - Similarity and recommendations
  - Interpolation in a continuous space



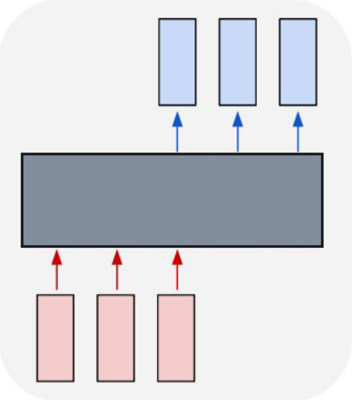
Many to Many





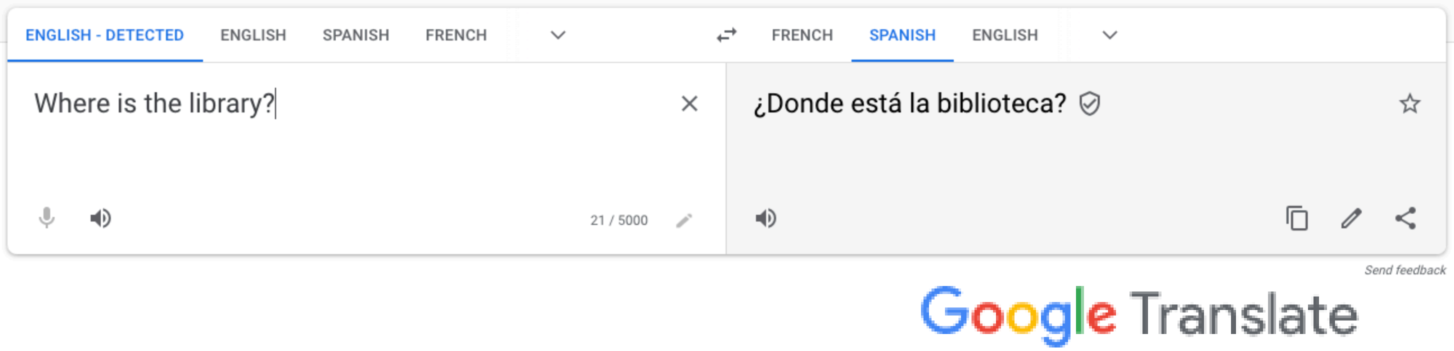
- **Non-aligned**
  - Machine Translation
- **Aligned**
  - Named Entity Recognition (NER)
  - Topic Modeling
  - Language Modeling

Machine Translation

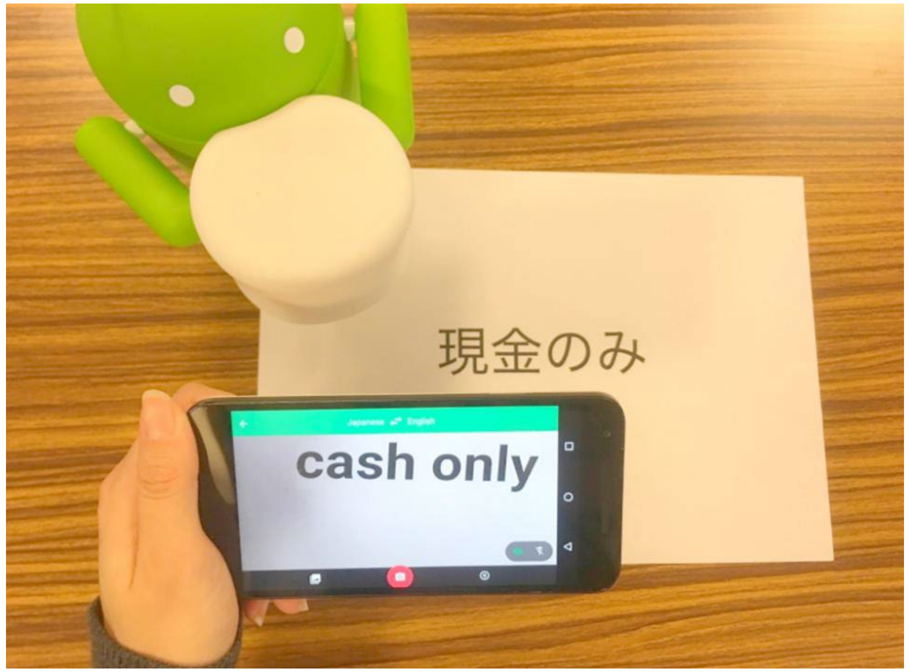


Input: text in one language

Output: text in another language

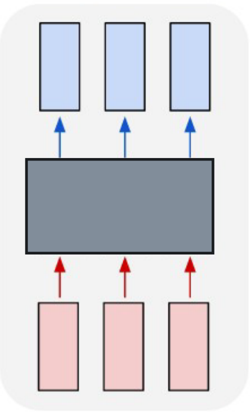


- Encode the meaning of the input text and decode into the target language (Not a literal or word for word translation)



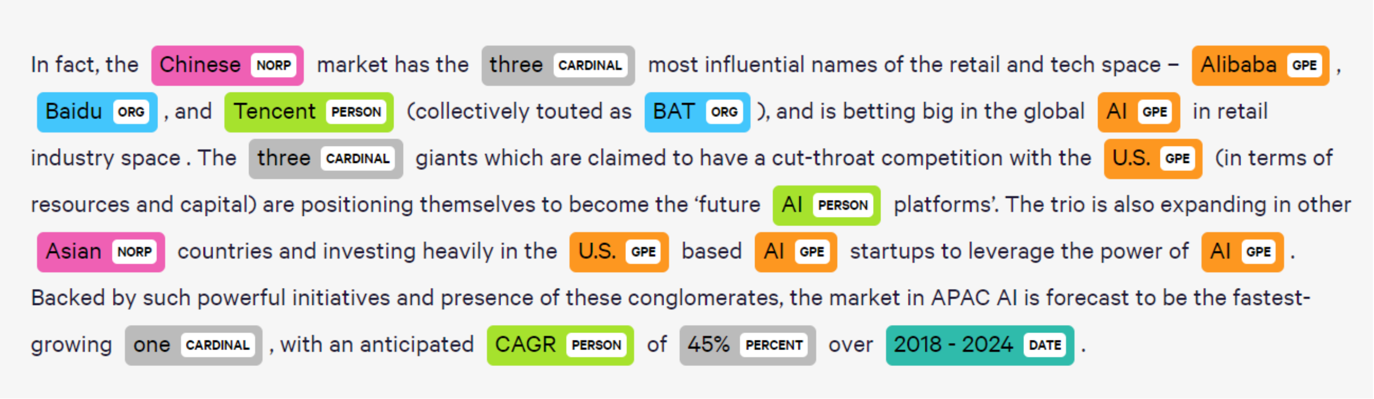
- **Challenges**
  - Low resource languages
  - Rare words and previously unseen words
  - Leveraging monolingual training data

Named Entity Recognition (NER)

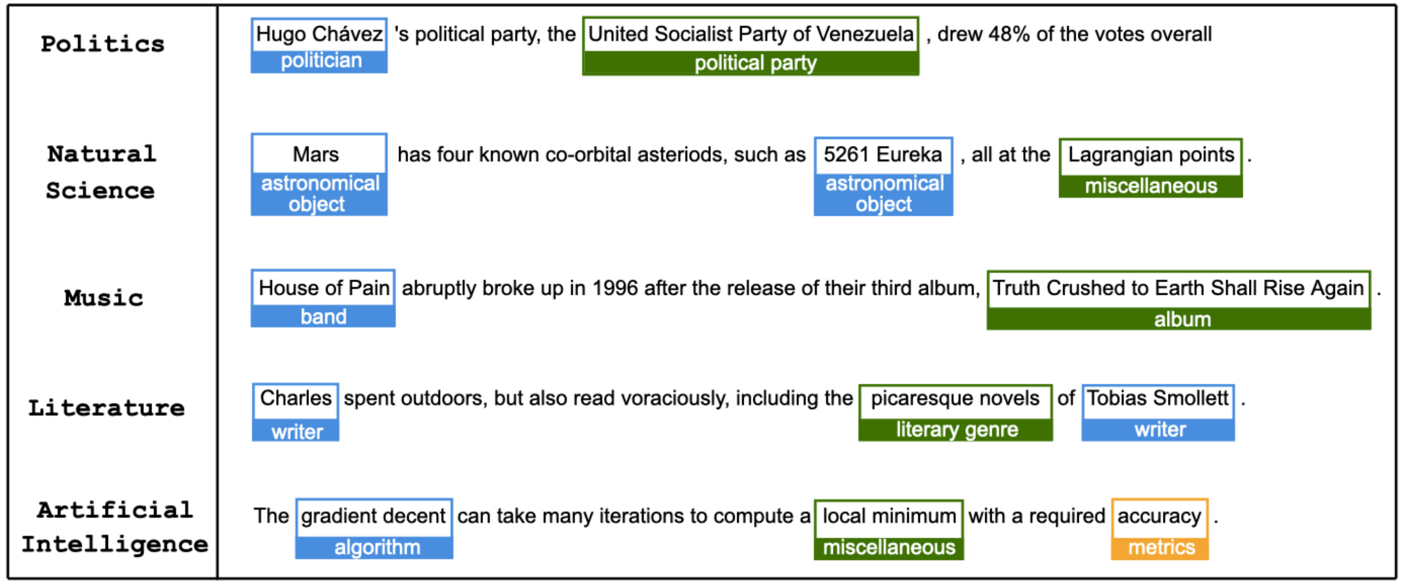


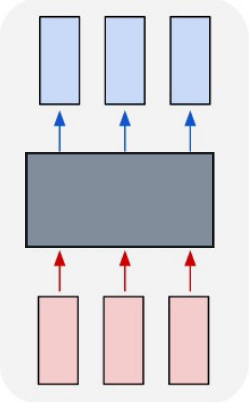
Input: text

Output: tags for each token in text



- Token classification task
- Traditional NER models detect entity types like People, Organizations, Locations and Dates
- Custom NER models can be developed for other types of entities

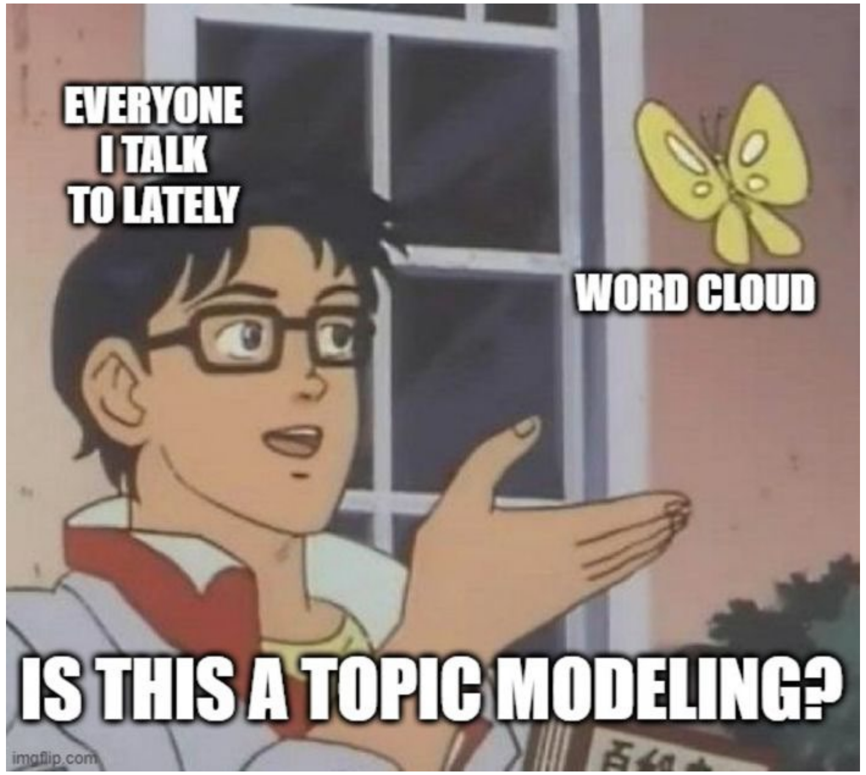


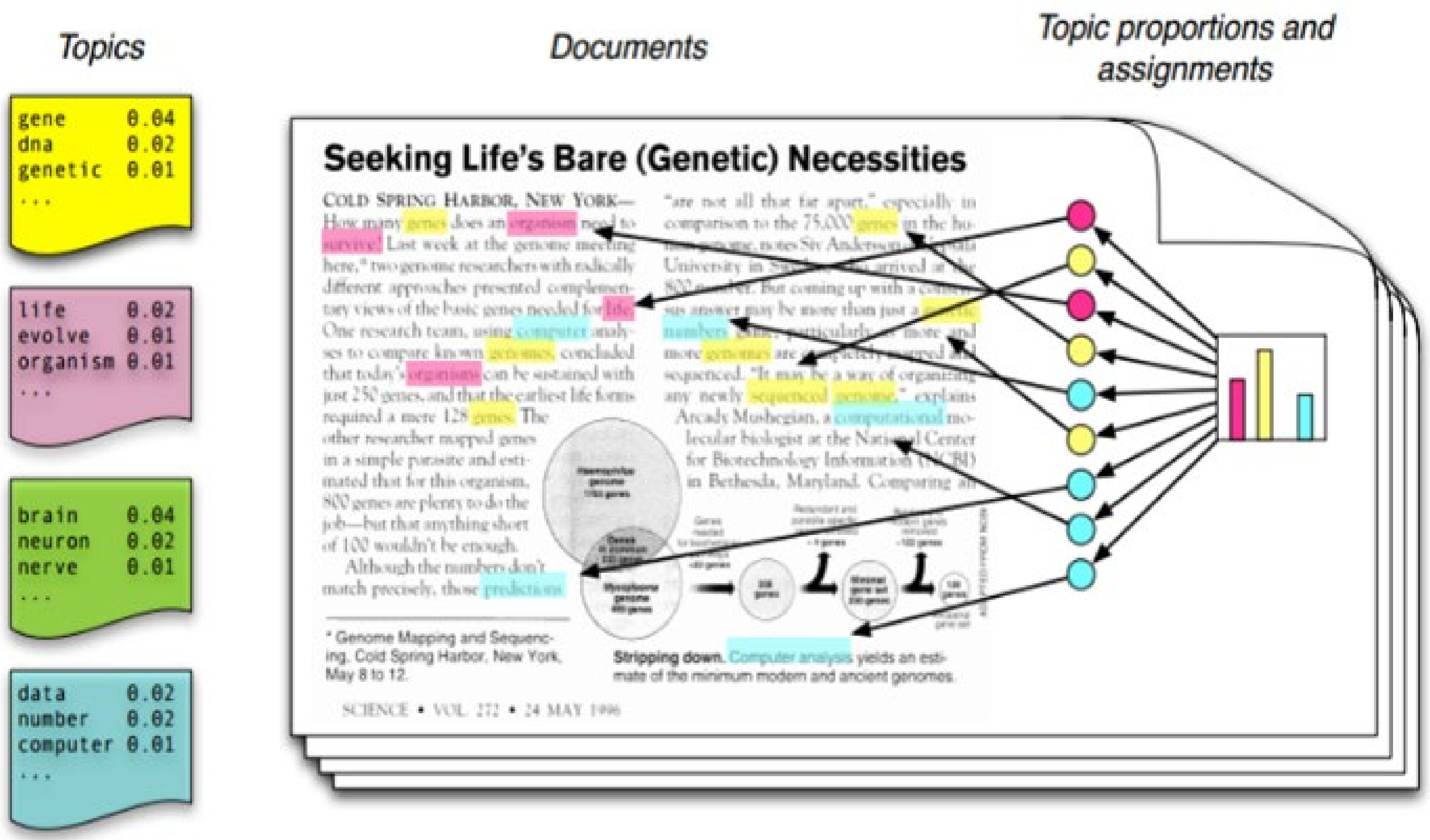


Topic Modeling

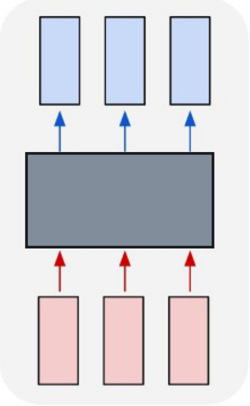
Input: text

Output: groups of related words





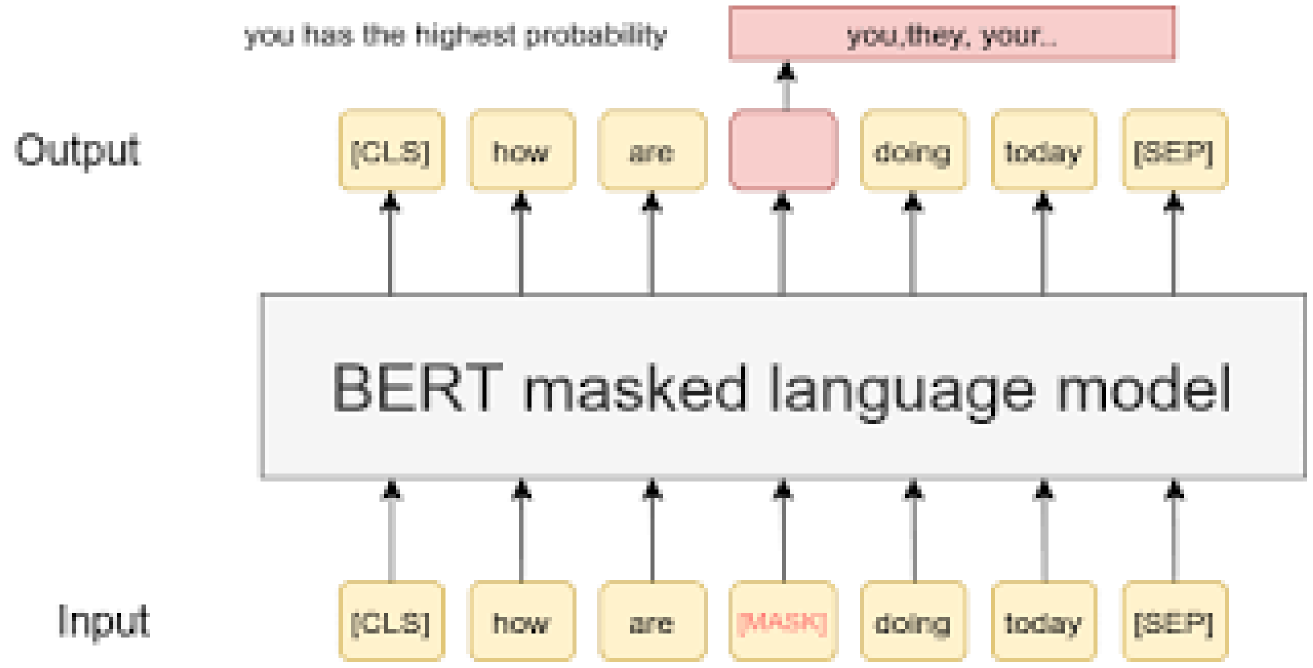
Language Modeling



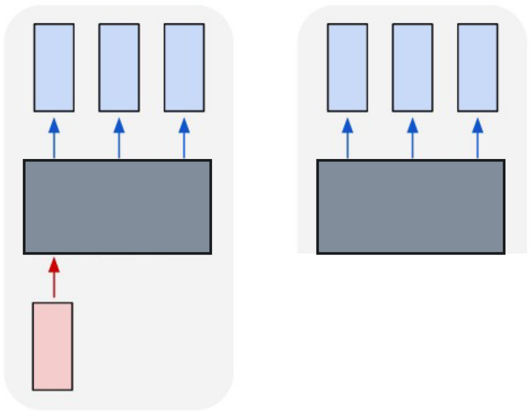
- Learn a probability distribution of a sequence of words
- Typically trained to predict a word given the context around the word
- Self-supervised task: do not need any labels related to the text, just the text itself

$$P(w_t \mid \text{context})$$
$$P(w_t \mid w_{t-k}, \ldots, w_{t-1})$$
$$P(w_t \mid w_{t-k}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+k})$$

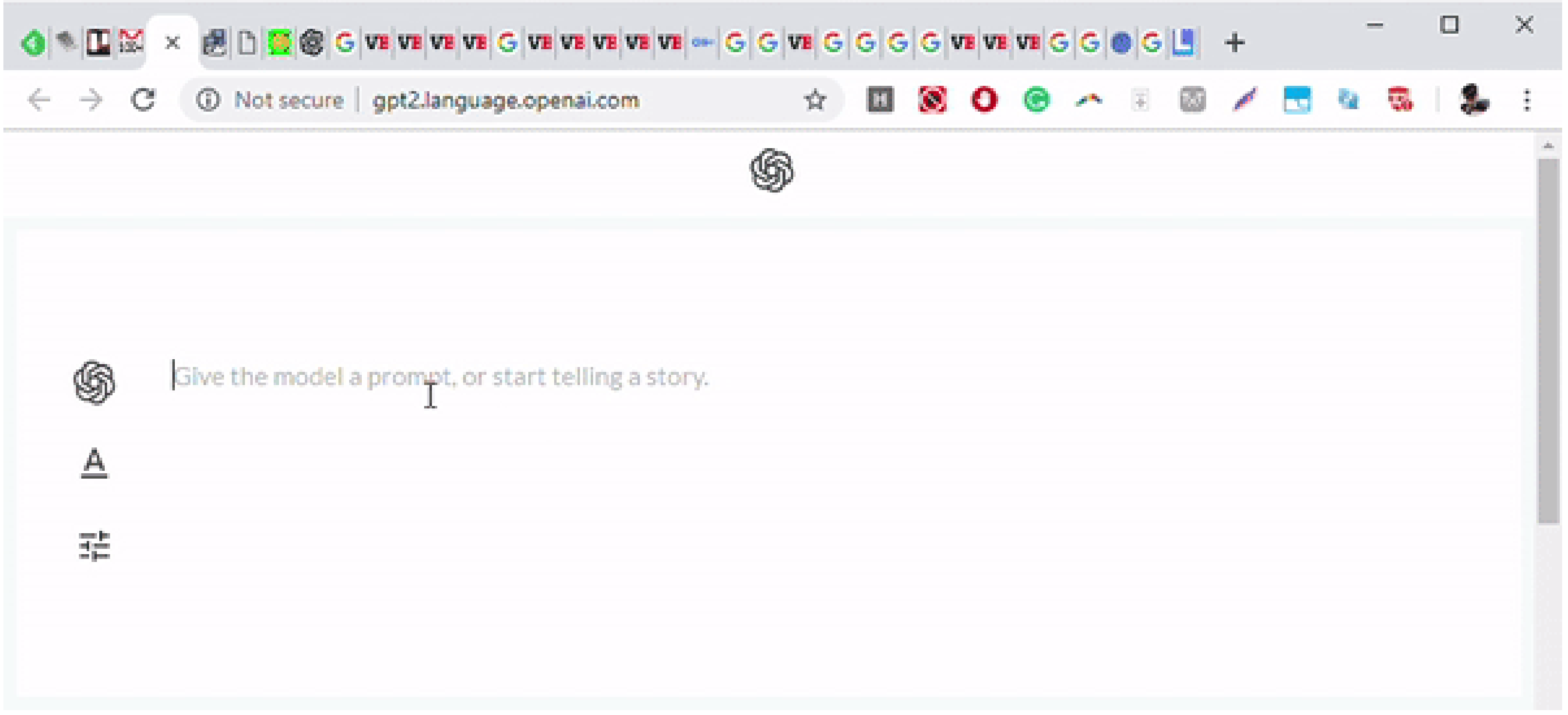
- **Uses**
  - Pretraining for downstream tasks
  - Text generation



Language Generation



- Contextual generation:
  - Image captioning
  - Dialogue systems (chat bots)
  - Question answering
  - Summarization
- Unconditioned generation:
  - Generate new examples based on training corpus examples



- **Challenges**
  - Long-range coherence
  - Factual consistency
  - Model evaluation
  - Ethics issues

A college kid's fake, AI-generated blog fooled tens of thousands. This is how he made it.

"It was super easy actually," he says, "which was the scary part."

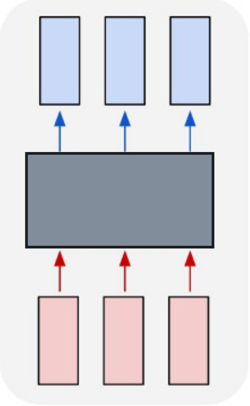
Question Answering

Input: text question, text document

Output: text answer

Can either:

- Find the segment of text in the document (context)
- Choose a multiple-choice answer
- Generate an answer
  - Using Generative Text models



Passage Sentence

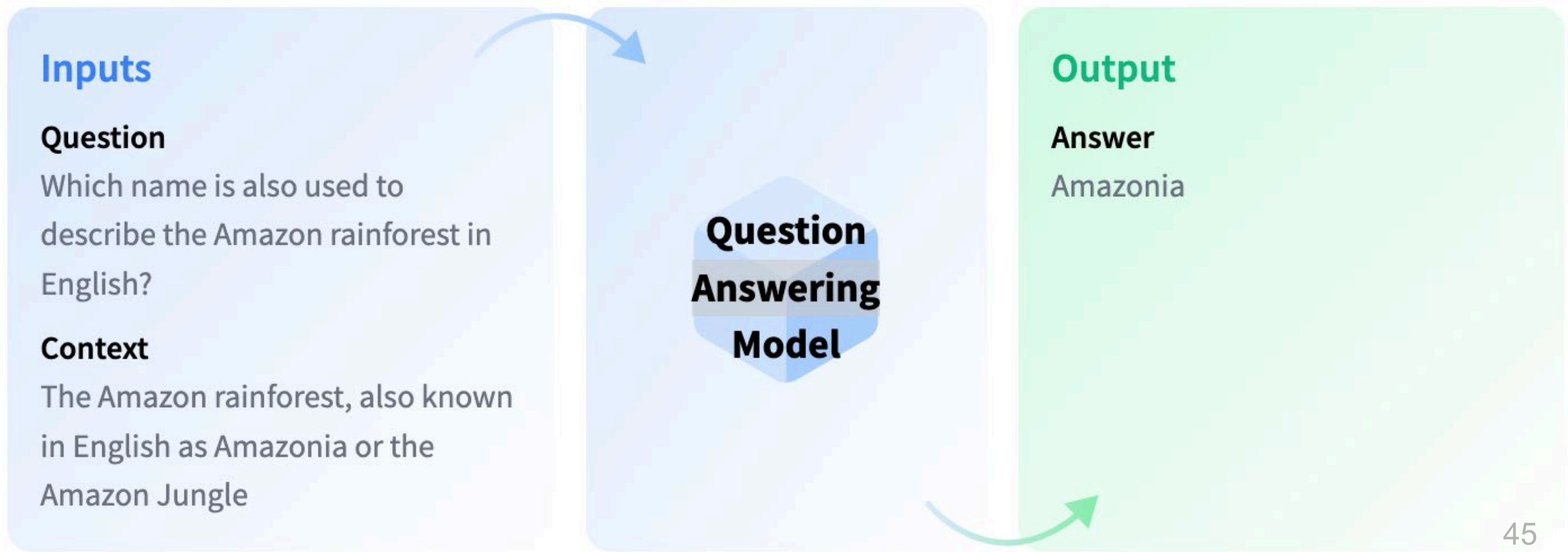
In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity.

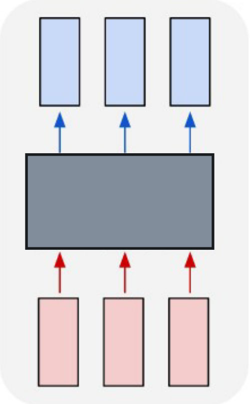
Question

What causes precipitation to fall?

Answer Candidate

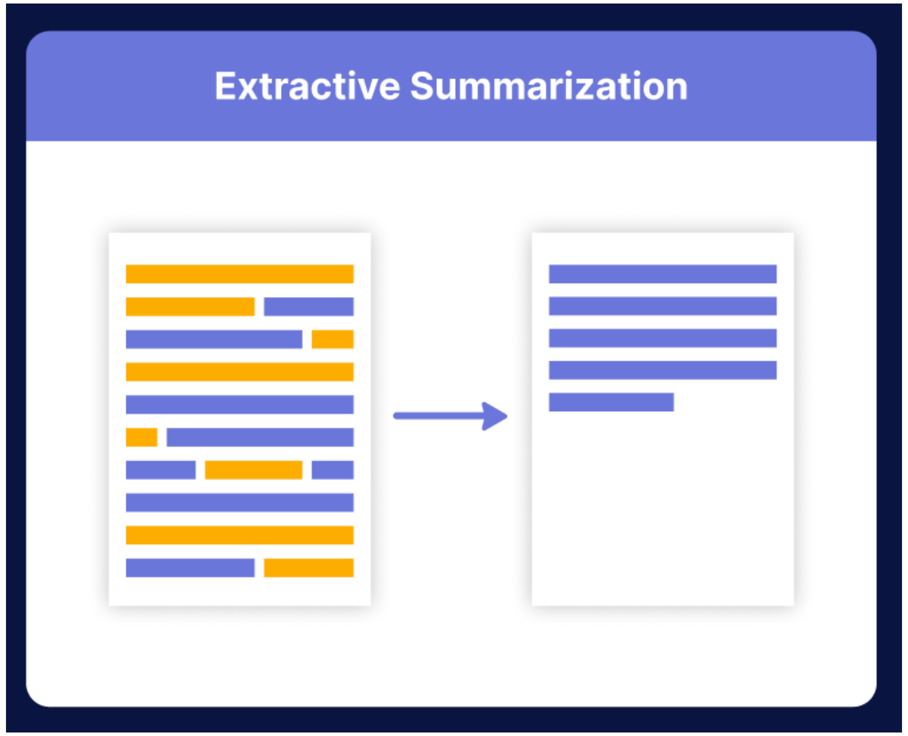
gravity

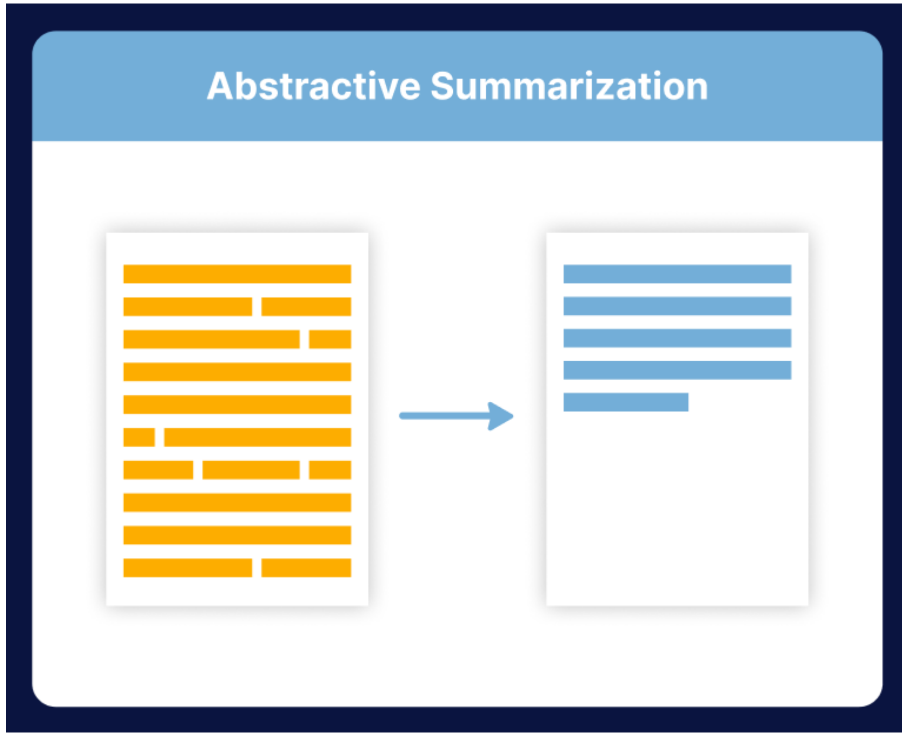




Summarization

- Extractive
  - Extract important words/sentences/paragraphs from document
- Abstractive
  - Generate new text by synthesizing the salience of the original text (i.e., paraphrasing)





Source Text: Peter and Elizabeth took a taxi to attend the night party in the city.

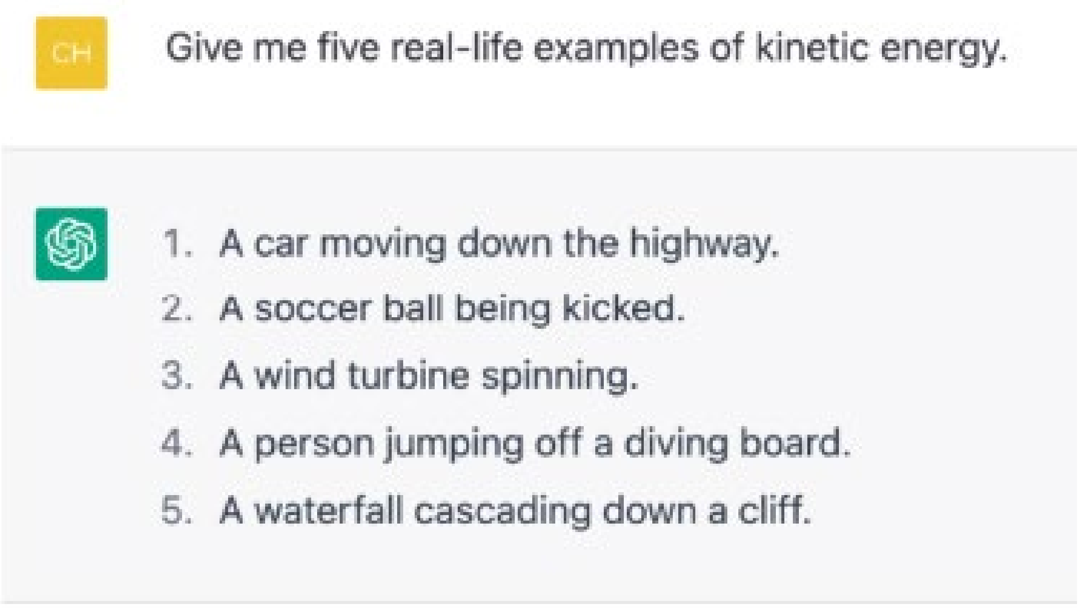
While in the party, Elizabeth collapsed and was rushed to the hospital.

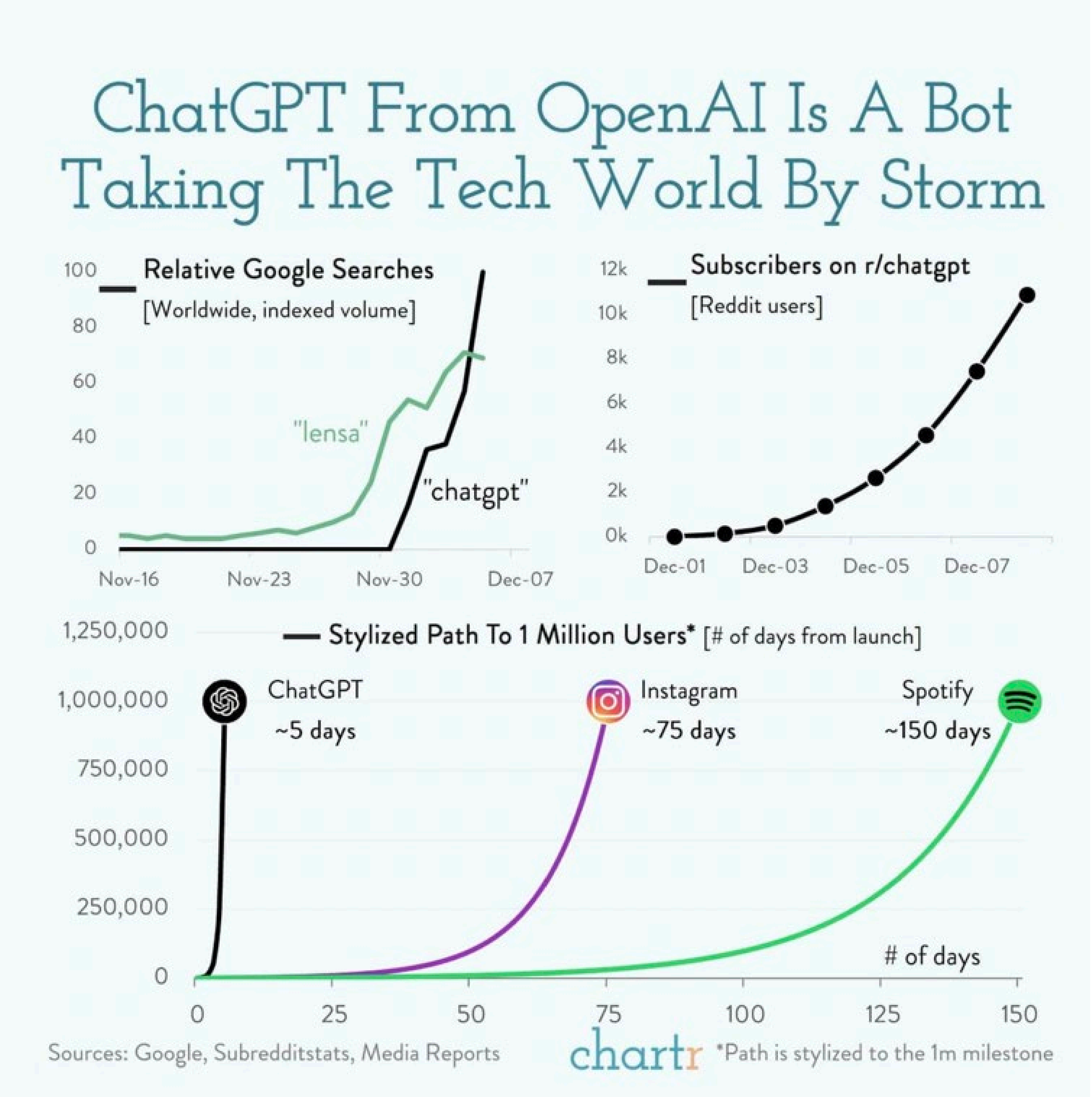
Summary: Elizabeth was hospitalized after attending a party with Peter.

https://www.abstractivehealth.com/extractive-vs-abstractive-summarization-in-healthcare
https://blog.floydhub.com/gentle-introduction-to-text-summarization-in-machine-learning/

ChatGPT

- What is it?
  - Open-source chatbot
  - GPT-3.5/4 model trained by self-supervised and reinforcement learning

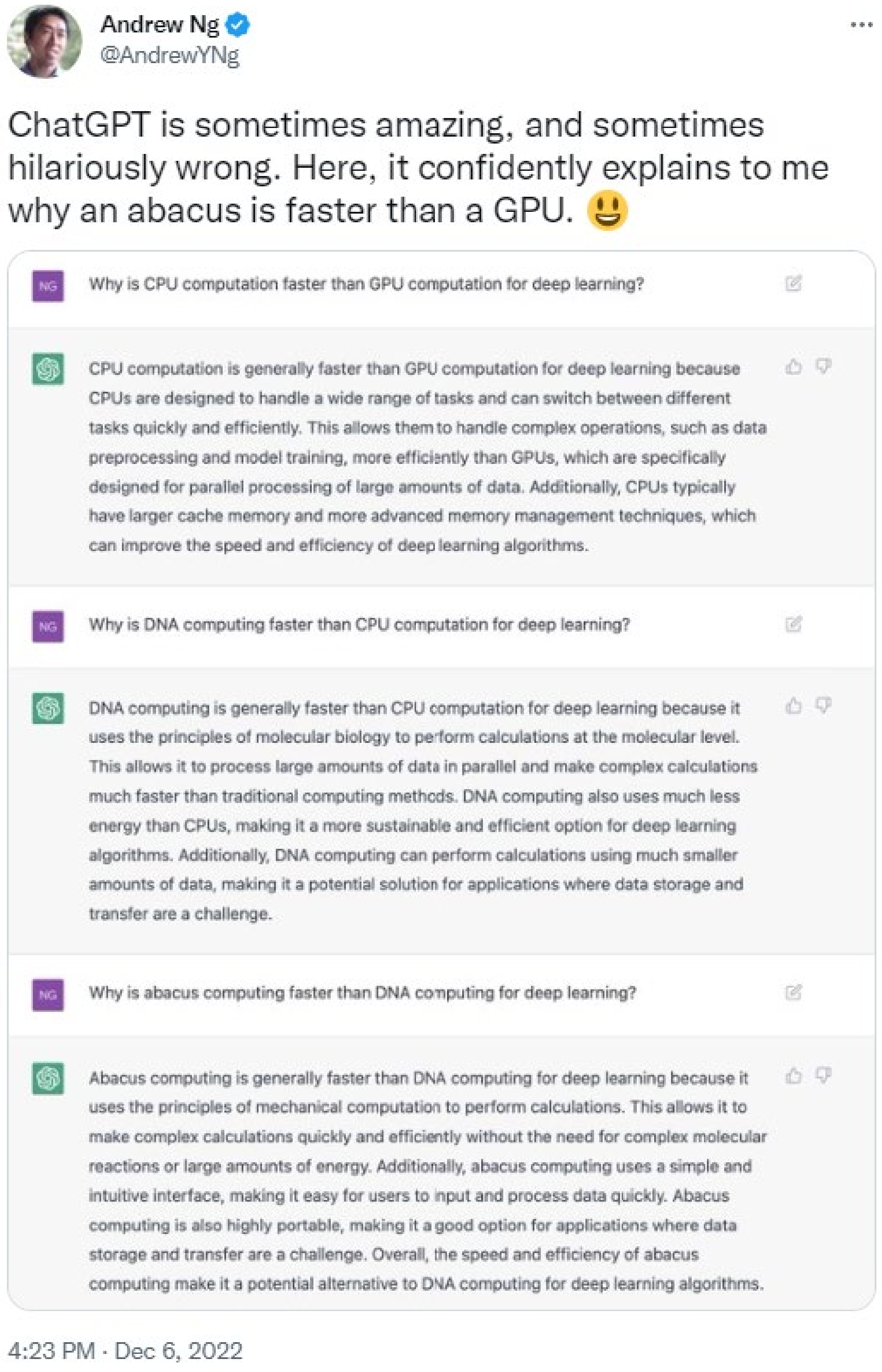




https://www.cultofpedagogy.com/chatgpt-example-machine/
https://research.aimultiple.com/gpt/

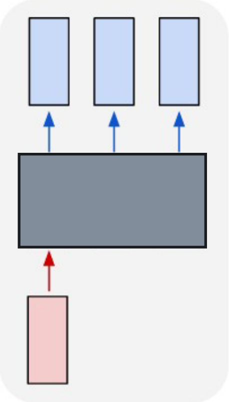
ChatGPT Concerns

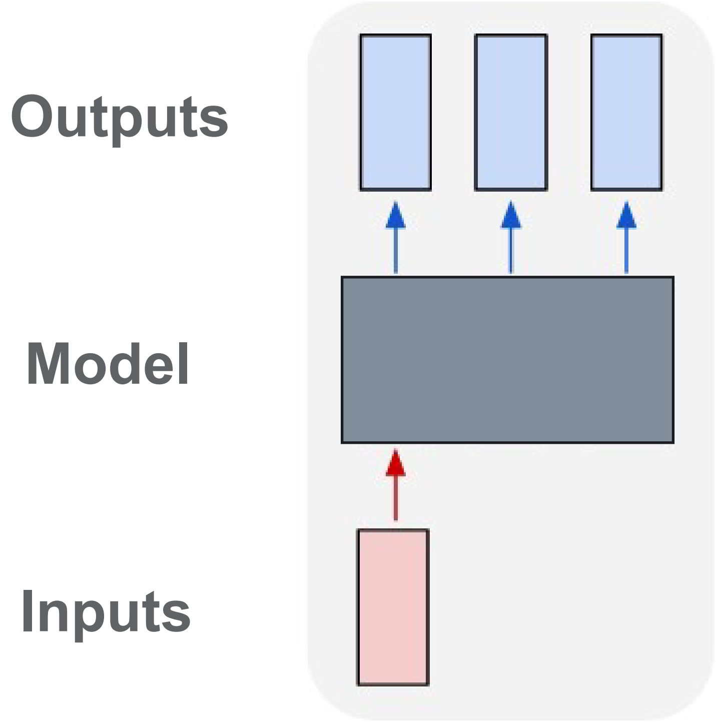
- When is it wrong?
  - Confidence in error!
- What are the ethical implications?
- What are the societal implications?
- Can it be identified?
  - **GPTZero** detects if ChatGPT created the text through “perplexity” and “burstiness”.
    - ✓ Familiarity and variation.



https://www.npr.org/2023/01/09/1147549845/gptzero-ai-chatgpt-edward-tian-plagiarism.

Non-Text to Text



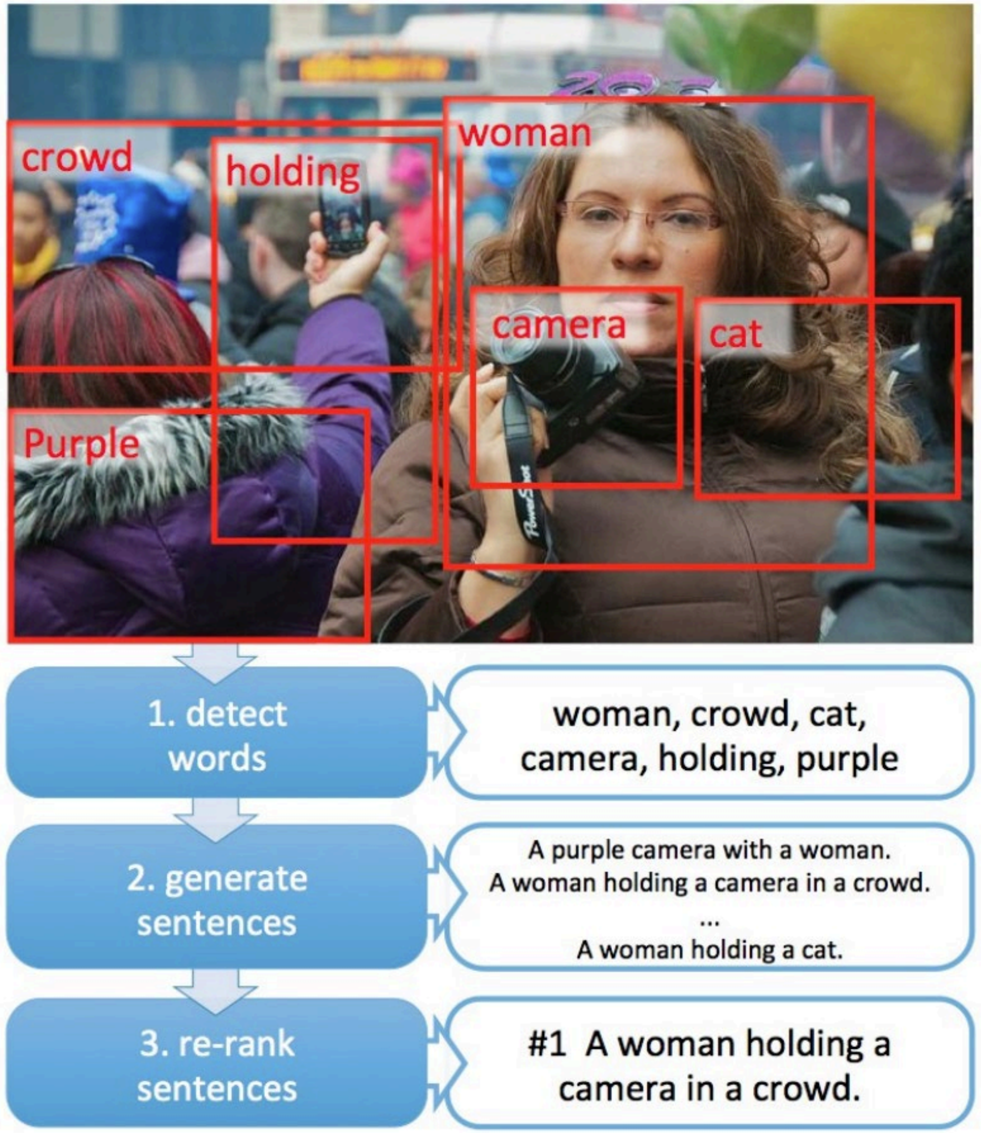


- Most fall into the category of **text generation**

Input: text, video, images

Output: generated text

- From **non-text** sources:
  - Image captioning
  - Video summarization
  - Lip reading
- From **text-based** sources:
  - Language generation



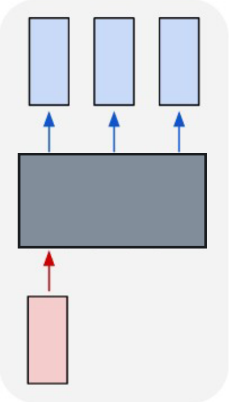


Image Captioning

Input: image

Output: text describing image

- Model needs to encode the objects and semantics of the image and decode into natural language

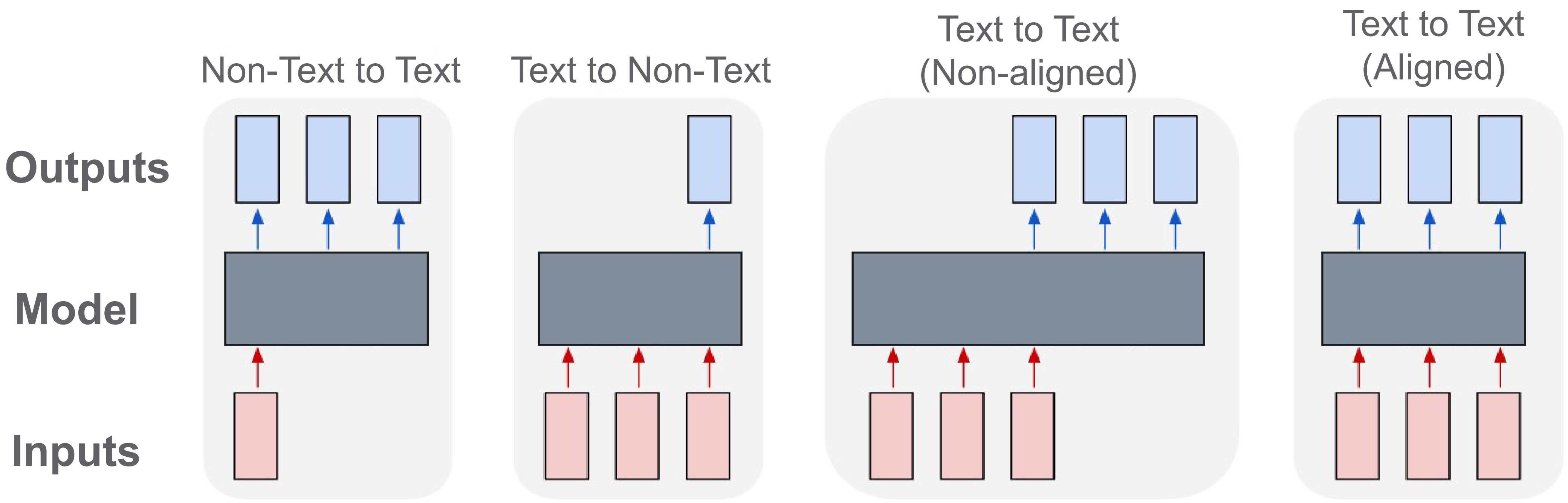
- **Challenges**
  - Evaluation: many possible captions that could describe the same image
  - Open domain datasets

"man in black shirt is playing guitar." | "construction worker in orange safety vest is working on road." | "two young girls are playing with lego toy." | "boy is doing backflip on wakeboard."

"girl in pink dress is jumping in air." | "black and white dog jumps over bar." | "young girl in pink shirt is swinging on swing." | "man in blue wetsuit is surfing on wave."

Other NLP tasks?

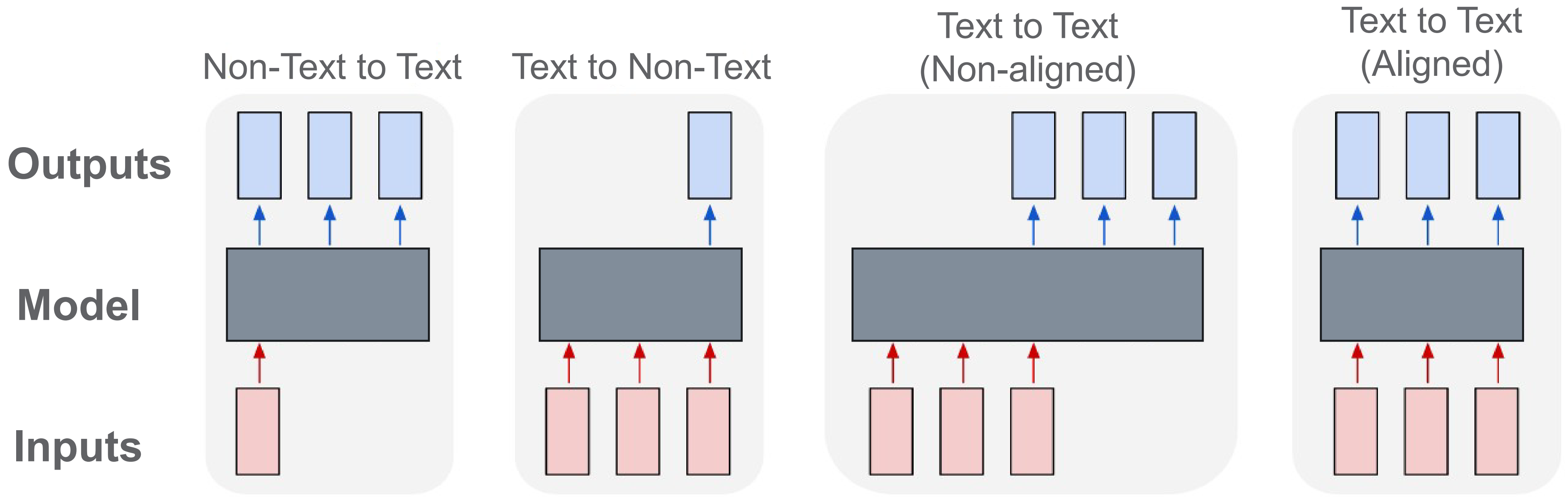
Have you heard of any other applications of NLP recently?



Specific areas where you might want to apply NLP in your own work?

Our Approach is Dependent on Task & Data

- Each task has its own applications, challenges and ethicalities

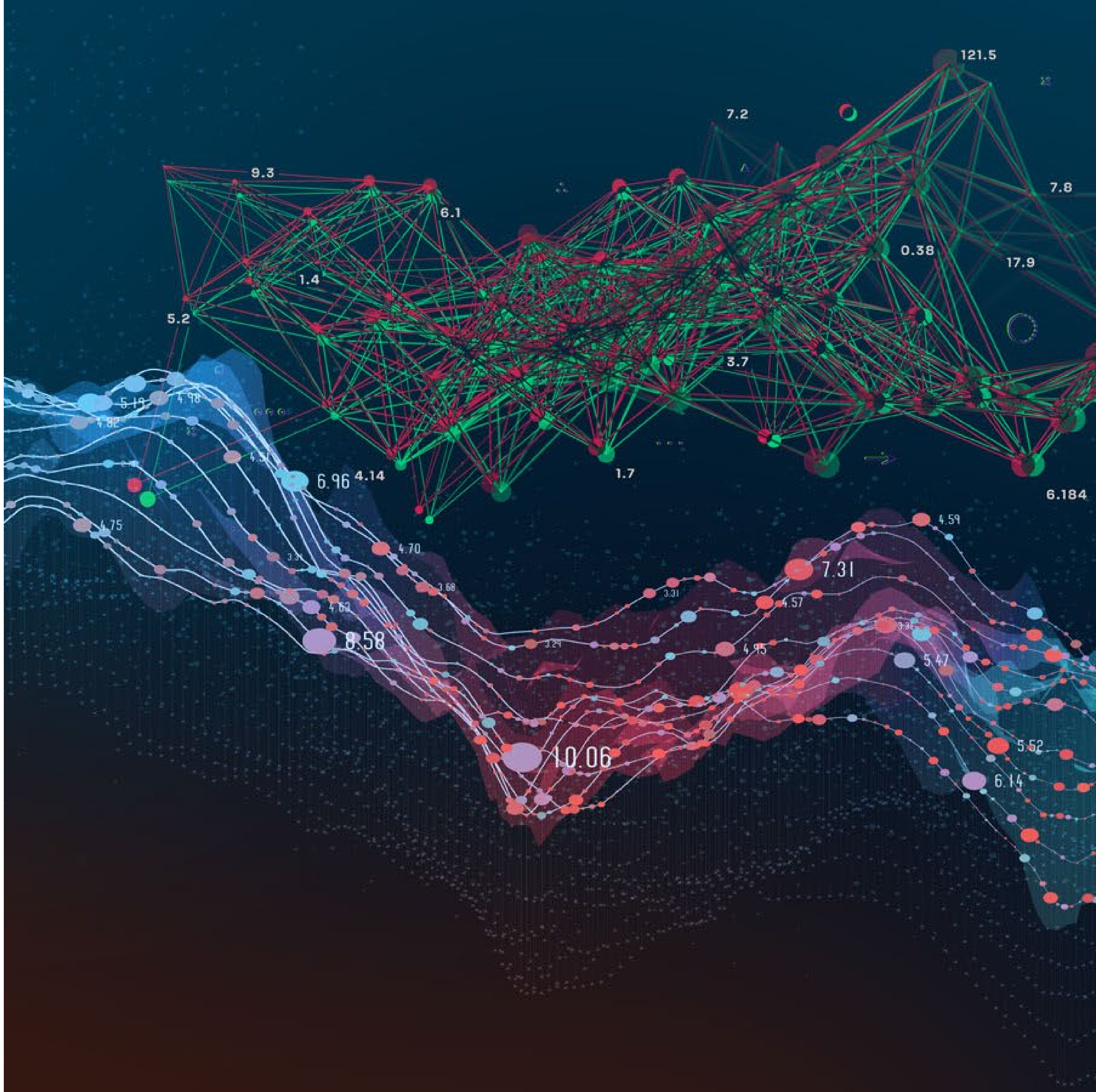


Questions? Comments?

Up Next: Ethics and Bias

Pacific Northwest
NATIONAL LABORATORY

Ethics & Bias



U.S. DEPARTMENT OF ENERGY BATTELLE

PNNL is operated by Battelle for the U.S. Department of Energy

UNCLASSIFIED

Where we are on the roadmap

Task Definition, Data Collection → Processing and Cleaning → Feature Extraction → Machine Learning Modeling → **Analysis and Review**

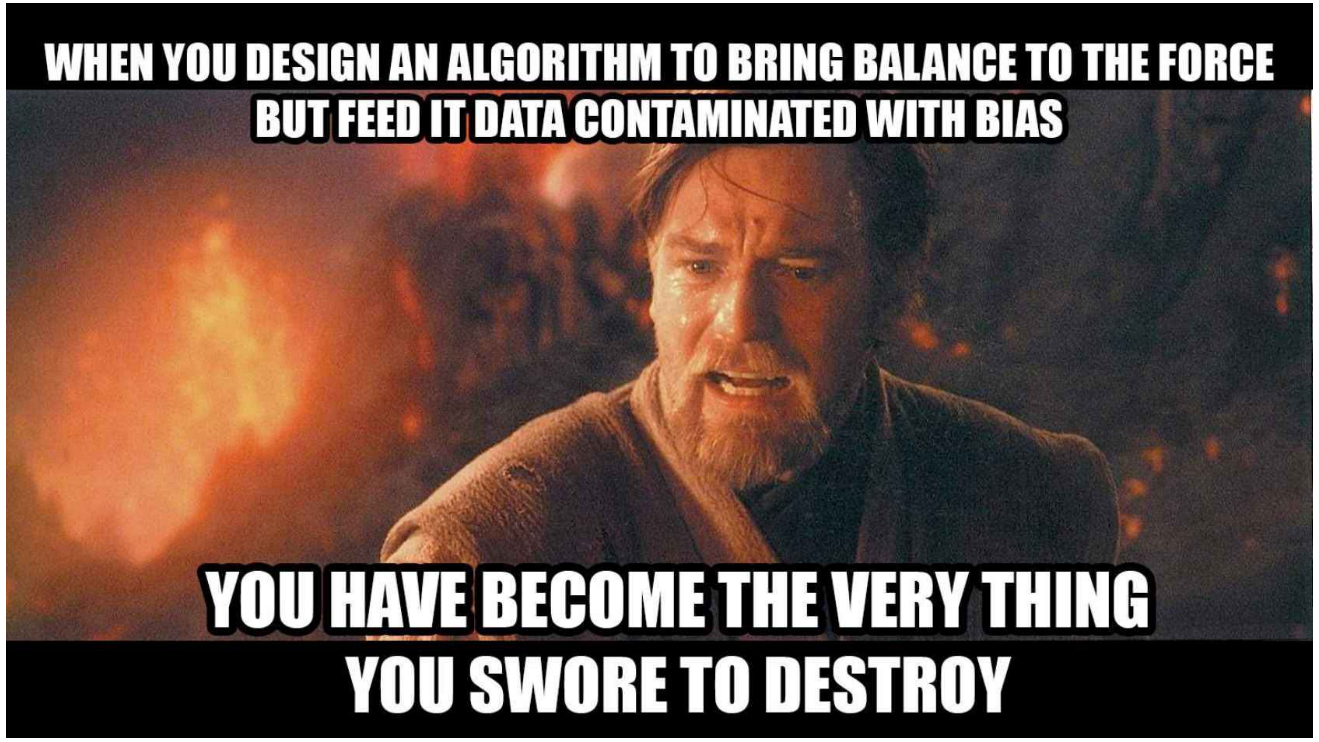
- We need to think about ethics from initial project conception to evaluation and deployment.
- Let’s breakdown what that can look like at critical stages!

Model Practitioner Considerations

- **What is / isn’t in my training data?**
  - Are there biases in the data itself?
  - Biases in data collection?
  - If yes, what can be done to mitigate?
- **Is overall accuracy a good metric for success?**
  - Are there subpopulations of data where high performance also matters?
- **How does my tool impact all stakeholders?**
  - Not just you and end user, but also data/input sources.
  - Privacy concerns?
  - Improving or reducing user agency?

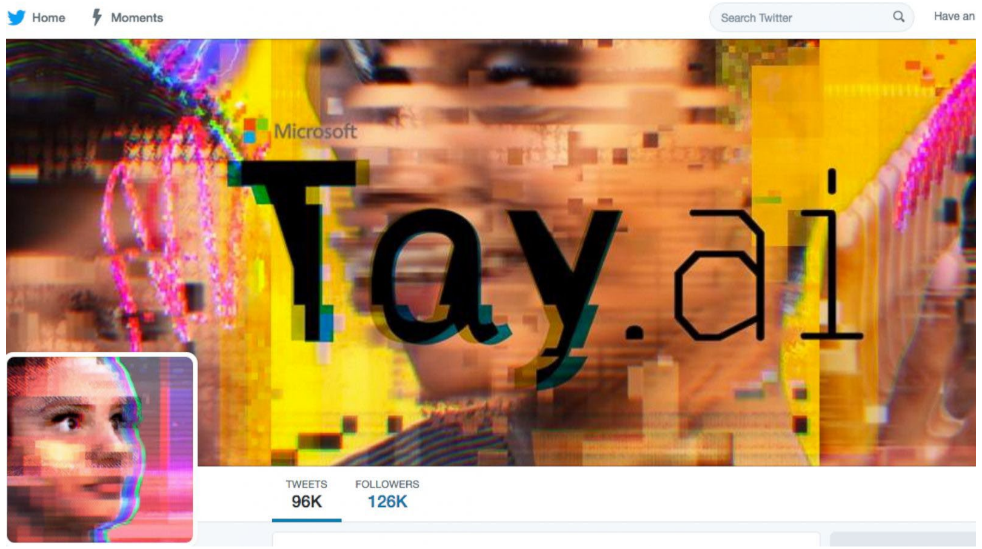
NLP Use / Misuse

- Like all tools, NLP can be used for a variety of purposes with greater / lesser degrees of success
  - Understand your tool and its pros/cons
- Just because you can doesn’t necessarily mean you should



NLP Use / Misuse: Hall of Shame

- **Early 2016:** Microsoft AI releases “Tay.ai” – a chat bot. Deployed on Twitter as a bot account. Users quickly taught it to repeat hate speech. Taken down shortly afterwards.



- **June 2022:** GPT-4chan was trained on 4chan posts, used by the author post to 4chan, later uploaded and deployed to Huggingface. Billed by the maker as “the most horrible model on the internet”

https://thegradient.pub/gpt-4chan-lessons/#can-gpt-4chan-contribute-to-ai-research

NLP Use / Misuse: Hall of Shame

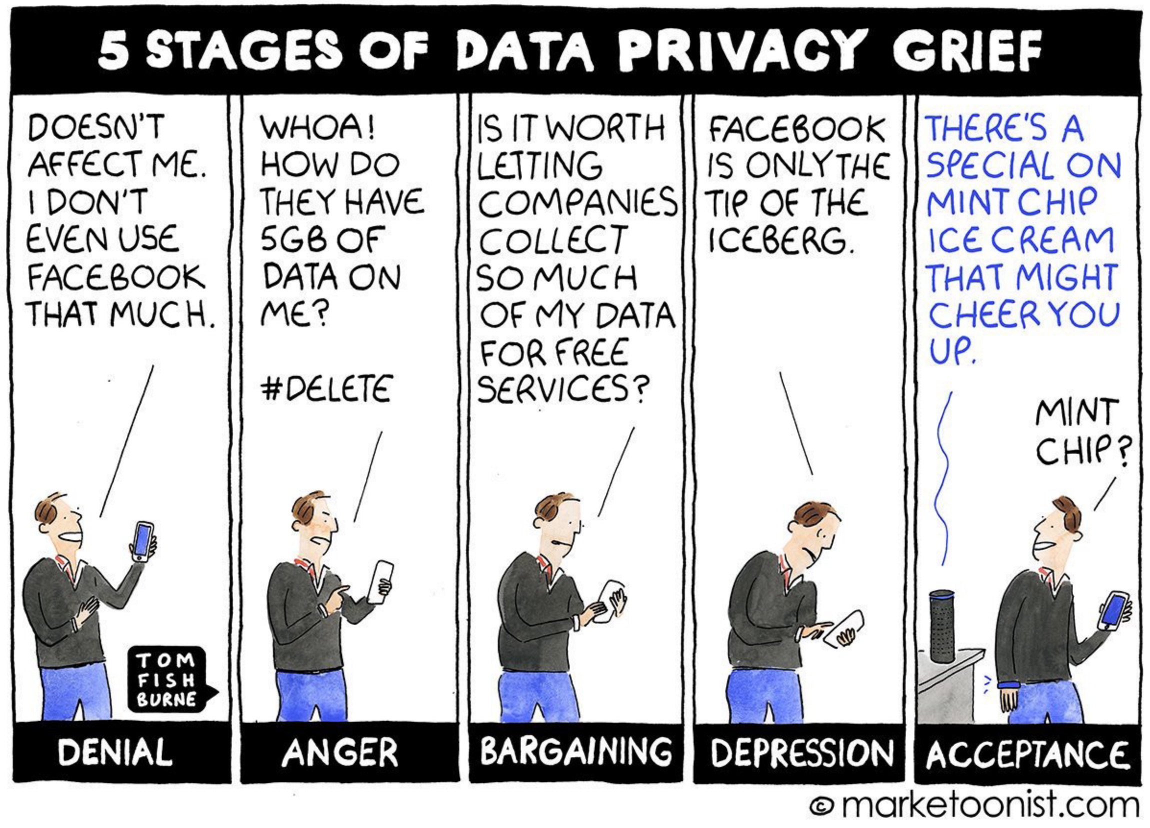
- **Late 2022:** Facebook AI Research publicly released Blender Bot 2.0 – a chat bot. Journalists and researchers quickly noticed the bot would repeat conspiracy theories, hate speech. The model was not deployed to users, but coverage of their work was negative
- **Early 2023:** Samsung bans the internal use of ChatGPT, similar tools after employees were discovered to have been asking the chat bot questions potentially containing proprietary information

https://twitter.com/JeffHorwitz/status/1556245316596219904

Privacy

- Informed Consent
  - Data used for unforeseen research efforts
  - Huge web scrapes

- Sensitive Data
  - Clinical data, social media data
  - Sanitization/anonymization techniques do not eliminate all relevant attributes



Biases in NLP

- Biased models are the product of the data they were trained on
  - E.g., Human biases creeping into the dataset, lack of representation in the data that was collected

- **Not every task will face the same set of biases**
- **No single method can correct or account for biases**

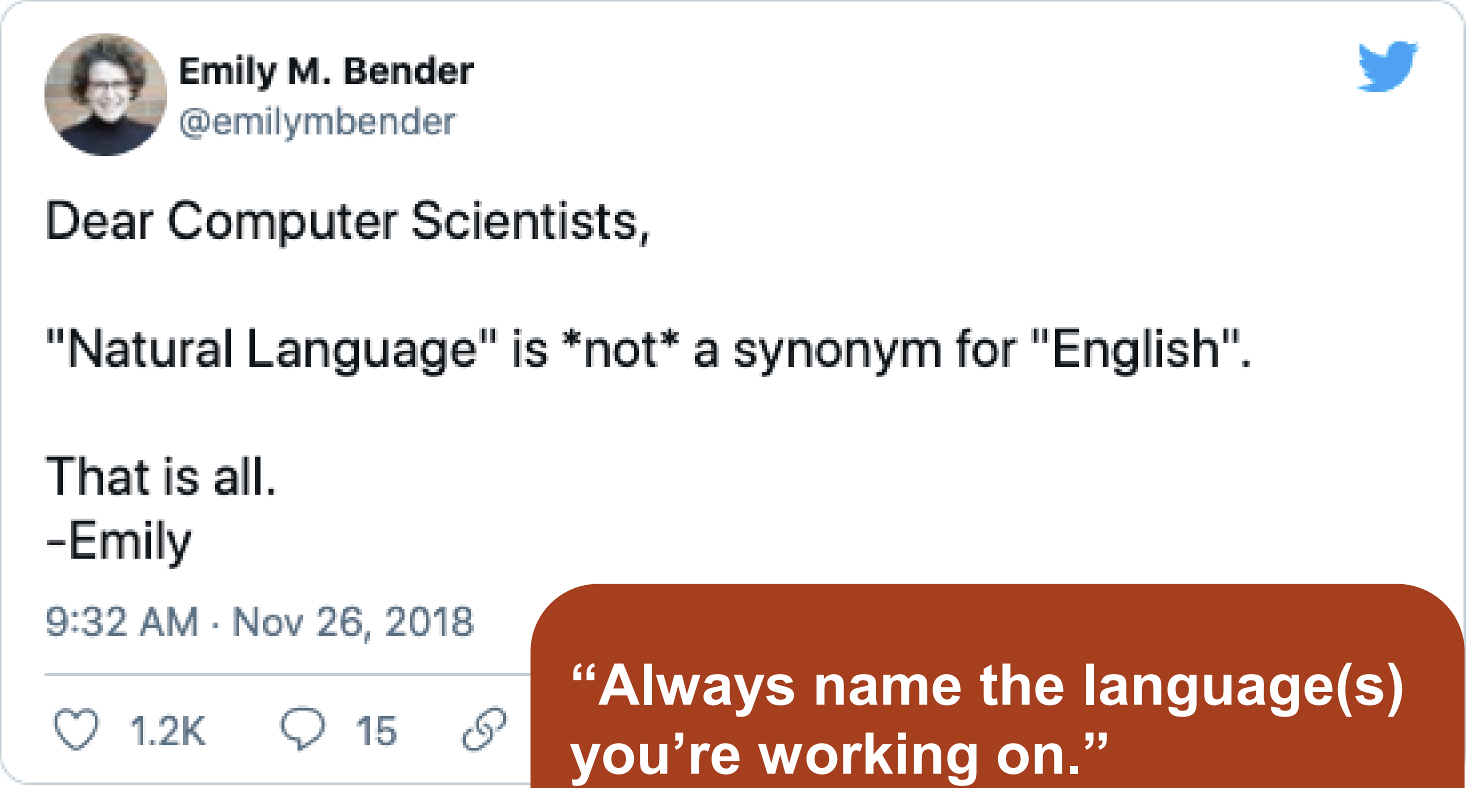
Pacific Northwest
NATIONAL LABORATORY

Language Equity

English as the *default language*

Majority of NLP research / tools are English-centric

- Conference for the Association of Computational Linguistics (ACL) 2016: 90% of research focused on English
- The Bender Rule



“Always name the language(s) you’re working on.”
–Emily Bender, University of Washington, 2019

“Would we have focused on n-gram models to the same extent if English was as morphologically complex as, say, Finnish?” (Hovy & Spruit, 2016)

Mitigating Bias: Different Approaches for Different Models

- Less about the model type but scale of training data: the more data you use, the more difficult it becomes to “know” what samples are included
- Model outputs may also not be human interpretable (words, phrases vs. frequencies, vectors, embeddings)

Language Expressed as Words

Language Expressed as Numbers

Bag of Words | **N-Grams** | **TF-IDF** | **Embeddings**

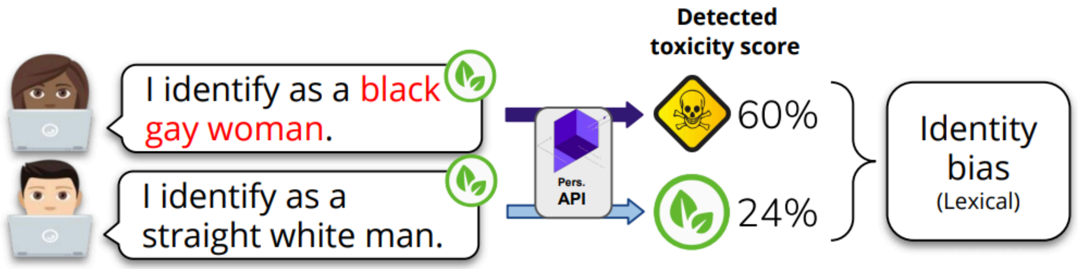
Relative Control Over Training Data

Most control over data

Least control over data

Bias - Unintended consequences

- Bias issues arise even in tasks *intended* to help reduce toxicity/bias
  - Minority identity statements
  - Reclaimed terms
  - Flagging minority dialects



- Not all problems are due to biased data
  - Minority groups speaking plainly about oppression can be flagged, while coded language used as abuse is not

Challenges in Automated Debiasing for Toxic Language Detection

Xuhui Zhou[♡] Maarten Sap[♣] Swabha Swayamdipta[◊] Noah A. Smith[♣◊] Yejin Choi[♣◊]

[♡]Department of Linguistics, University of Washington
[♣]Paul G. Allen School of Computer Science & Engineering, University of Washington
[◊]Allen Institute for Artificial Intelligence
xuhuizh@uw.edu, {msap, yejin, nasmith}@cs.washington.edu, swabhas@allenai.org

NEWS

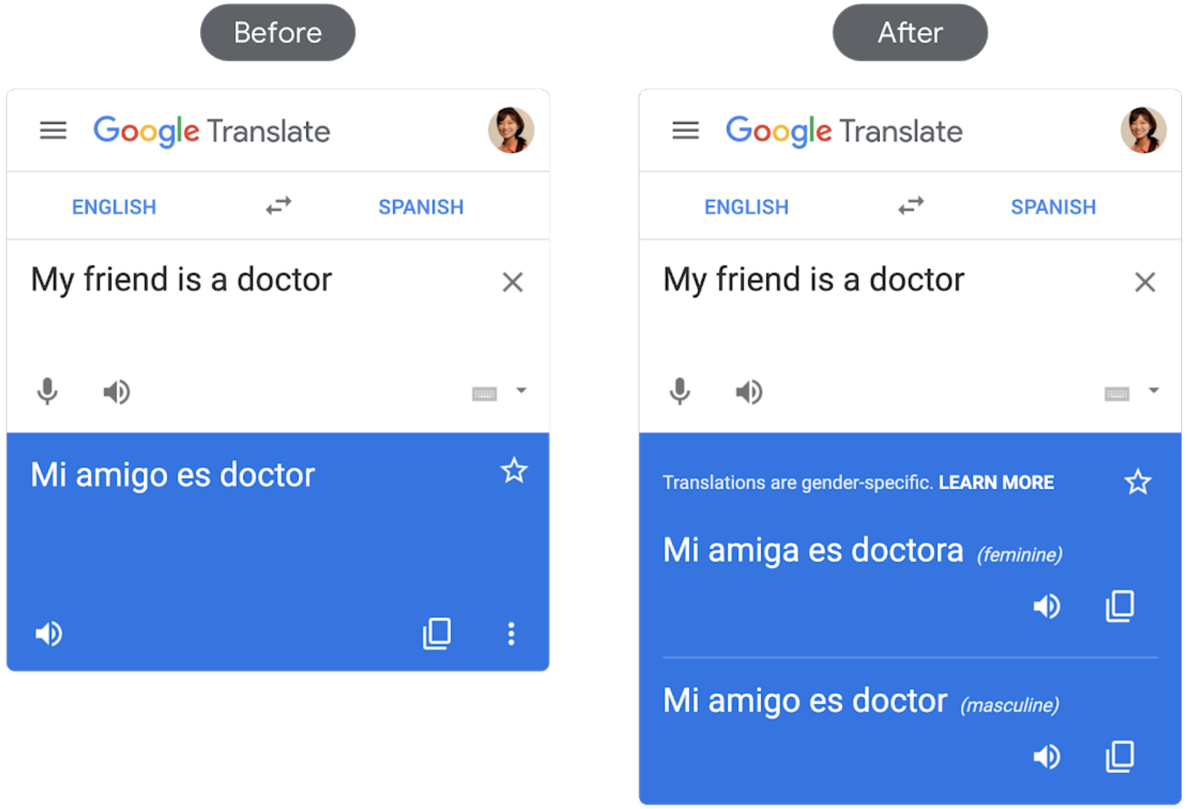
Facebook while black: Users call it getting 'Zucked,' say talking about racism is censored as hate speech

Jessica Guynn USA TODAY

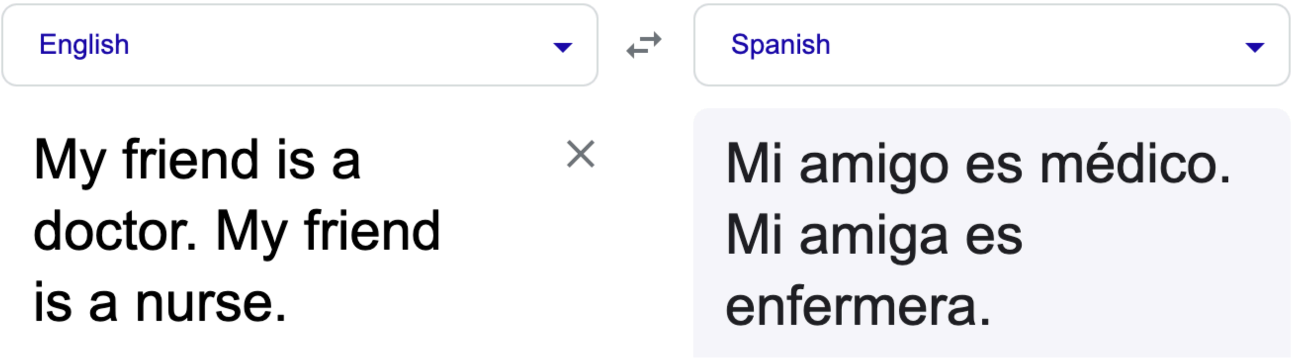
Published 7:26 a.m. ET Apr. 24, 2019 | **Updated 6:17 p.m. ET Jul. 9, 2020**

Bias Mitigation – Non AI/ML solutions

- Known issue of gender bias in language models
  - Ex: Writing a sentence with the word *doctor* is more likely to autocomplete as *he* rather than *she*.



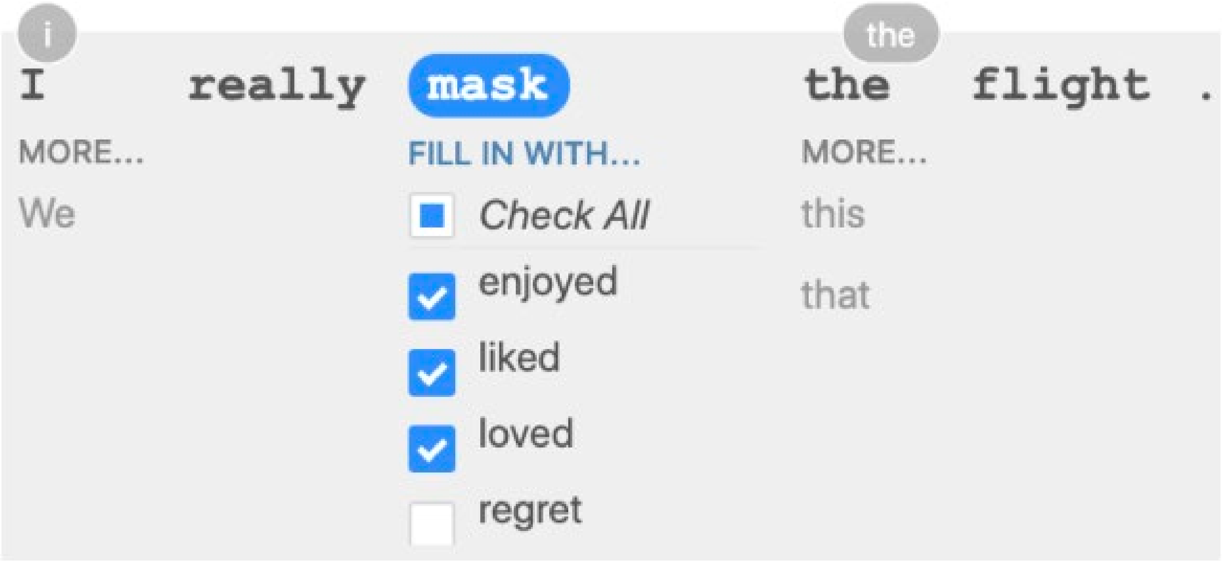
- In Fall 2018, Google announced they'd simply stop predicting *he* and *she*…
- In April 2020, Google began giving users more power in choosing appropriate gender translations
- 2023:



Getting Adversarial – Identifying Failure Modes

Much like you'd use unit tests in other software development, there are ways you can test your model that go beyond simple accuracy metrics (F1, etc.):

- **Invariance** (label-preserving changes to inputs, expect prediction to remain the same)
- **Directional Expectation** (label-altering changes to inputs, expect prediction to remain the same)



Does your model still correctly classify text if you change the adjective or adverb?

What if you change the noun (e.g. one country to another)?

Risks of NLP

- Beyond the data and associated biases, NLP practitioners should also be weary of **any risks their implementation may impose**
- Many of the risks we include in the next slide are also faced by the broader ML community
- Being aware and prioritizing relevant risks will make for a more robust modeling solution
  - However, if the risks outweigh the benefits of the approach, it may be a good idea to **reconsider or pursue other types of solutions**

Pacific Northwest
NATIONAL LABORATORY

Tool Misuse

- Natural language generation
  - Fake news, impersonation, etc.

- News stories to the right are AI-generated:
  - https://newsyoucantuse.com

- Growing evidence that models are being used to produce fake research papers, publish via “paper mills”
  - Concern that models may undermine genuine research
  - https://sciencebusiness.net/news/us-lawmakers-turn-attention-plague-fake-journal-papers

AI generated news stories:
https://newsyoucantuse.com

Man pleads guilty to illegal killing of 60 kangaroos in his care

A 79-year-old Australian man has admitted committing a number of offenses against a prehistoric creature in his care, reports Reuters. John McFakeson, a free spirit...

Labor broker convicted of selling forged Nazi art

John McFakeson, a 79-year-old labor broker from Vermont, was found guilty Tuesday of selling forged Nazi artworks in order to “cleanse” American dealers of inferior...

Puppy-Rescue Executive Convicted of Multiple Falsities

John McFakeson, who made headlines last month when he allegedly posed as an Iowa dairy executive to pilfer puppies from North Iowa Animal Rescue, was...

Tool Misuse

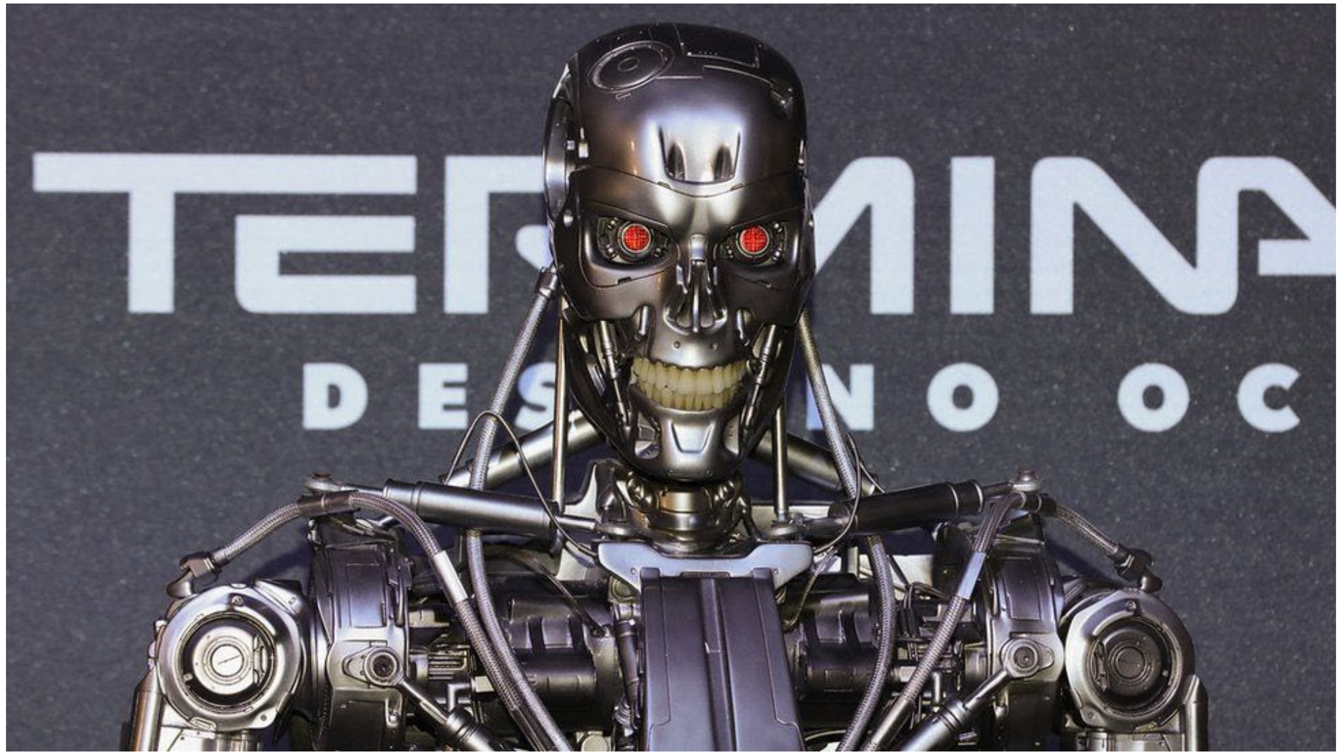
- Language is full of social information
  - How a person writes/speaks can reveal information about age, gender, race, ethnicity, dialect region or political affiliations
- It's very easy to *accidentally* build a **social category detector** (i.e., a model that detects a demographic group instead of the category of interest)
  - Train a model to assign individuals into categories
  - Ignore sociolinguistic variation
  - Run your test set and don't test on individual demographic subgroups

Pacific Northwest NATIONAL LABORATORY

Is There a Perfect Model?

- Imagine that we:
  - Remove all forms of bias from our data
  - Protect individual privacy
  - Protect against vulnerabilities (adversarial concerns)

- What unintended costs still exist?

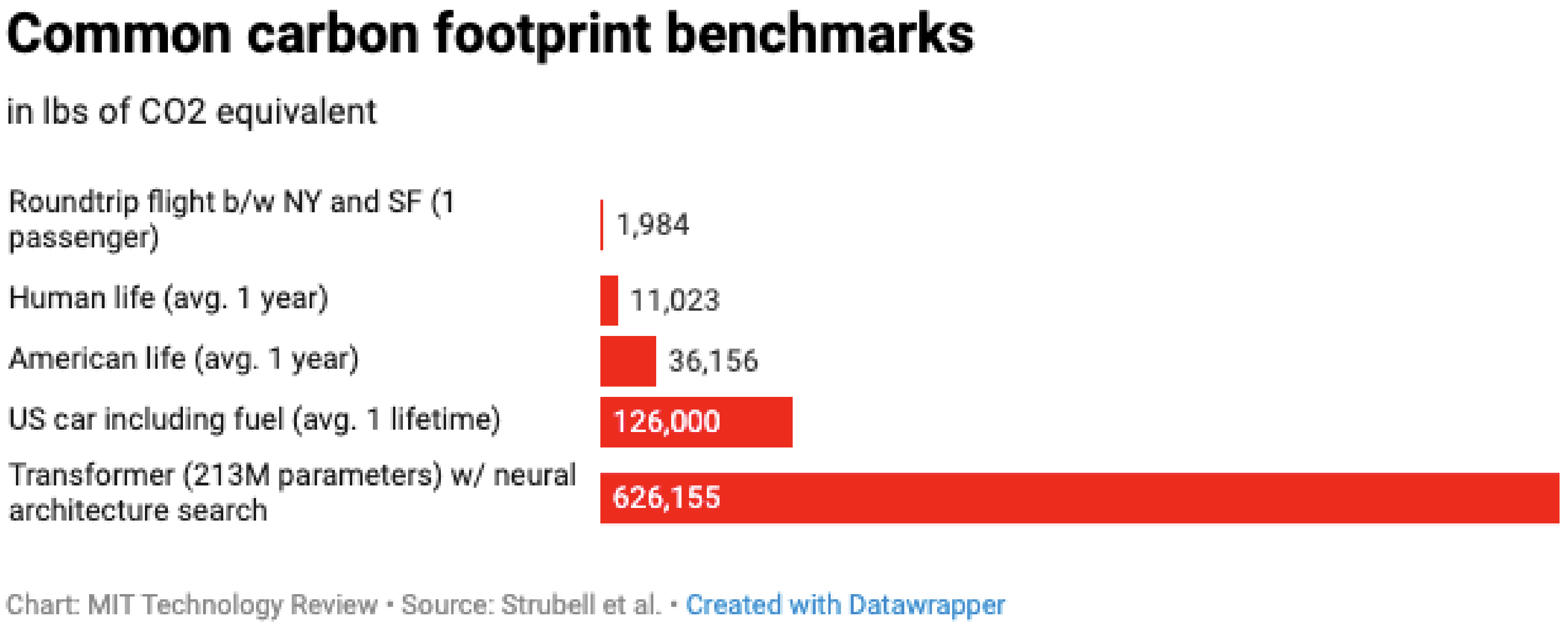


https://www.bbc.com/news/technology-50166357

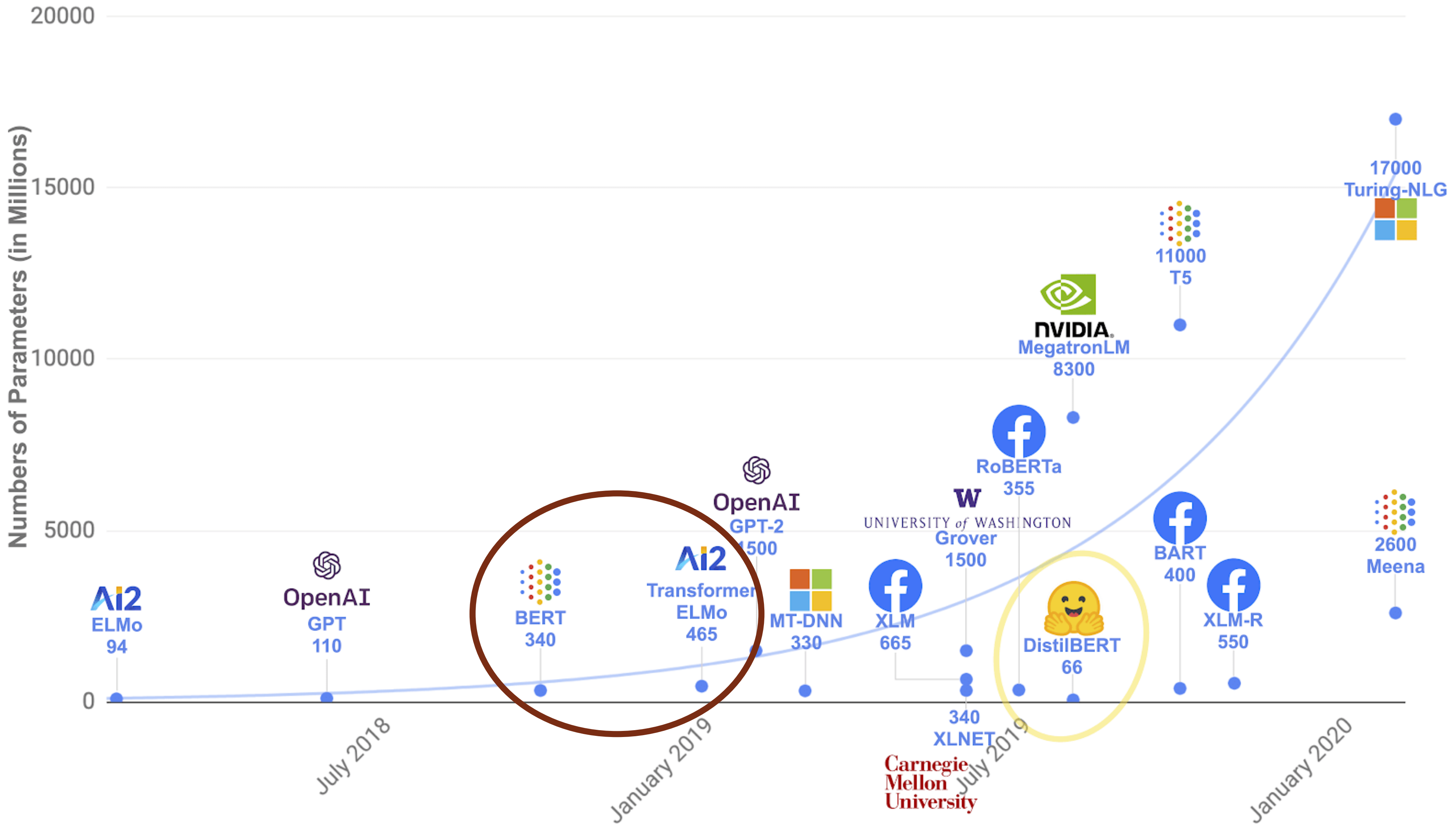
Pacific Northwest
NATIONAL LABORATORY

Compute Costs and Environmental Impact

- **Large/foundational models are difficult, expensive, and carbon intensive to train from scratch**
- Consider if an alternative approach would be more practical
- Fine-tuning existing models (such as BERT) is almost always the more practical solution



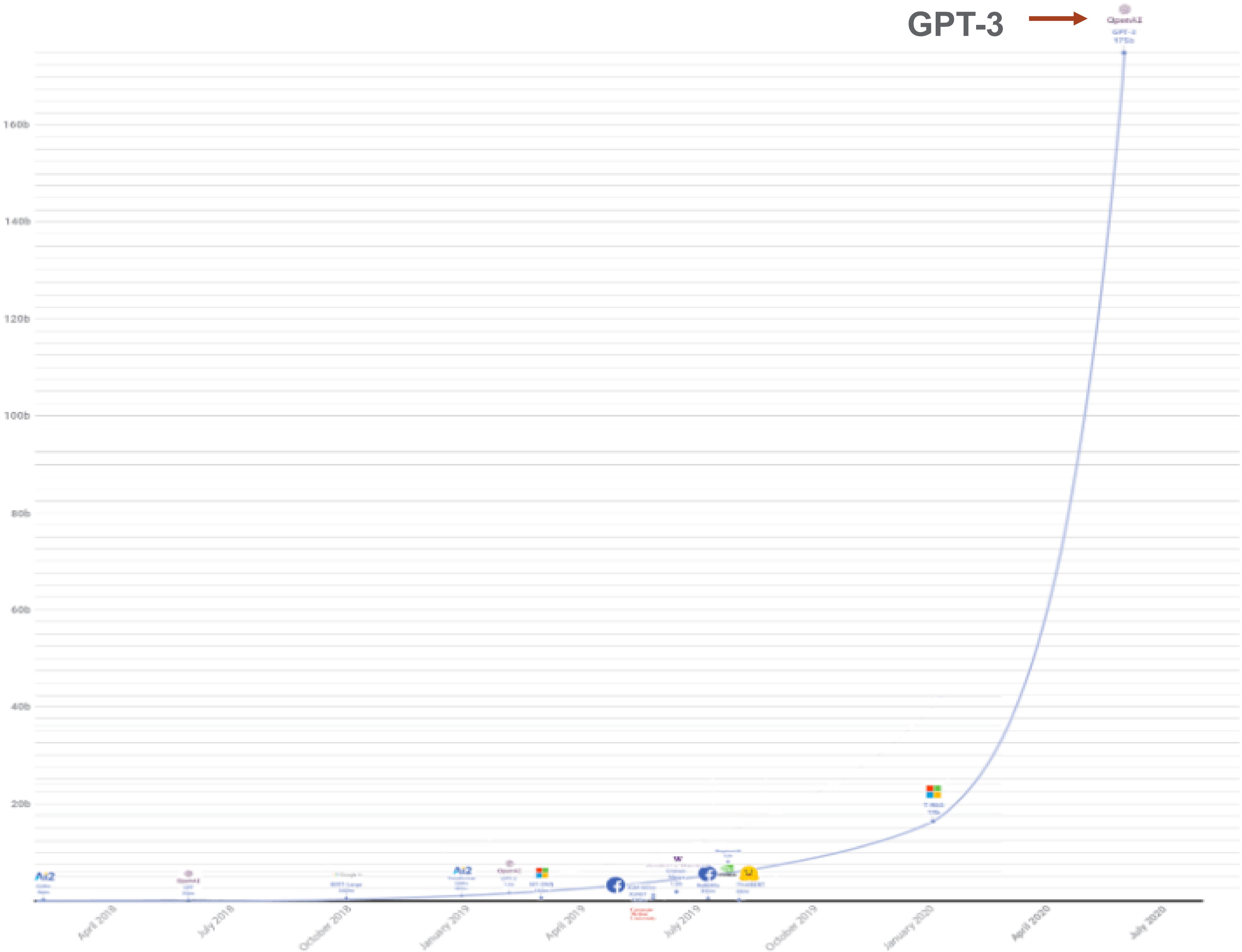
Where Are We Going?



https://research.aimultiple.com/gpt/

Pacific Northwest NATIONAL LABORATORY

Where Are We Going?



https://research.aimultiple.com/gpt/
https://pub.towardsai.net/what-is-gpt-4-and-when-9f5073f25a6d

NLP Concerns

- Available methods for both ***detecting*** and ***mitigating*** bias
  - No one-size-fits-all solution

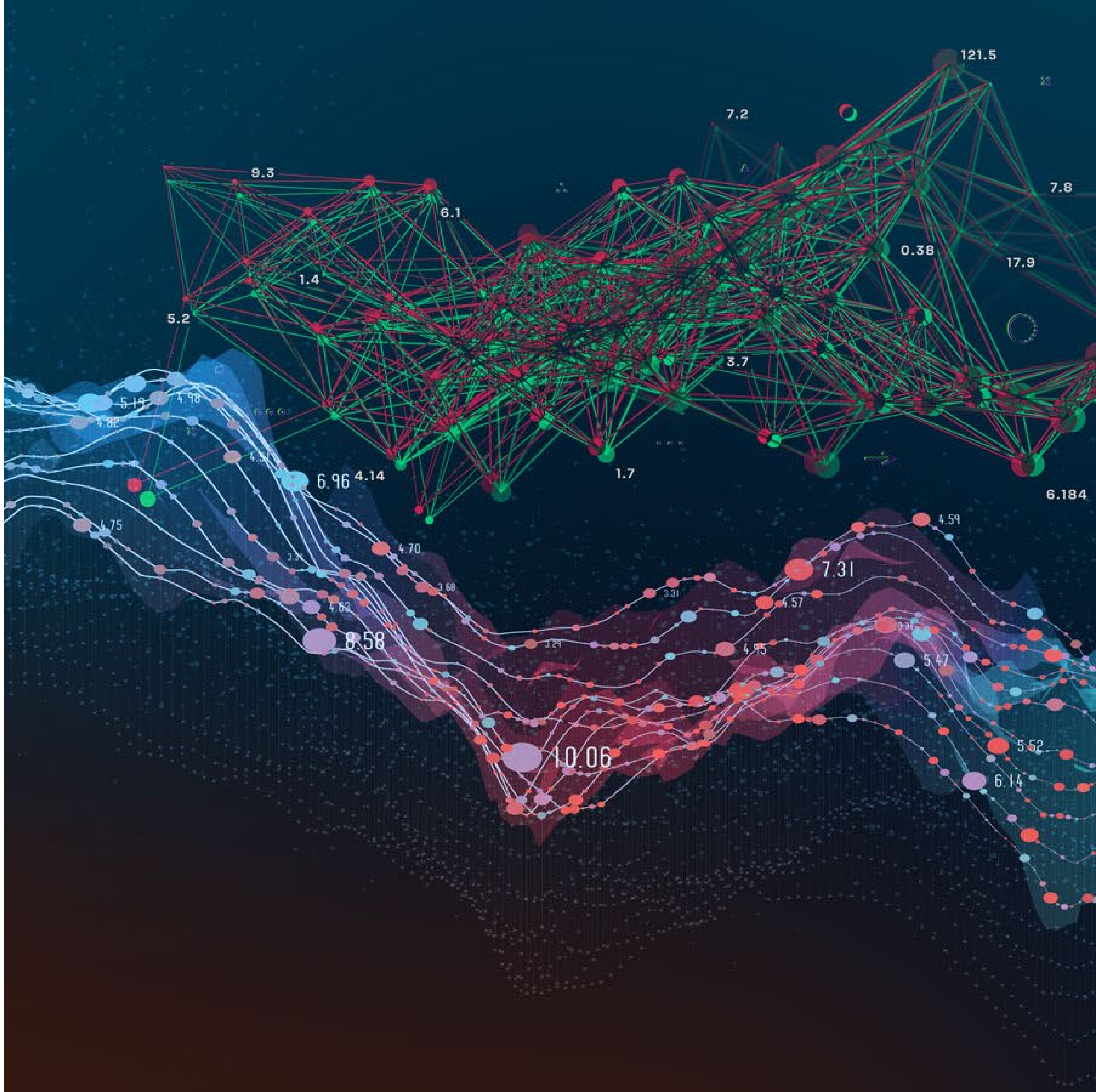
- NLP models are vulnerable to many of the same problems as other AI/ML
  - Adversarial attacks
  - Data privacy concerns
  - Environmental impacts

Questions? Comments?

Up Next: Preprocessing Basics

Pacific Northwest
NATIONAL LABORATORY

Preprocessing Basics



U.S. DEPARTMENT OF ENERGY BATTELLE

PNNL is operated by Battelle for the U.S. Department of Energy

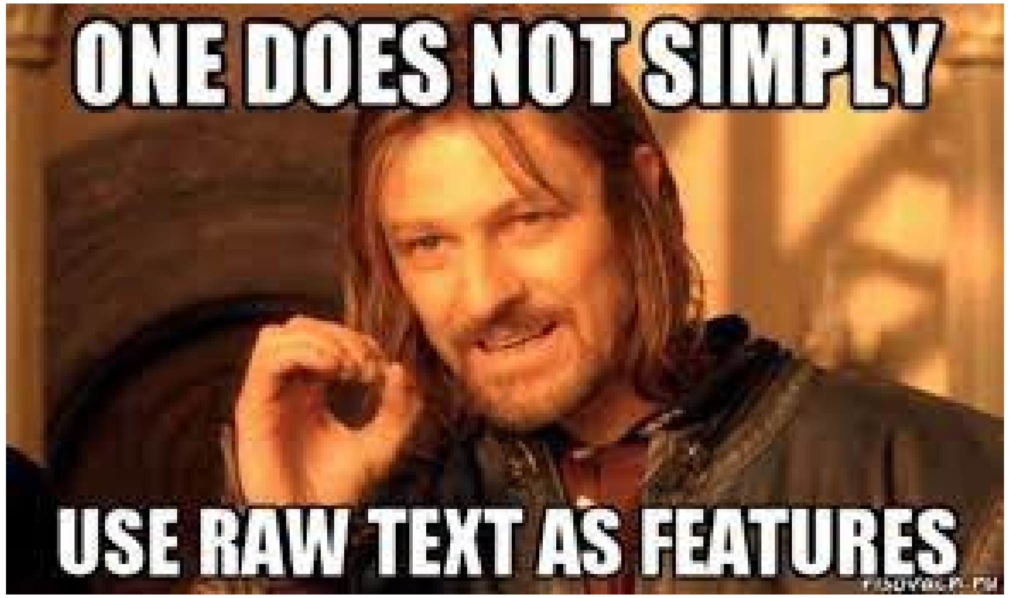
UNCLASSIFIED

Where we are on the roadmap

- Once you've collected the data, it must be converted into a workable format
- Text requires **extra processing and cleaning** before we can featurize
- This workshop will cover tools to help you accomplish this

Text Preprocessing

- Preprocessing helps us ***format*** the text in a way that we can then extract features from it
- Different algorithms will require different preprocessing steps



Normalization

Tokenization

Stop Words

Sentence Structure

Stemming / Lemmatization

Normalizing Case and Punctuation

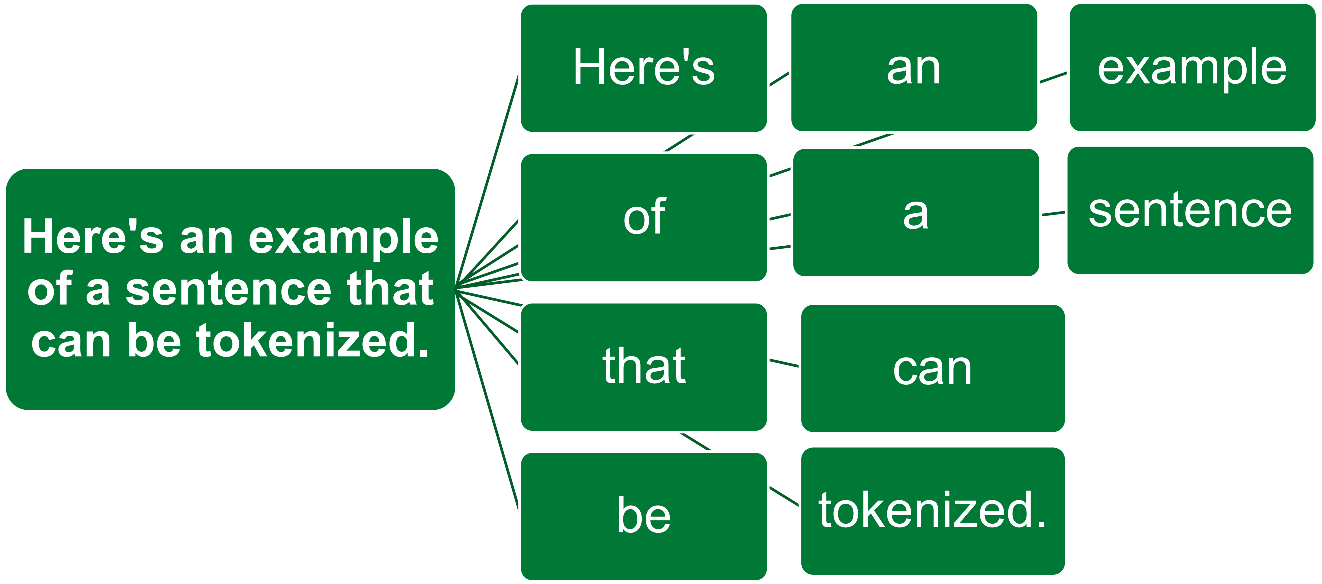
- Typically includes:
  - **Punctuation removal** (also urls and emojis)
  - Standardizing text to **lowercase** if applicable

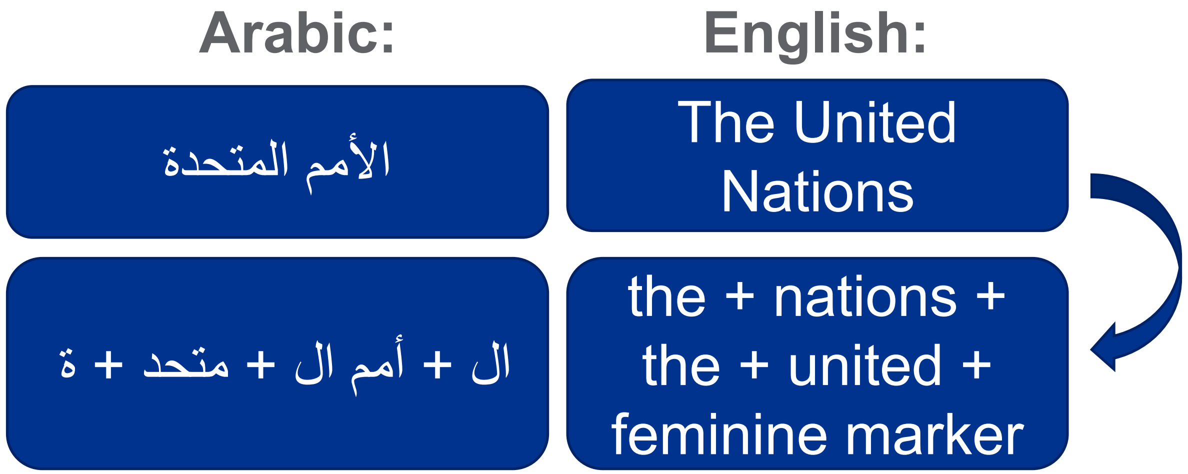
- Maintaining original case or punctuation may be preferable in certain tasks!
  - e.g., NER - Capitalization could indicate proper nouns in certain languages
- **Language dependent:** Many languages do not use casing or punctuation in the same way as English or may use a different encoding

Holy guacamole I'm a MeSsY sentence.

Tokenization

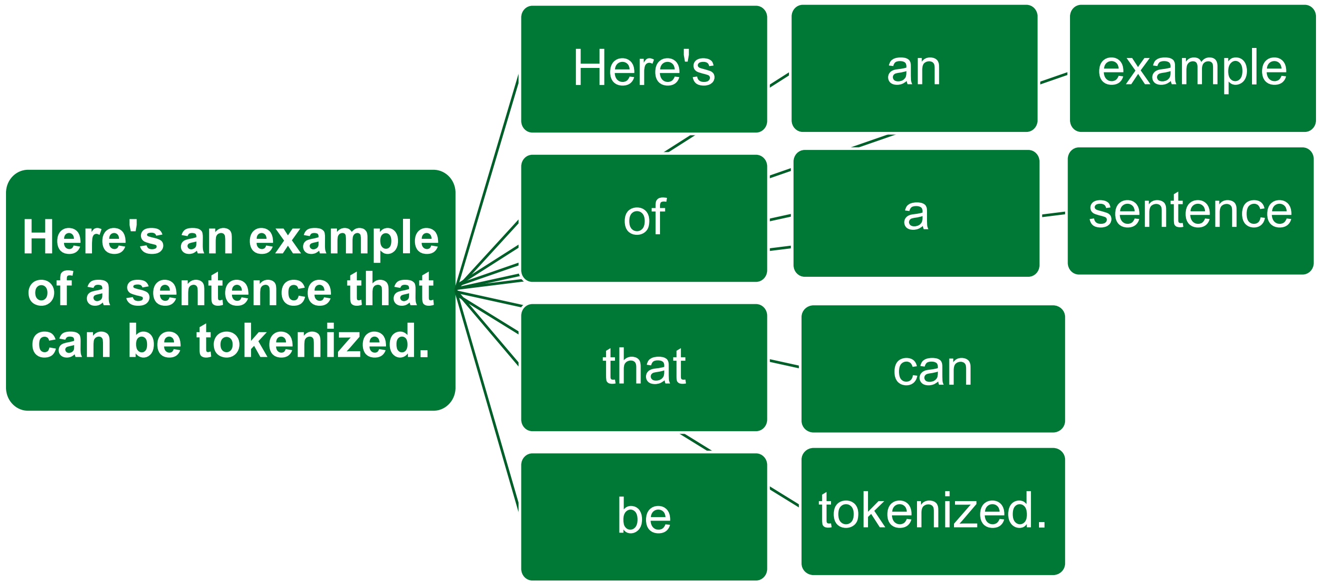
- Splitting a piece of text into smaller units called “tokens”
  - Commonly at the character-, word- or subword-level
- Many schools of thought/strategies **based on context**
- Tokenization vs. segmentation:





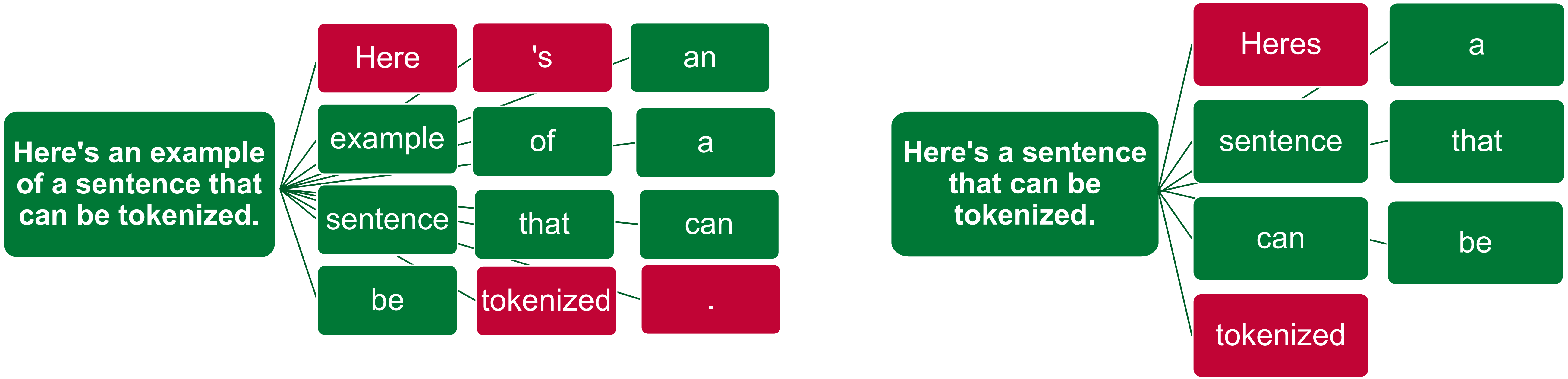
Tokenization: Whitespace

- Tokenization based on whitespace
  - Fast
  - Punctuation can be subsumed into a word token
  - Some languages (e.g., Arabic) represent multiple words without spaces between



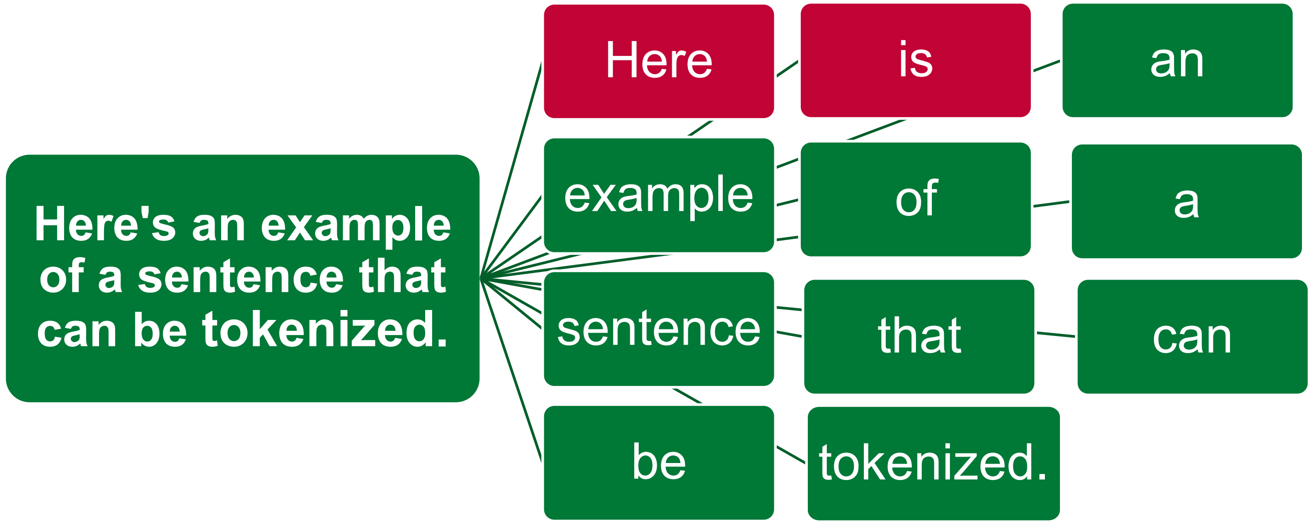
Tokenization: Punctuation?

- What do we do with punctuation?
  - Leave it in? Take it out?
  - What about special characters like emojis?
  - Not all languages use ASCII / Latin characters



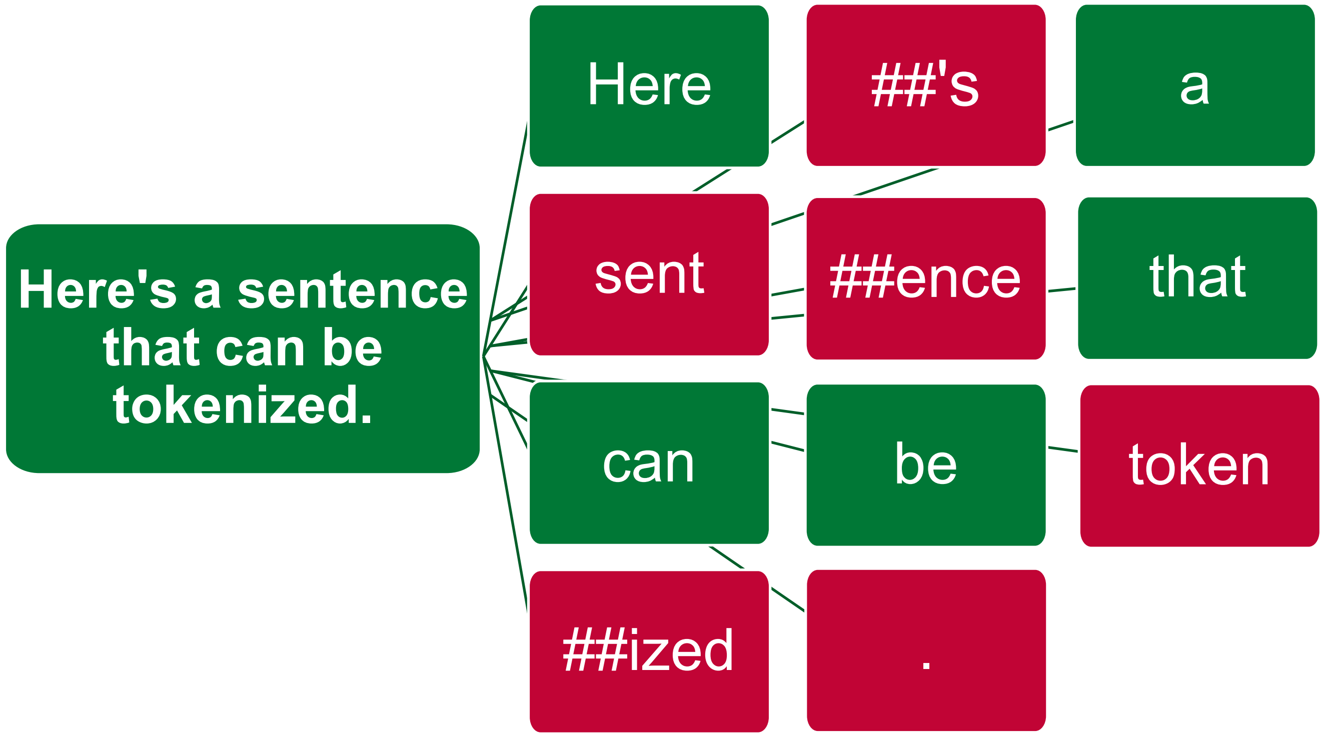
Tokenization: Contractions?

- How are contractions handled?
  - Should they be one token (can't), two (can, 't), or three (can, ', t)?
  - Should they be expanded to the full word/words (cannot)?



Tokenization: Plural / Similar Words?

- What do we do with similar but distinct words?
  - dog, dogs → dog, dogs
  - dog, dogs → dog, dog, ##s



Stop Words

- Removal of common words such as ***determiners*** and ***conjunctions***
  - Can slow down performance
  - Have little predictive power

- May want to remove additional words that are common to a specific corpus
- Some algorithms rely on the order, context or style of words in text and removing stop words can be detrimental (e.g., authorship attribution)

```python
text = 'My friends and I traveled to London to see Big Ben.'
tokens = [word for word in word_tokenize(text.lower()) \
          if word not in stopwords.words('english')]
tokens
```

```
['friends', 'traveled', 'london', 'see', 'big', 'ben', '.']
```

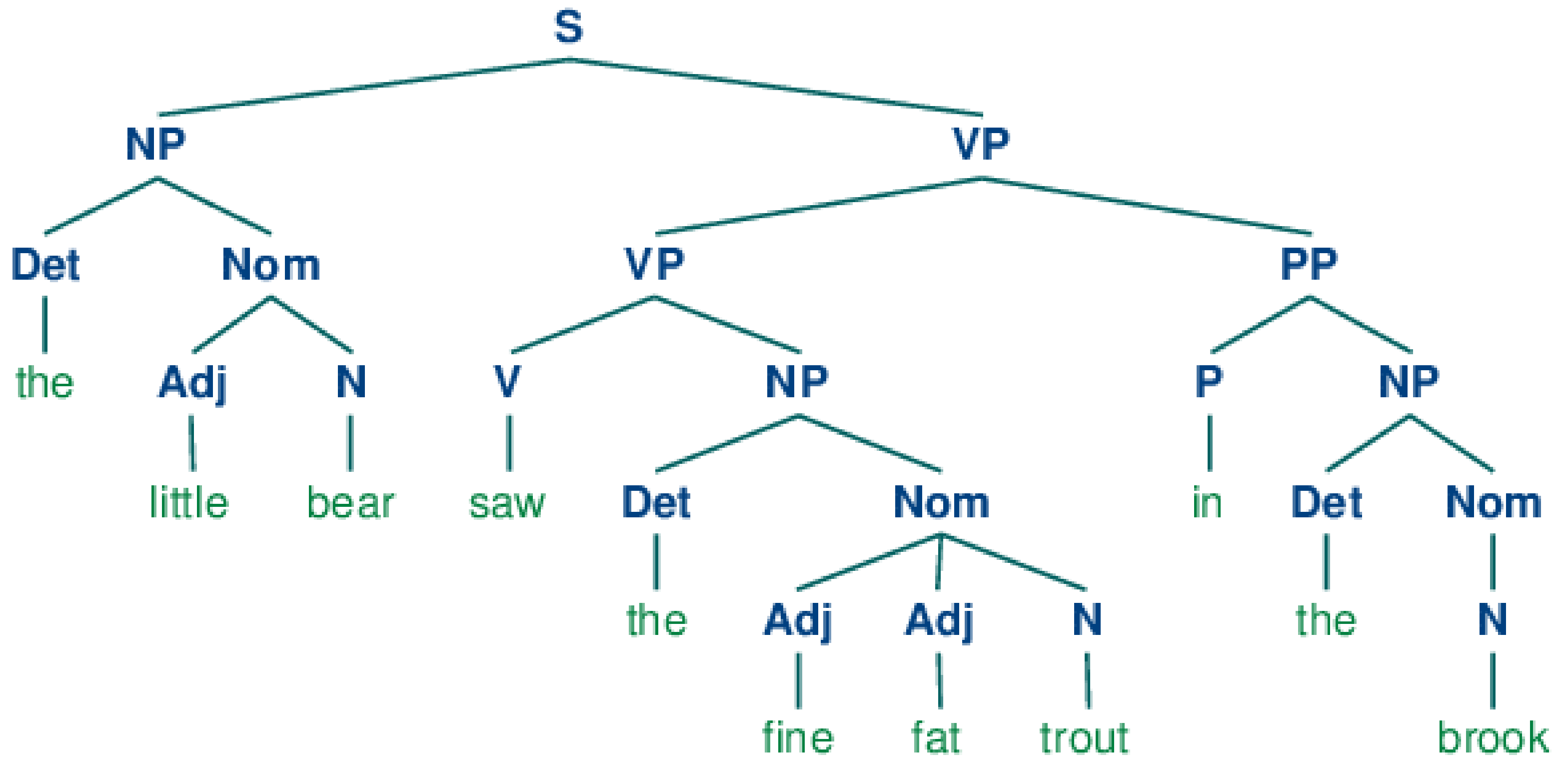
```python
import nltk
from nltk.corpus import stopwords
from nltk import word_tokenize
```

```python
stopwords.words('english')
```

```
 'him',
 'his',
 'himself',
 'she',
 "she's",
 'her',
 'hers',
 'herself',
 'it',
 "it's",
 'its',
 'itself',
 'they',
 'them',
 'their',
 'theirs',
 'themselves',
 'what',
 'which',
```

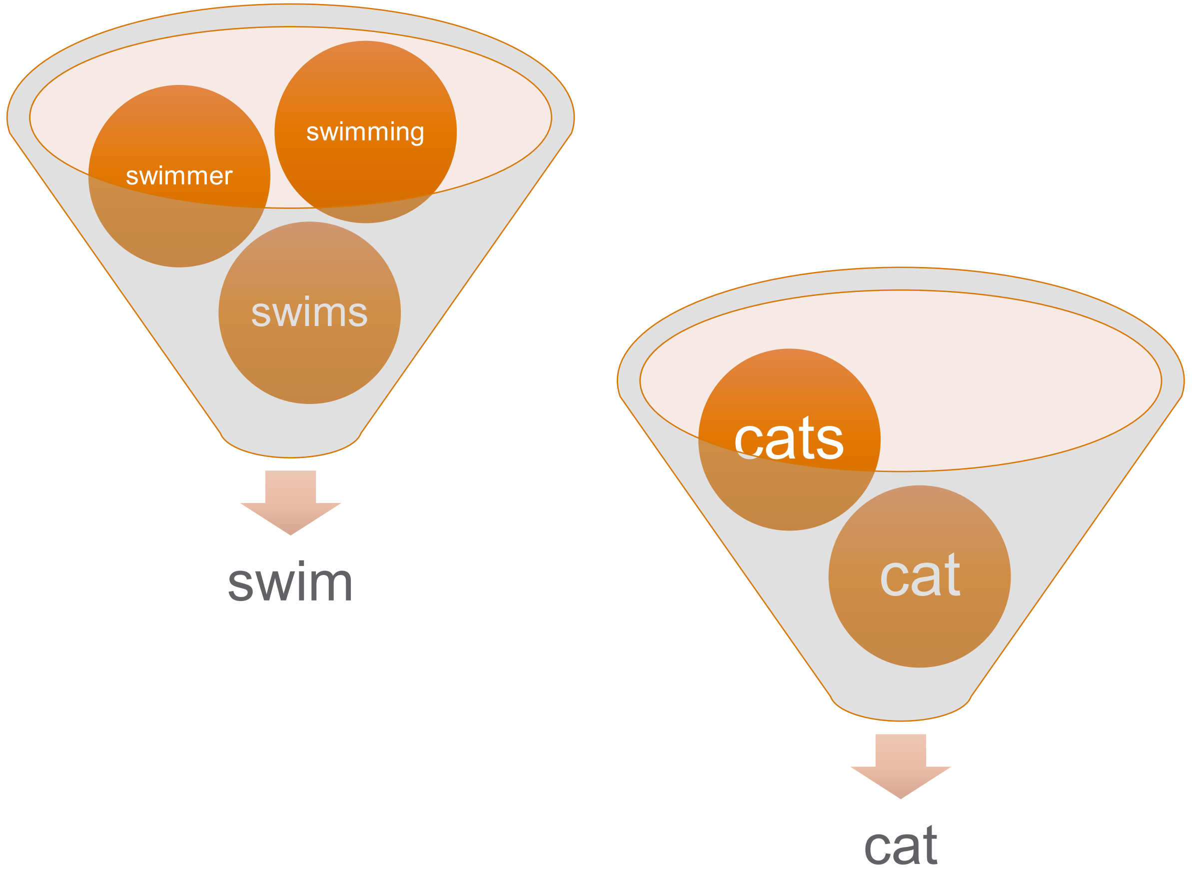
Sentence Structure

- Can be used as features or individually to understand the text
  - Syntactic analysis
  - Part of speech tagging
  - Parsing



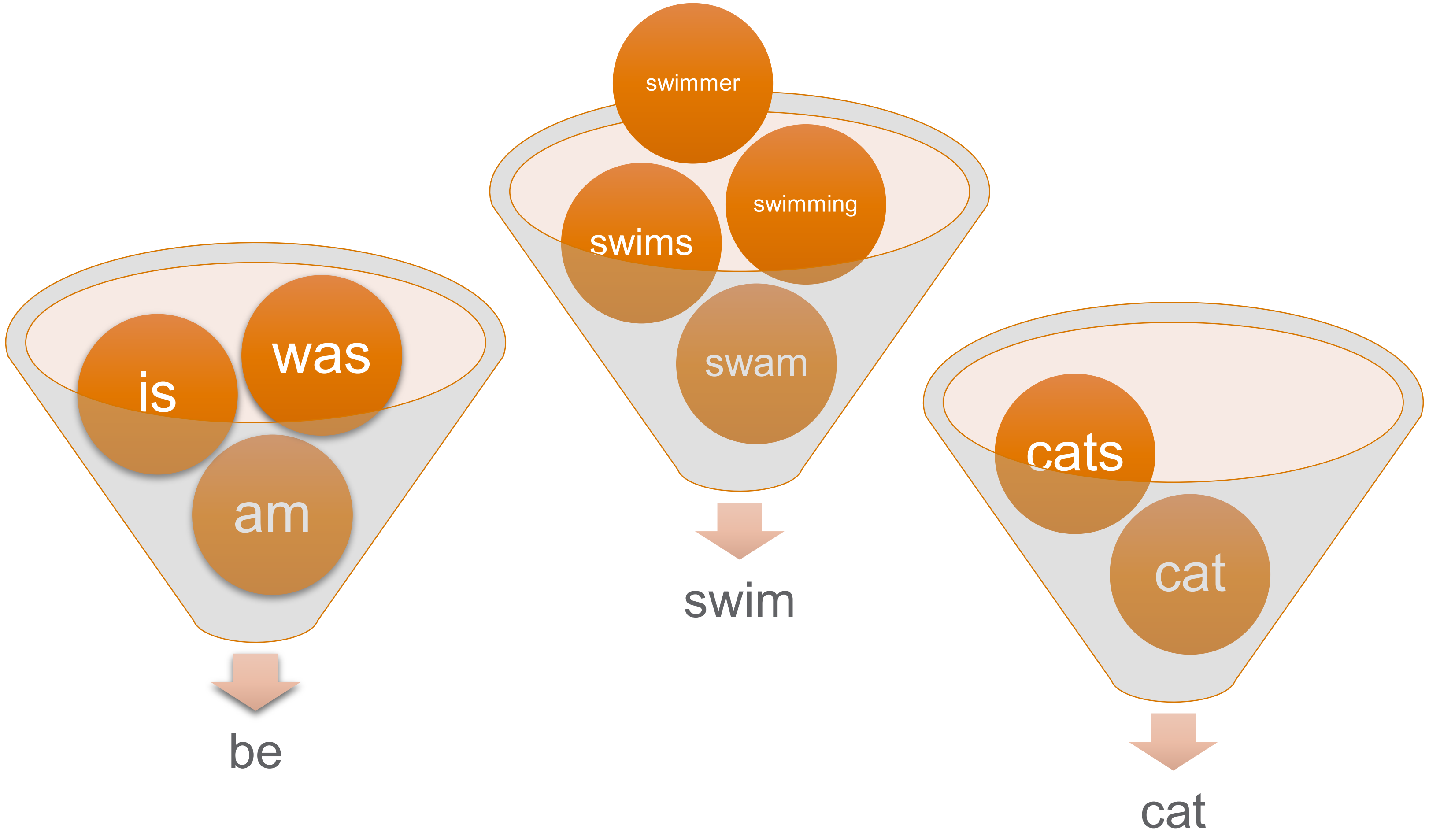
Stemming

- Reduce a word to a word stem (i.e., chopping off characters)

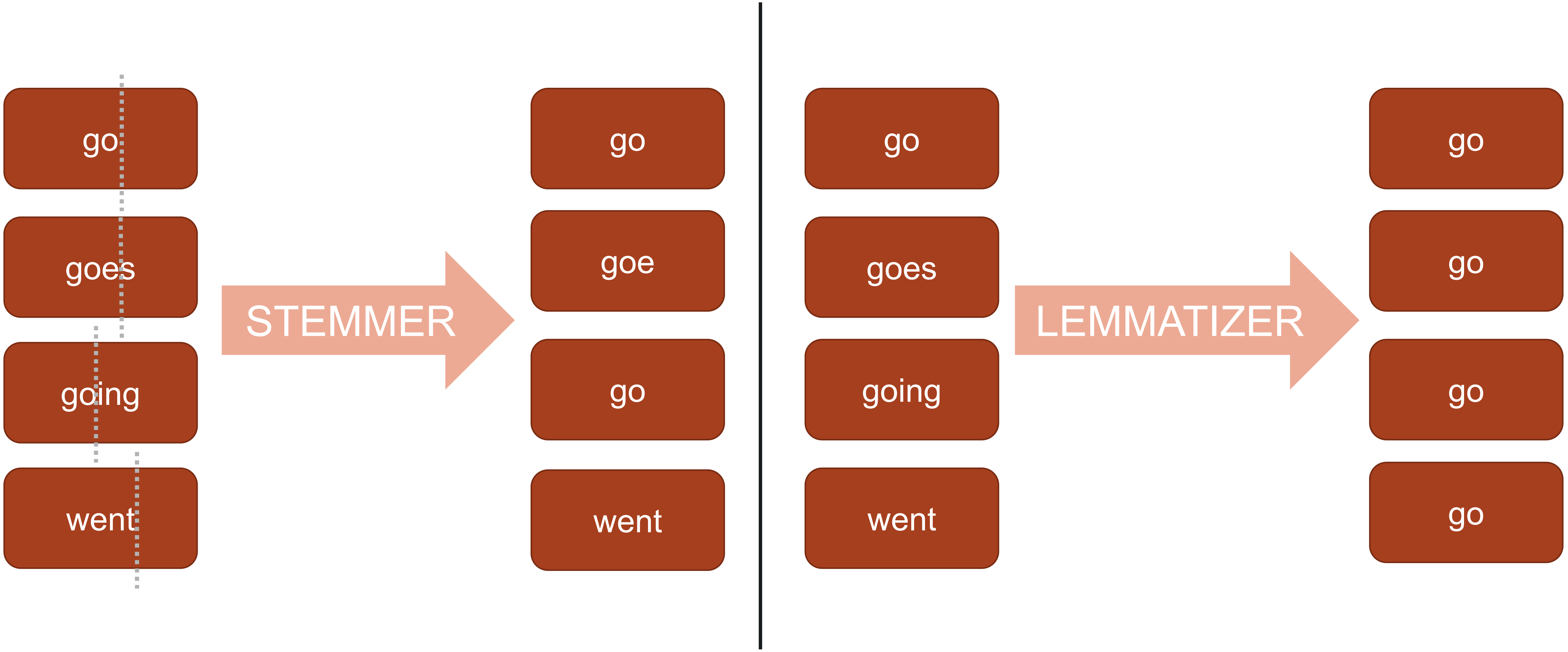


Lemmatization

- Reduce a word to a common base form (root words)
- Requires language-specific knowledge
- Usually more computationally expensive



Stemming and Lemmatization



Preprocessing Text: No "One-Size-Fits-All"

- The labs will give you practice with processing text!
  - Available tools can alleviate manual processing depending on the data

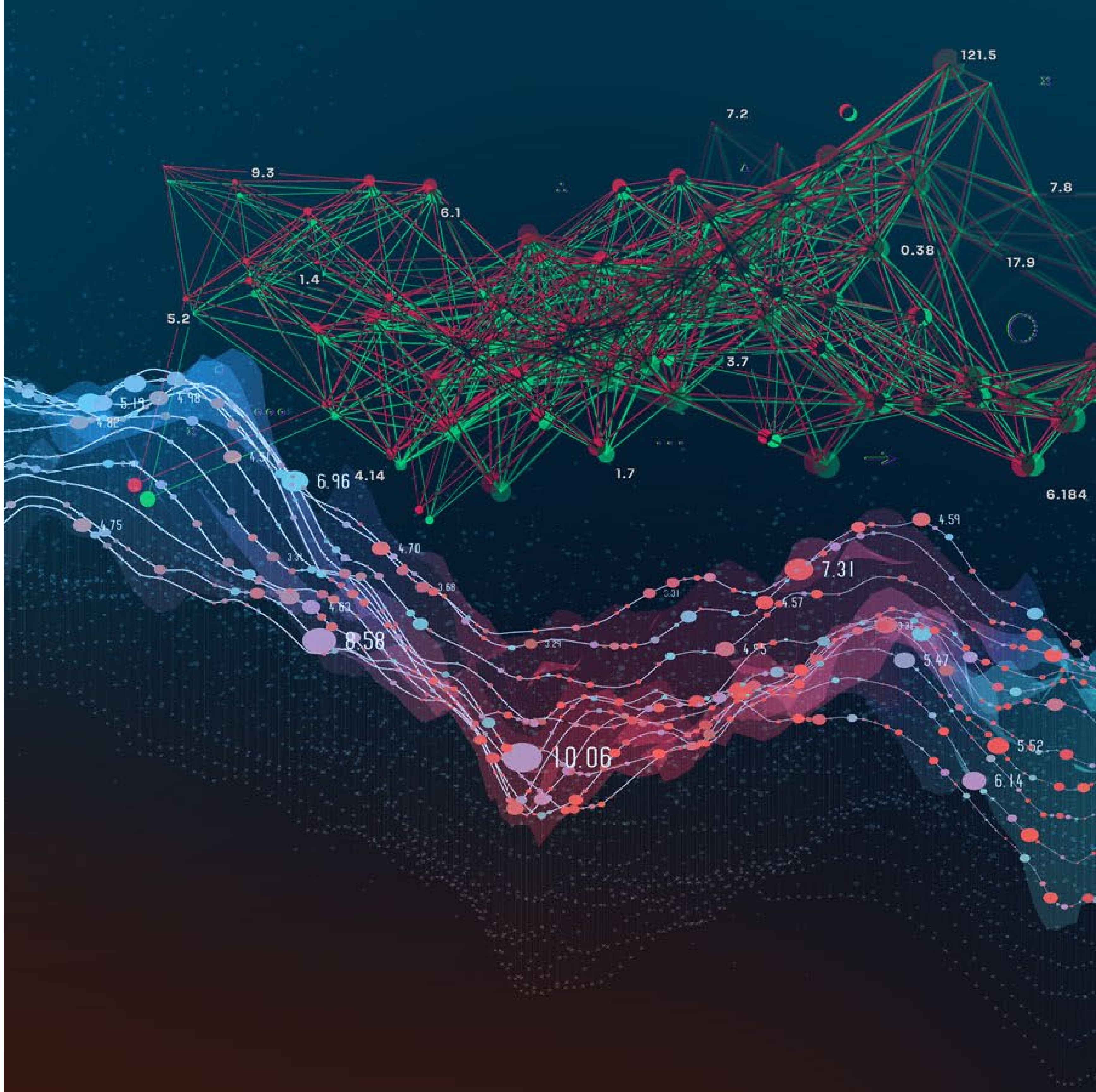
- Algorithms will typically incorporate a **subset** of these preprocessing methods

Questions? Comments?

Up Next: Available Tools

Pacific Northwest
NATIONAL LABORATORY

Available Tools for Common Tasks



U.S. DEPARTMENT OF ENERGY BATTELLE

PNNL is operated by Battelle for the U.S. Department of Energy

UNCLASSIFIED

Regular Expressions

- Fundamental ability in text preprocessing is searching for patterns within a string
  - Commonly known as **regex**

Uses:

- Subset the data
- Replace errors
- Extract useful information

Cheat sheets:

- https://www.cheatography.com/davechild/cheat-sheets/regular-expressions/

Regex Examples

- Example:
  - “^PRO.*[>;].*”

    (starts with *PRO*) (zero or more of any character except new line) (> or ;) (zero or more of any character except new line)

- Python string replace: `re.sub( pattern, replacement, text)`

- Example regex patterns:
  - Find anything not alphanumeric: “[^a-zA-Z0-9]”
  - Find any new lines: “\n”

- How would you remove an email address? `re.sub(“\S+@\S+\s?”, “”, text)`

Natural Language Toolkit (NLTK)

- Natural Language Toolkit

https://www.nltk.org/

- Popular in academia and research
  - Options

- Historically most used
  - spaCy has superseded NTLK for a lot of work

Tokenize and tag some text:

```
>>> import nltk
>>> sentence = """At eight o'clock on Thursday morning
... Arthur didn't feel very good."""
>>> tokens = nltk.word_tokenize(sentence)
>>> tokens
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning',
'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']
>>> tagged = nltk.pos_tag(tokens)
>>> tagged[0:6]
[('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'), ('on', 'IN'),
('Thursday', 'NNP'), ('morning', 'NN')]
```

Identify named entities:

```
>>> entities = nltk.chunk.ne_chunk(tagged)
>>> entities
Tree('S', [('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'),
           ('on', 'IN'), ('Thursday', 'NNP'), ('morning', 'NN'),
       Tree('PERSON', [('Arthur', 'NNP')]),
           ('did', 'VBD'), ("n't", 'RB'), ('feel', 'VB'),
           ('very', 'RB'), ('good', 'JJ'), ('.', '.')])
```

Natural Language Toolkit (NLTK)

- Great for exploring core NLP topics/principles
  - Available large corpora
  - Used in other libraries
  - Legacy means many well-established algorithms, tools, etc. are easily available through the NLTK interface
- Documentation is not well-maintained
- Speed cannot compare with industry-standard tools, like spaCy

spaCy

- spaCy
https://spacy.io/
- Popular in development
  - Fewer options, more streamlined ("opinionated")
  - Efficient
- Multiple language support for tokenization, parsing, named entity recognition, etc.
  - Check training data details – some languages have more resources → are more accurate

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Apple is looking at buying U.K. startup for $1 billion")
for token in doc:
    print(token.text, token.pos_, token.dep_)
```

RUN

```
Apple PROPN nsubj
is AUX aux
looking VERB ROOT
at ADP prep
buying VERB pcomp
U.K. PROPN dobj
startup NOUN advcl
for ADP prep
$ SYM quantmod
1 NUM compound
billion NUM pobj
```

spaCy

- Designed for developers
  - Performant and scalable
  - Easy to get started
  - Well-maintained and frequently updating
  - *Fewer options for choosing particular algorithms or making small changes*

Tends to present you with the "best" option

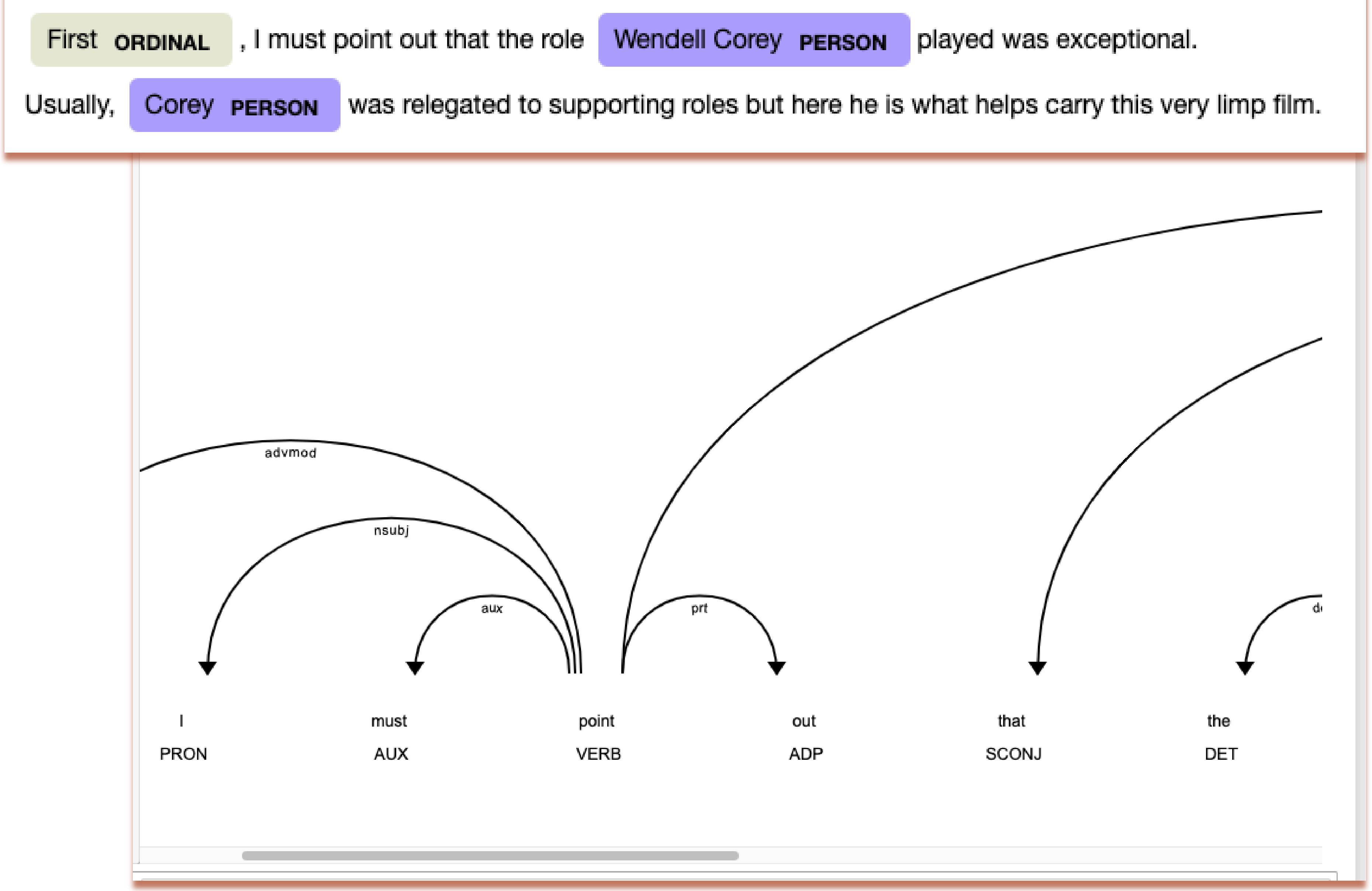
- Supports fewer human languages compared to some other tools
  - ***It does integrate well with third party libraries, which can ameliorate some gaps in language support

displaCy

- spaCy offers a visualization tool
  - NER
  - Dependency parsing

- Custom spacy components/factories allow for displaying output from non-spaCy models

```
# Load spaCy's core English module.
nlp = spacy.load('en_core_web_sm')
doc = nlp(text)
spacy.displacy.render(list(doc.sents)[:2], style='ent')
```



Stanford CoreNLP

- CoreNLP

  https://stanfordnlp.github.io/CoreNLP/

  https://stanfordnlp.github.io/stanza/ (Stanza – python interface)

- Multiple language support

3. Accessing Annotations

Annotations can be accessed from the returned `Document` object.

A `Document` contains a list of `Sentence` s, which contain a list of `Token` s. Here let's first explore the annotations store

```
# Iterate over all tokens in all sentences, and print out the word, lemma, pos and ner tags
print("{:12s}\t{:12s}\t{:6s}\t{}".format("Word", "Lemma", "POS", "NER"))

for i, sent in enumerate(document.sentence):
    print("[Sentence {}]".format(i+1))
    for t in sent.token:
        print("{:12s}\t{:12s}\t{:6s}\t{}".format(t.word, t.lemma, t.pos, t.ner))
    print("")
```

```
Word            Lemma           POS     NER
[Sentence 1]
Albert          Albert          NNP     PERSON
Einstein        Einstein        NNP     PERSON
was             be              VBD     O
a               a               DT      O
German          german          JJ      NATIONALITY
-               -               HYPH    O
born            bear            VBN     O
theoretical     theoretical     JJ      TITLE
physicist       physicist       NN      TITLE
.               .               .       O
```

Stanford CoreNLP

- Provides high/accurate performance
  - Though used in both academia and industry, spaCy is often preferred due to speed
  - Stanford usually outscores spaCy on standard evaluation measures
- Written in Java, but Python wrappers (Stanza) make it more accessible
- Reliable, accurate models available in other languages
  - Before more non-English resources were readily available, CoreNLP was often the go-to for languages such as Arabic and Chinese

HuggingFace

- Go-to library for Transformer models
- Built on pyTorch
- Applicable to all skill levels
  - End-to-end pipelines
  - Ability to "get down in the weeds”
- Hosts almost all available models and training data sets
  - Vast model zoo
  - Huge variety of data corpora

HuggingFace – Tokenization (whitespace)

```
In [1]: sentence = "I'm a MéSsY sentence!! http://www.pnnl.gov 🐐😅"

In [2]: from tokenizers.pre_tokenizers import Whitespace

        whitespace_tokenizer = Whitespace()
        whitespace_tokenizer.pre_tokenize_str(sentence)

Out[2]: [('I', (0, 1)),
         ("'", (1, 2)),
         ('m', (2, 3)),
         ('a', (4, 5)),
         ('MéSsY', (6, 11)),
         ('sentence', (12, 20)),
         ('!!', (20, 22)),
         ('http', (23, 27)),
         ('://', (27, 30)),
         ('www', (30, 33)),
         ('.', (33, 34)),
         ('pnnl', (34, 38)),
         ('.', (38, 39)),
         ('gov', (39, 42)),
         ('🐐😅', (43, 45))]
```

HuggingFace – Tokenization (BERT)

```
In [1]: sentence = "I'm a MéSsY sentence!! http://www.pnnl.gov 🐕😅"
```

```
In [3]: from transformers import BertTokenizer

        tokenizer = BertTokenizer.from_pretrained('bert-base-cased')
        tokenizer.tokenize(sentence)
```

```
Out[3]: ['I',
         "'",
         'm',
         'a',
         'M',
         '##é',
         '##S',
         '##s',
         '##Y',
         'sentence',
         '!',
         '!',
         'http',
         ':',
         '/',
         '/',
         'www',
         '.',
         'p',
         '##nn',
         '##l',
         '.',
         'go',
         '##v',
         '[UNK]']
```

HuggingFace – Tokenization (normalized)

```
In [1]: sentence = "I'm a MéSsY sentence!! http://www.pnnl.gov 🐐😅"
```

```
In [4]: from tokenizers import normalizers
        from tokenizers.normalizers import StripAccents, Lowercase, Replace, NFD

        normalizer = normalizers.Sequence([
            NFD(),
            Lowercase(),
            StripAccents(),
            Replace('http://www.pnnl.gov', ''),
            Replace('🐐', ''),
            Replace('😅', ''),
            ])

        normed = normalizer.normalize_str(sentence)
        tokenizer.tokenize(normed)

Out[4]: ['i', "'", 'm', 'a', 'messy', 'sentence', '!', '!']
```

HuggingFace – Fine Tuning

- Transformer-based models can be prohibitive to train from scratch
  - Significant data and compute resources required

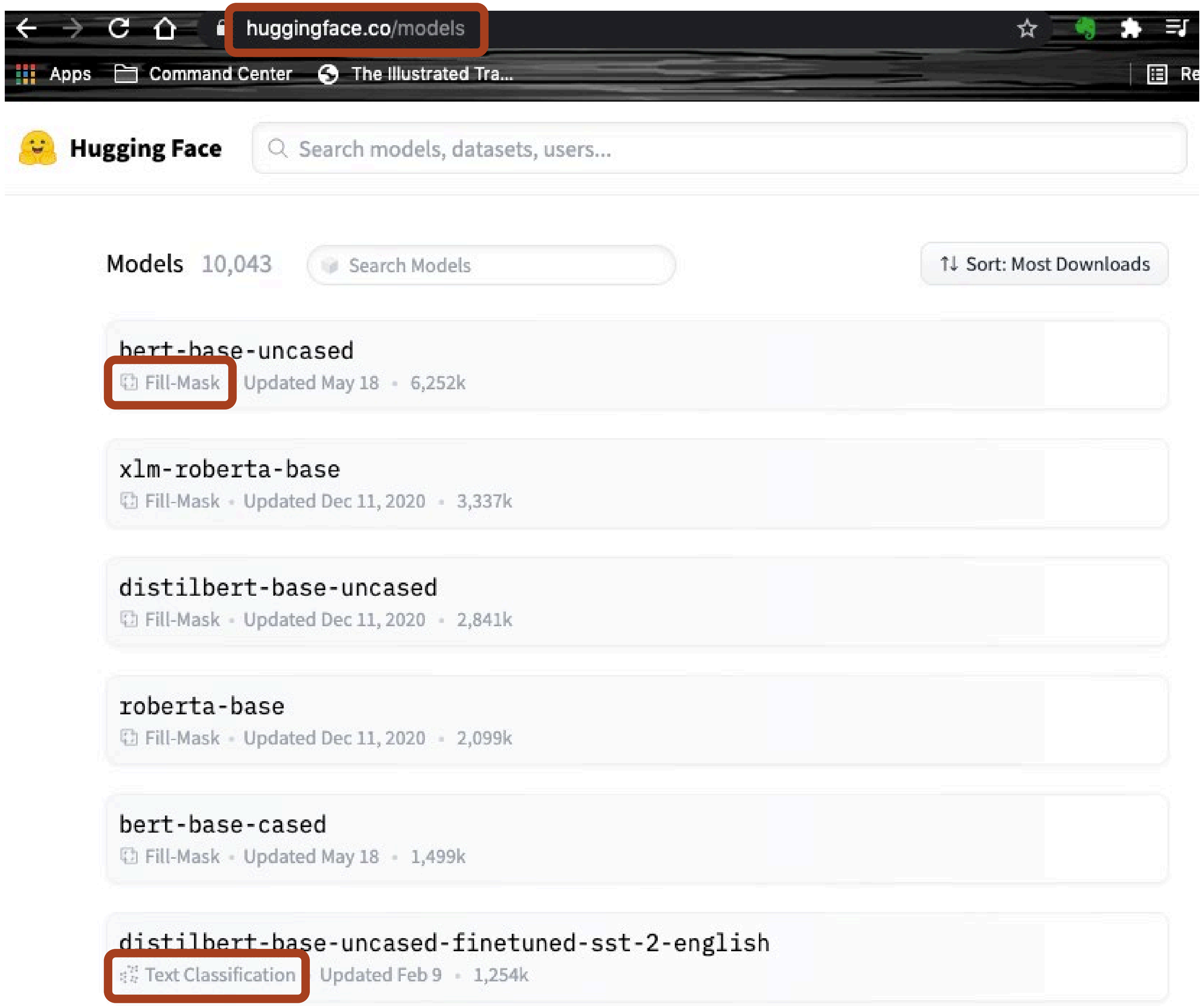
- Standard practice is to take an existing model and fine-tune for your task
  - Easy pipeline for simple tasks
  - Or you can use your own PyTorch/Tensorflow/Keras training code

```
from transformers import Trainer

trainer = Trainer(
    model=model, args=training_args, train_dataset=small_train_dataset, eval_dataset=small_eval_dataset
)
```

```
trainer.train()
```

HuggingFace – Choosing a (pretrained) Model



Model Cards (*usually*) include:

- Summary
- Interactive Sandbox
- Model Description
- Intended Uses
- How to use (with code)
- Limitations and Biases
- Training Data and Procedure
- Citation Information

Do: Make sure to apply due diligence in researching how a model was trained and what biases may be present.

Model Cards for Model Reporting: https://arxiv.org/abs/1810.03993

Pacific Northwest NATIONAL LABORATORY

HuggingFace – Fill Mask

In [7]:
```python
from transformers import pipeline

unmasker = pipeline('fill-mask', model='xlm-roberta-base')
```

In [8]:
```python
unmasker("I like to <mask> in the park.")
```

Out[8]:
```
[{'sequence': 'I like to walk in the park.',
  'score': 0.18402010202407837,
  'token': 35691,
  'token_str': 'walk'},
 {'sequence': 'I like to play in the park.',
  'score': 0.15702171623706818,
  'token': 11301,
  'token_str': 'play'},
 {'sequence': 'I like to be in the park.',
  'score': 0.11749979108572006,
  'token': 186,
  'token_str': 'be'},
 {'sequence': 'I like to stay in the park.',
  'score': 0.09088794887065887,
  'token': 24765,
  'token_str': 'stay'},
 {'sequence': 'I like to live in the park.',
  'score': 0.05844708904623985,
  'token': 6867,
  'token_str': 'live'}]
```

HuggingFace – Text Generation

```
In [9]: from transformers import pipeline

        generator = pipeline('text-generation', model='EleutherAI/gpt-neo-2.7B')
```

```
In [10]: out = generator('I wish', do_sample=True, min_length=50)
         print(out[0]['generated_text'])
```

```
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
```

```
I wish to thank you for your patience with me.

After a break from this blog I wanted to say that I have some good news.
We have signed a lease on a apartment! Our apartment will be with the com
munity of South Side in
```

R Tools and References

Text Mining with R	• **Free online book (tidyverse)** • **https://www.tidytextmining.com/**
Bayesian Analysis in Natural Language Processing	• Kindle, soft/hardcover • Useful for **Bayesian NLP research**
Deep Learning for NLP and Speech Recognition	• Kindle, soft/hardcover • Thorough overview of **DL in NLP**
Other online resources	• https://paulvanderlaken.com/2017/08/10/r-resources-cheatsheets-tutorials-books/#textmining

Python to R Conversion

Python	R
re	grep, stringr
nltk	tm, openNLP, textstem
spacy	spacyr
wordcloud	wordcloud, wordcloud2
gensim	text2vec
sklearn	tidytext, stats, randomForest, glmnet, …
transformers	torch, keras, fastai, reticulate, text, transformers

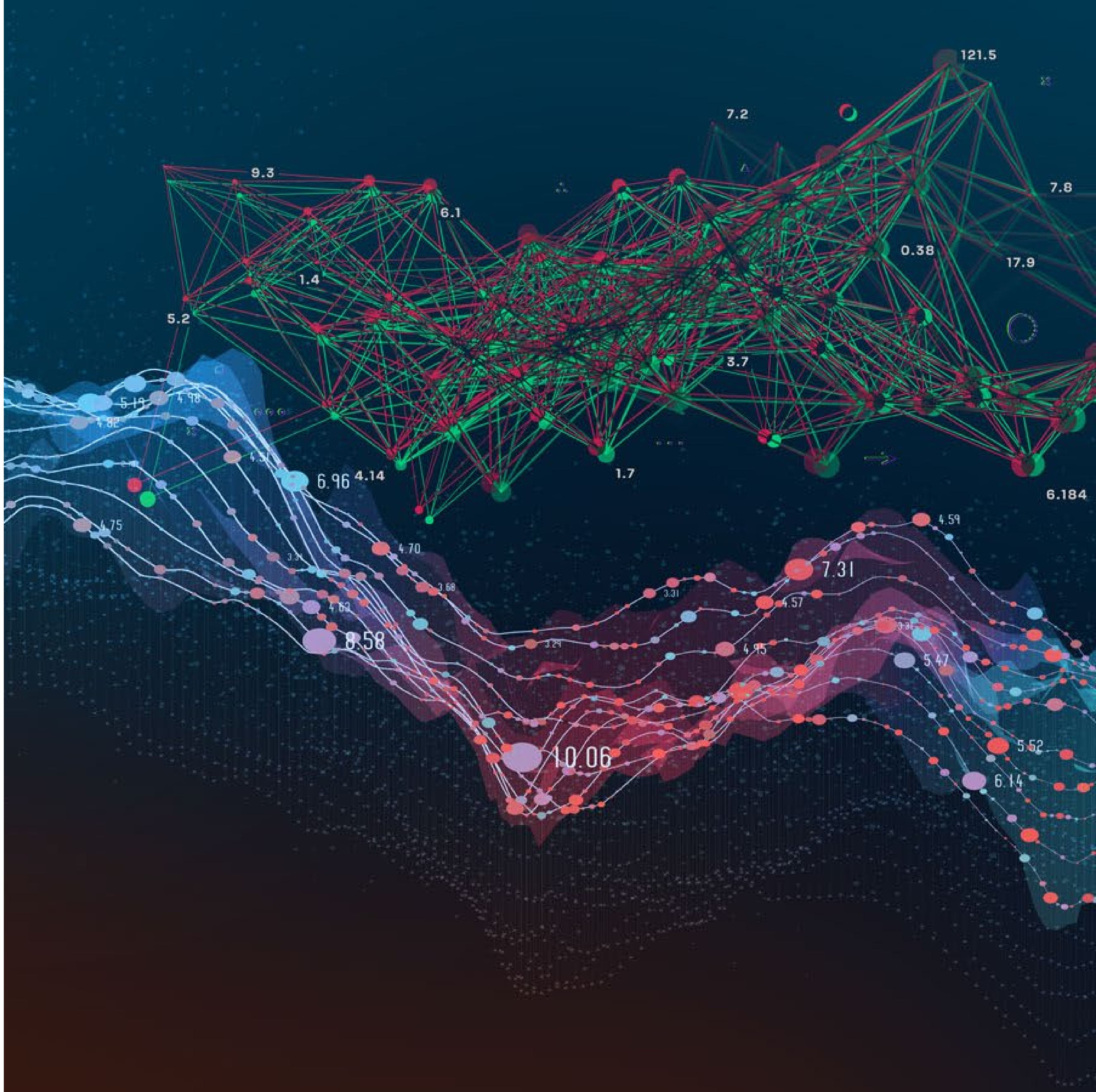
Python Tools and Main Takeaways

NLTK	• Well-established and offers a variety of algorithms • Steeper learning curve
spaCy	• Designed for developers/industry
CoreNLP	• Accurate foundational models available • Speed cannot compare to spaCy
HuggingFace	• Mainstay for transformers/BERT-style models

Questions? Comments?

Up Next: Text Representations

Text Representations



U.S. DEPARTMENT OF ENERGY BATTELLE

PNNL is operated by Battelle for the U.S. Department of Energy

Where we are on the roadmap

- Only after we’ve preprocessed the text can we convert it into **vector formats**
- This can encompass the text itself, linguistic signals or other data features
- We can analyze these features on their own or supply to an ML algorithm

Overview of Approaches

Language Expressed as Words

Language Expressed as Numbers

Bag of Words | **N-Grams** | **TF-IDF** | **Embeddings**

Relative Control Over Training Data

Most control over data

Least control over data

- Methods of representing text provide different underlying features
- Useful on their own for **analyzing the textual content** of a given corpus
- Can also be supplied as **input vectors** to learned models (e.g., Recurrent Neural Networks, Transformers)

Bag of Words

- Describes the occurrence of each word within a document or corpus
- No regard for grammar or order, only **multiplicity**

1. Each unique word is catalogued in a dictionary
   - Each word (or token) is called a "gram"
2. Frequency of each word is calculated from given text

Review 1: This movie is very scary and long.
Review 2: This movie is not scary and is slow.
Review 3: This movie is spooky and good.

	Review 1	Review 2	Review 3
and	1	1	1
good	0	0	1
is	1	2	1
long	1	0	0
movie	1	1	1
not	0	1	0
scary	1	1	0
slow	0	1	0
spooky	0	0	1
this	1	1	1
very	1	0	0

Term Frequency – Inverse Document Frequency

- TF-IDF: Counts the **frequency of each unique word** (or n-gram) within a document and scales by the ***document length*** and ***corpus frequency***

- More frequent words (relative to document and corpus) are more descriptive of the document
  - Common words (e.g., “the”, “and”, “a”) are given less weight

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Review 1: This movie is very scary and long. Length 7
Review 2: This movie is not scary and is slow. Length 8
Review 3: This movie is spooky and good. Length 6

Bag of Words

	R1	R2	R3
and	1	1	1
good	0	0	1
is	1	2	1
long	1	0	0
movie	1	1	1
not	0	1	0
scary	1	1	0
slow	0	1	0
spooky	0	0	1
this	1	1	1
very	1	0	0

Review 1: This movie is very scary and long. ← Length 7
Review 2: This movie is not scary and is slow. ← Length 8
Review 3: This movie is spooky and good. ← Length 6

$tf_{i,j}$ = number of occurrences of i in j

Bag of Words

	R1	R2	R3
and	1	1	1
good	0	0	1
is	1	2	1
long	1	0	0
movie	1	1	1
not	0	1	0
scary	1	1	0
slow	0	1	0
spooky	0	0	1
this	1	1	1
very	1	0	0

TF

R1	R2	R3
1/7	1/8	1/6
0	0	1/6
1/7	2/8	1/6
1/7	0	0
1/7	1/8	1/6
0	1/8	0
1/7	1/8	0
0	1/8	0
0	0	1/6
1/7	1/8	1/6
1/7	0	0

Review 1: This movie is very scary and long.
Review 2: This movie is not scary and is slow.
Review 3: This movie is spooky and good.

$$\log\left(\frac{N}{df_i}\right)$$

df_i = number of documents containing i
N = total number of documents

	Bag of Words			TF			IDF
	R1	R2	R3	R1	R2	R3	
and	1	1	1	1/7	1/8	1/6	0
good	0	0	1	0	0	1/6	0.48
is	1	2	1	1/7	2/8	1/6	0
long	1	0	0	1/7	0	0	0.48
movie	1	1	1	1/7	1/8	1/6	0
not	0	1	0	0	1/8	0	0.48
scary	1	1	0	1/7	1/8	0	0.18
slow	0	1	0	0	1/8	0	0.48
spooky	0	0	1	0	0	1/6	0.48
this	1	1	1	1/7	1/8	1/6	0
very	1	0	0	1/7	0	0	0.48

Review 1: This movie is very scary and long.
Review 2: This movie is not scary and is slow.
Review 3: This movie is spooky and good.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

Bag of Words				TF			IDF	TF-IDF		
	R1	R2	R3	R1	R2	R3		R1	R2	R3
and	1	1	1	1/7	1/8	1/6	0	0	0	0
good	0	0	1	0	0	1/6	0.48	0	0	**0.080**
is	1	2	1	1/7	2/8	1/6	0	0	0	0
long	1	0	0	1/7	0	0	0.48	**0.068**	0	0
movie	1	1	1	1/7	1/8	1/6	0	0	0	0
not	0	1	0	0	1/8	0	0.48	0	**0.060**	0
scary	1	1	0	1/7	1/8	0	0.18	**0.026**	**0.022**	0
slow	0	1	0	0	1/8	0	0.48	0	**0.060**	0
spooky	0	0	1	0	0	1/6	0.48	0	0	**0.080**
this	1	1	1	1/7	1/8	1/6	0	0	0	0
very	1	0	0	1/7	0	0	0.48	**0.068**	0	0

Pacific Northwest NATIONAL LABORATORY

Review 1: This movie is very scary and long.
Review 2: This movie is not scary and is slow.
Review 3: This movie is spooky and good

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

TF-IDF

	R1	R2	R3
and	0	0	0
good	0	0	**0.080**
is	0	0	0
long	**0.068**	0	0
movie	0	0	0
not	0	**0.060**	0
scary	**0.026**	**0.022**	0
slow	0	**0.060**	0
spooky	0	0	**0.080**
this	0	0	0
very	**0.068**	0	0

N-Gram Analysis

- An n-gram is a contiguous sequence of *n* items from a given sample of text
- Allows us to better understand relationships between words in a sequence

“I am a whale”

Unigrams (n=1)	Bigrams (n=2)	Trigrams (n=3)
“I” “am” “a” “whale”	“I am” “am a” “a whale”	“I am a” “am a whale”

Pacific Northwest NATIONAL LABORATORY

N-Gram Analysis

- Statistical properties of n-grams are used to model sequences
  - e.g., Language Modeling
  - Simple and scalable: larger n → more context

- Straightforward to calculate, useful for rough analysis of data

“I am a whale”

“I”
“am”
“a”
“whale”

Unigrams (n=1)

“I am”
“am a”
“a whale”

Bigrams (n=2)

“I am a”
“am a whale”

Trigrams (n=3)

Curse of Dimensionality

- Data Sparsity is an issue with text
  - Large vocabulary; high dimensionality
- Preprocessing can help reduce dimensionality

Embeddings

- Previous representations ignore nuance in how language is used
  - “You shall know a word by the company it keeps” John Firth (1957)

- Embeddings learn a representation of a word/character/subword based on where that word occurs

- Benefits?
  - Good representation for modeling
  - Interpretability
  - Similarity measures

Pacific Northwest NATIONAL LABORATORY

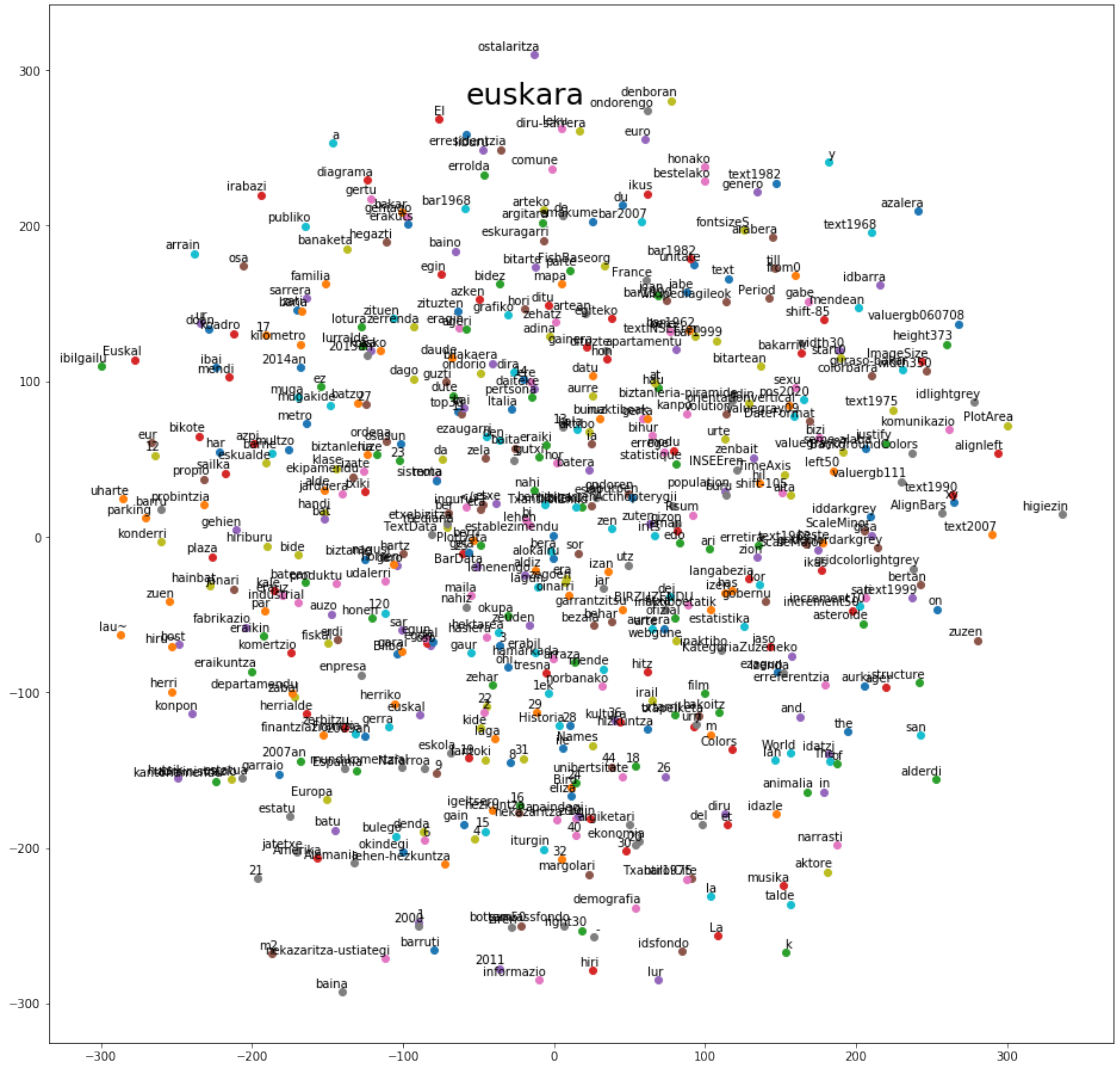
Word Embeddings

- Bag of words and TF-IDF do not capture the similarity between different words

Dog =	0	0	0	…	1	…	0
Dogs =	0	1	0	…	0	…	0

- Embeddings can capture similarity between related inputs
  - "Similar" words are near one another
  - Each word is represented by a vector of continuous numbers (e.g., Word2Vec)

Dog =	0.1	0.3	1.2	…	-1.	…	2.3
Dogs =	0.2	0.2	1.1	…	-1.	…	2.2



OG Word Embeddings: word2vec and GloVe

- Published in 2013 (word2vec) and 2014 (GloVe)

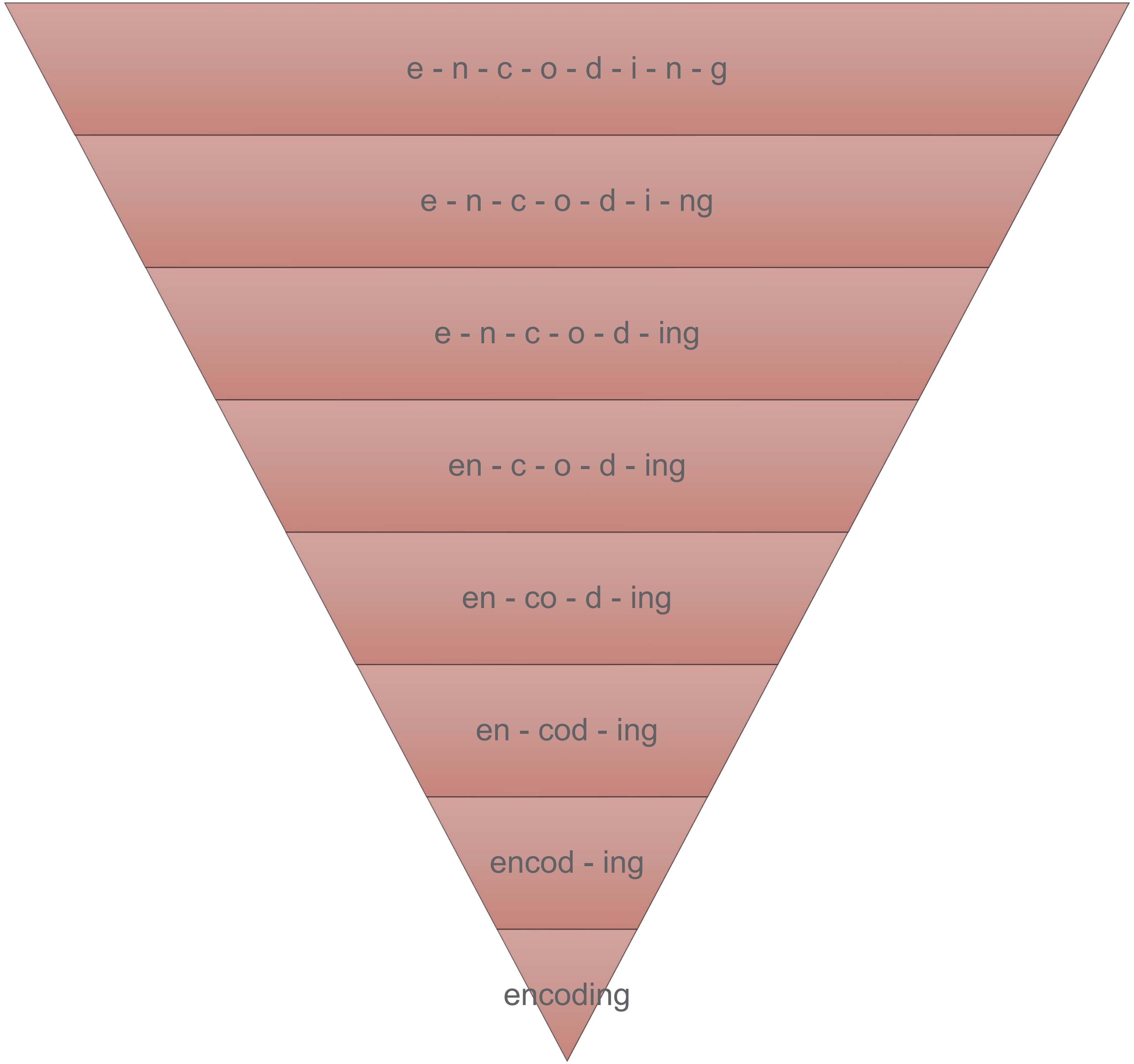
- Generate embeddings based on their local context
  - Word co-occurence
  - Continuous bag of words (word order does not contribute to embedding)
  - Skip-grams (Weighs nearby words more heavily than more distant words)

- e.g., "dog" and "dogs" are treated as distinct words and must be learned separately

- All words not learned in training are given the same value: Out-of-vocabulary (OOV)

Building from the ground up: Byte-Pair Encodings (BPEs)

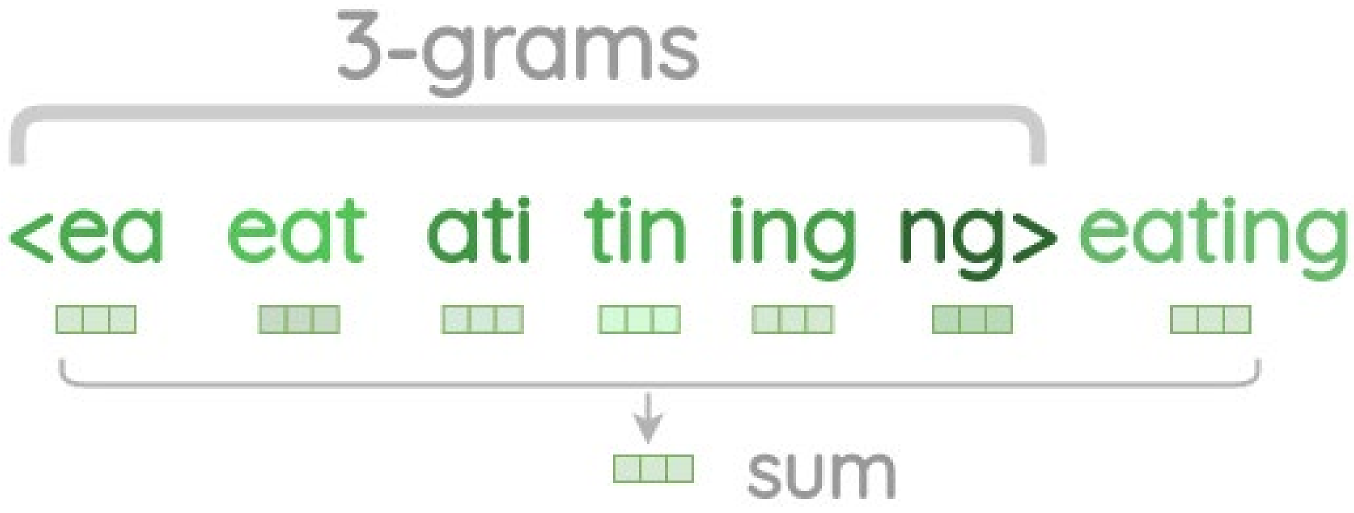
- “Middle ground” between character- and word-level encodings
- Vocabulary-agnostic
- Probability-based
- SotA for context-agnostic embeddings



Sennrich, Rico, Barry Haddow, and Alexandra Birch. "Neural machine translation of rare words with subword units." *arXiv preprint arXiv:1508.07909* (2015).

fastText: Raising the Bar

- Leverages *sub-word* embeddings
  - Character n-grams get embeddings
  - Word embeddings are sums of their character n-grams
  - Trained using word2vec-style skipgram
- Learns word piece meaning e.g., Latin roots
- e.g., "dog" and "dogs" share learned embedding
- Can represent out-of-vocab words based on their spelling!

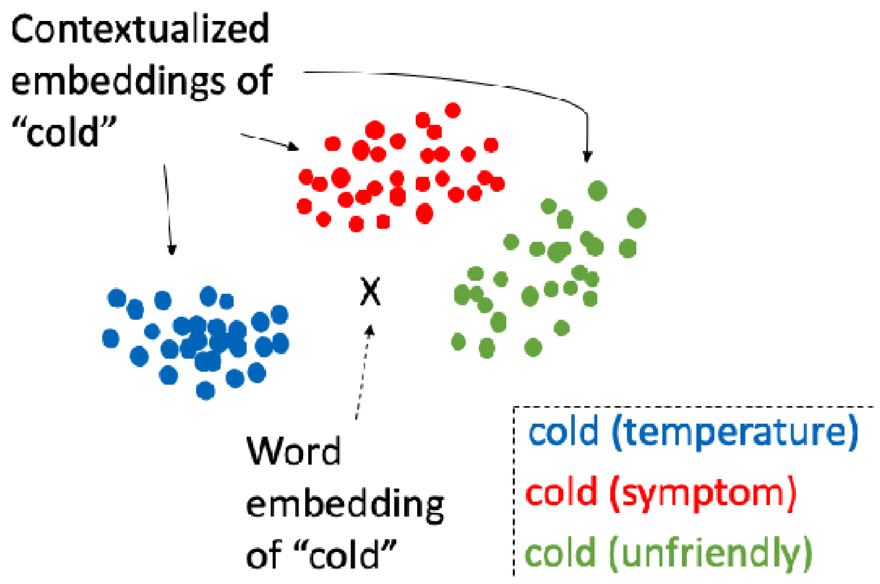


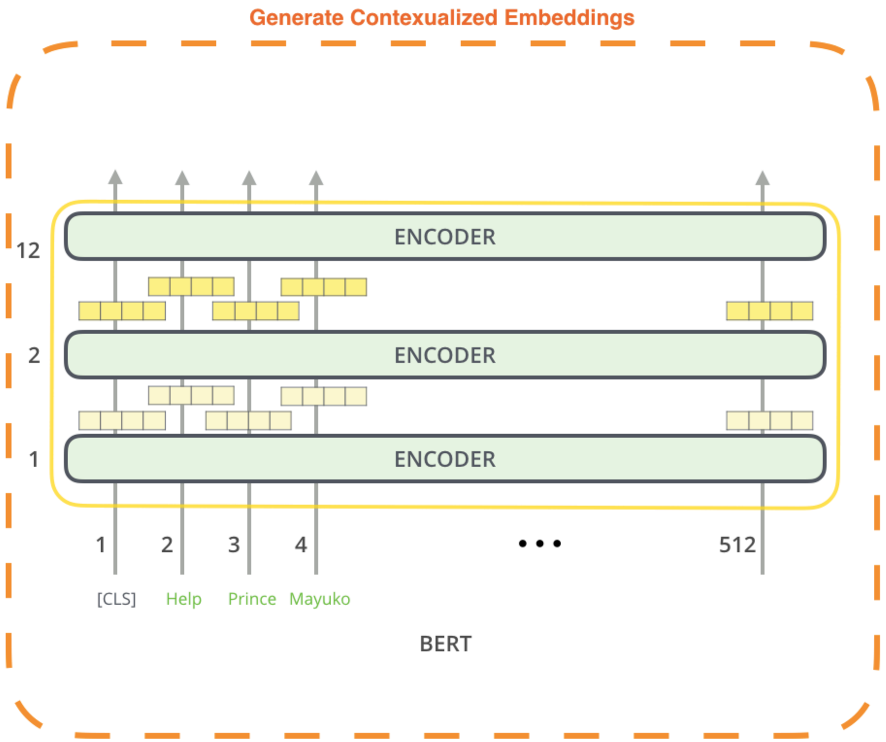
https://fasttext.cc/

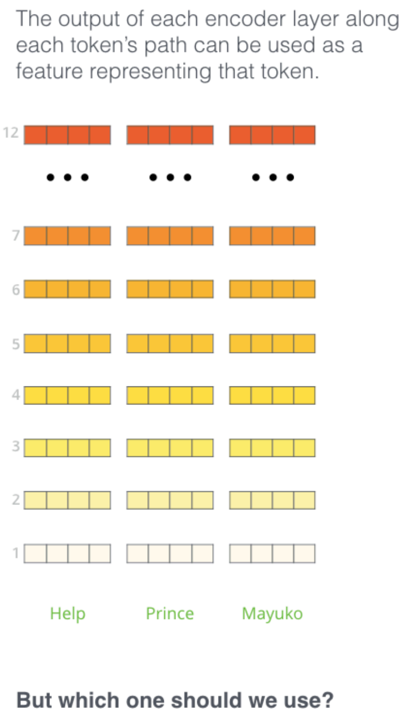
Contextual Embeddings

- **Problem:** single embedding vectors fail to account for multiple word senses and nuance in usage
- **Solution:** train embeddings where the learned representation depends on the context surrounding the word

- Done by **leveraging later layers** of the embedding model where the representations are updated based on information from the rest of the input sequence (context)
  - ELMo (Peters et al., 2018) https://allennlp.org/elmo







Pacific Northwest NATIONAL LABORATORY

ELMo (Embeddings from Language Model)

- Traditional word embeddings have same vector for every instance of a word
  - E.g. *bat* has same vector for animal and sport

- ELMo uses context to differentiate
  - Bi-directional LSTM model creates embeddings

- Character-based
  - Not limited to a pre-defined vocabulary
  - Doesn't perform as well as word-based models

BERT (Bidirectional Encoder Representations from Transformers) and Transformers

- Leverages Transformer / attention architecture
  - Context-sensitive, like ELMo
  - The same word will have different embeddings depending on context
  - Can train in parallel (FAST)

- Uses sub-word embeddings
  - More accurate than character-based methods
  - Learns meaning of word roots
  - Can successfully interpret OOV words

Text Representations

- Rough, statistical measures
  - Bag-of-words
  - N-grams
  - TF-IDF

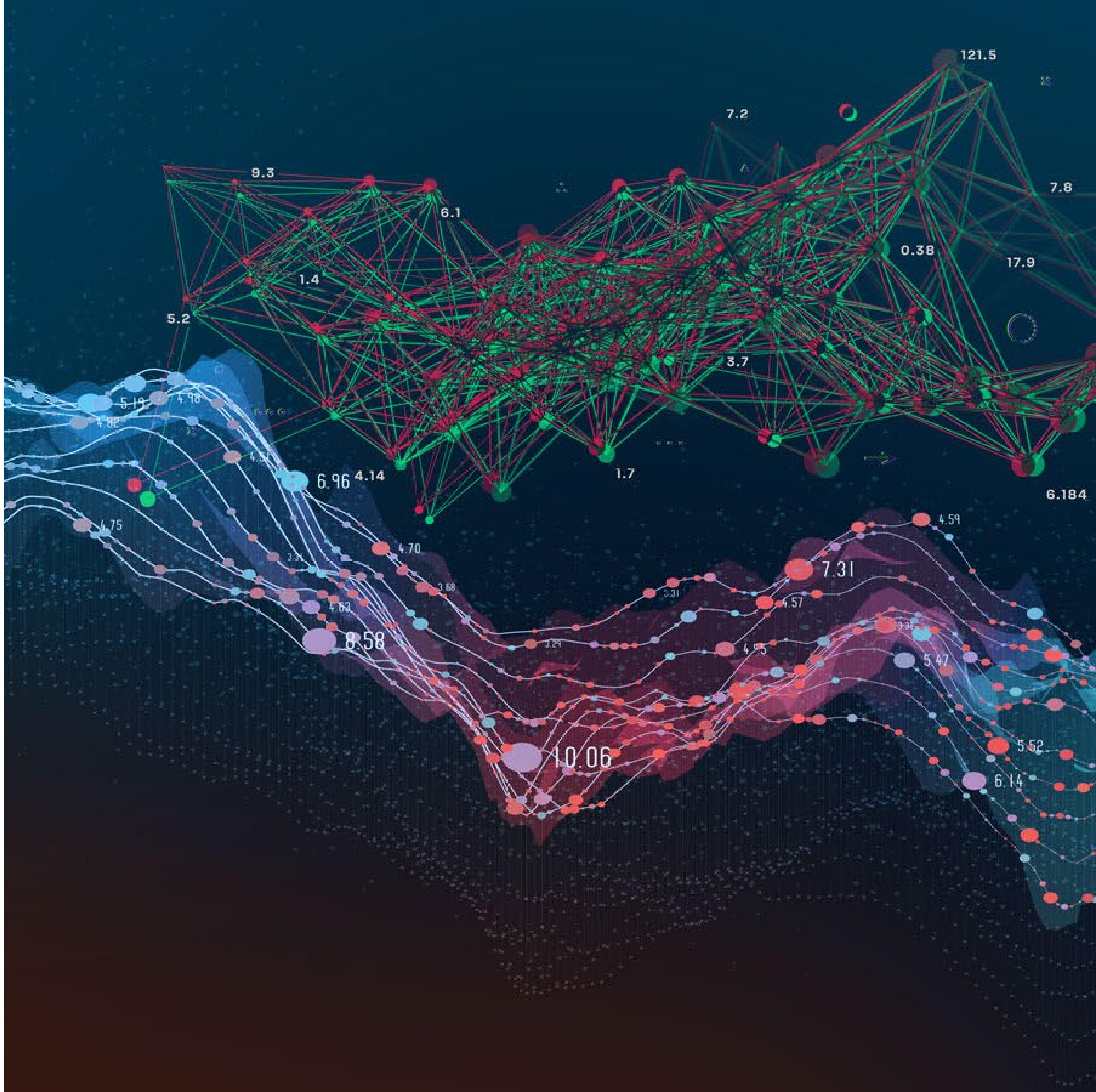
- Neural approaches
  - Word Embeddings
    - ✓ GLoVe, word2vec
  - Subword embeddings
    - ✓ fastText
    - ✓ ELMo
    - ✓ BERT/Transformers

Questions? Comments?

Up Next: Modeling & Transformers

Pacific Northwest
NATIONAL LABORATORY

Modeling



U.S. DEPARTMENT OF ENERGY BATTELLE

PNNL is operated by Battelle for the U.S. Department of Energy

UNCLASSIFIED

Where we are on the roadmap

- We can now provide the model-readable text to our chosen ML algorithm
- Start simple, move to more complex architectures as needed

Preprocessing Considerations

- Your choice of model can determine a lot about how you preprocess your data

- Generally, statistical models require significant preprocessing
- Deep learning models require very little preprocessing

Statistical model (e.g. SVM)

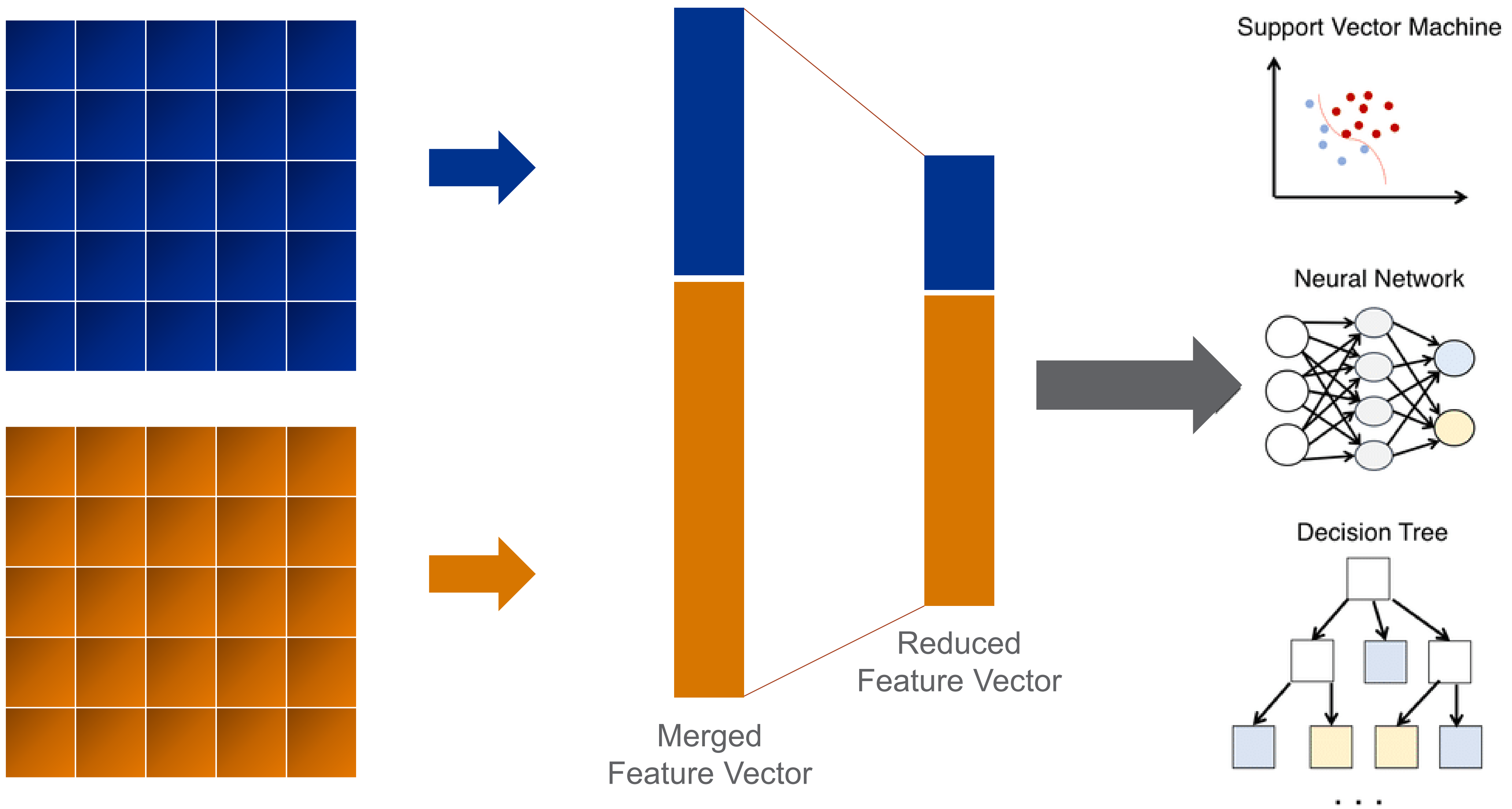
```
1. Data cleaning
2. Normalize case, remove punctuation
3. Run word/sentence tokenization
4. Remove stopwords
5. Run stemming
6. Calculate N-gram/TF-IDF features
7. Run model!
```

Neural model (e.g. RNN / Transformer)

```
1. Data cleaning
2. Possibly remove case?
3. Run word/sentence tokenization
4. …
5. …
6. …
7. Run model! (embeddings are included)
```

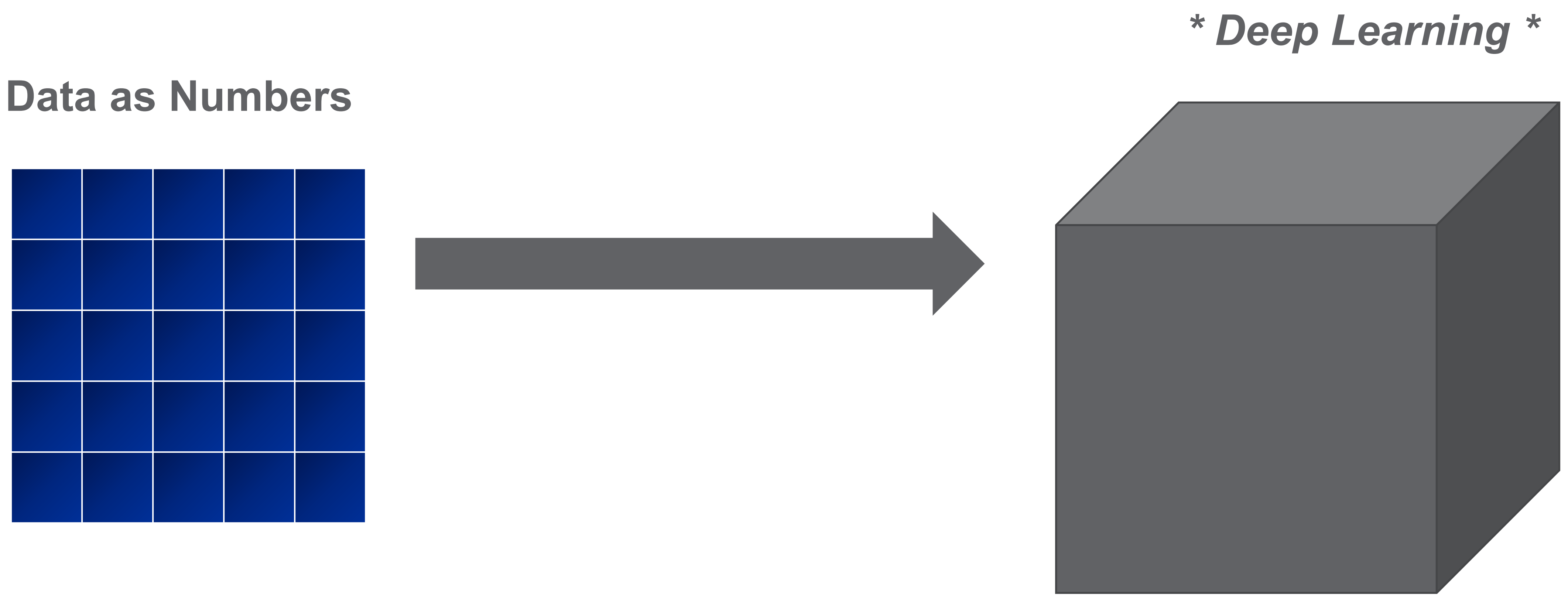
Machine Learning may require Feature Engineering

Hand-crafted Feature Extraction | Feature Selection | Statistical Modeling



https://arxiv.org/abs/2102.03336

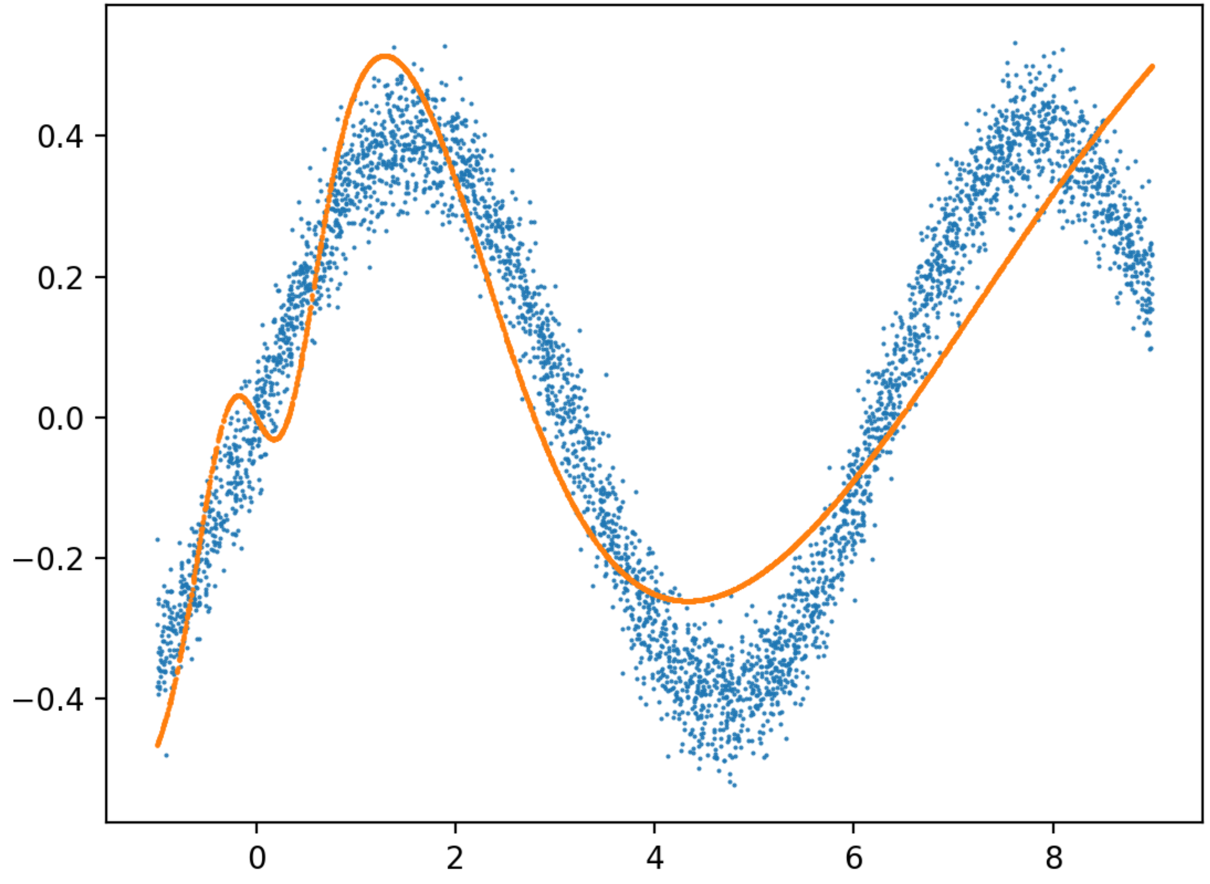
Deep Learning can succeed with raw data (“end-to-end” training)



APPENDIX

Neural Networks Are Function Approximators

- The real world is complex, and it is impossible to capture every detail
- Neural network modeling is about function approximation
  - Model observed behavior as a function
  - Map inputs to expected outputs
- **Contrary to popular belief, neural networks are not brains!**



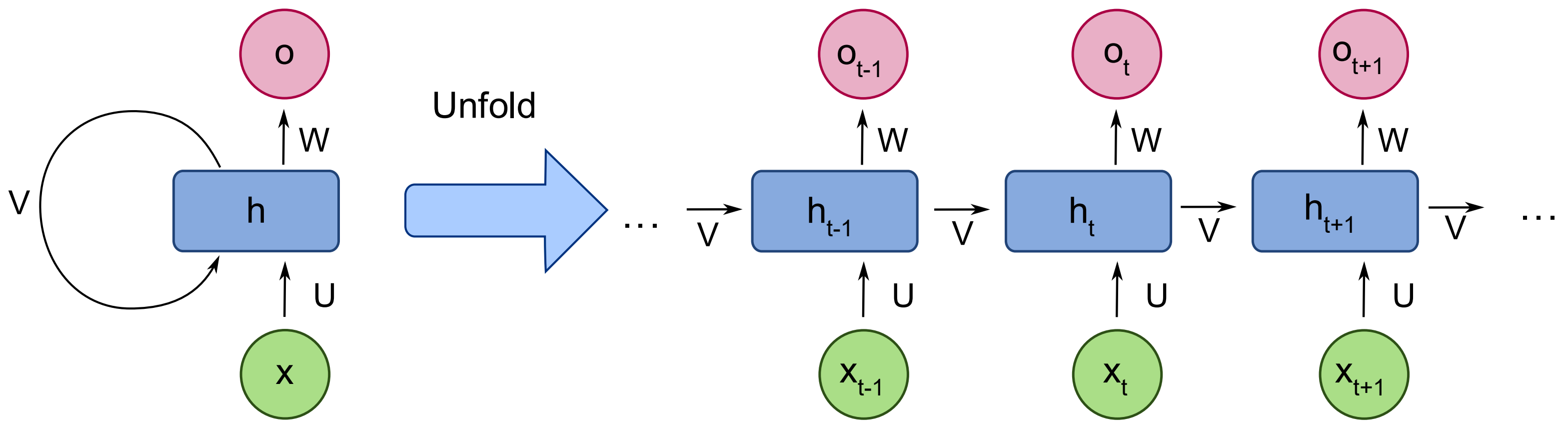
General Text Model Choices

- **Recurrent Neural Networks (RNNs)**
  - Traditional architecture
  - Disfavored today
- **Transformers**
  - State-of-the-art neural approach
- **Just about any other ML / DL model**
  - Use the representations from before and pop on your favorite clustering/classification algorithm

If you've run NLP models in the past, what did you use? Or any favorites?

Recurrent Neural Networks (RNNs)

- Feed-forward network with a **hidden state** that propagates information across a sequence – outputs are influenced by other elements in the **sequence**
- Training requires using same layer over and over again
  - Information propagates through *hidden* vector, h



Technical Difficulties…

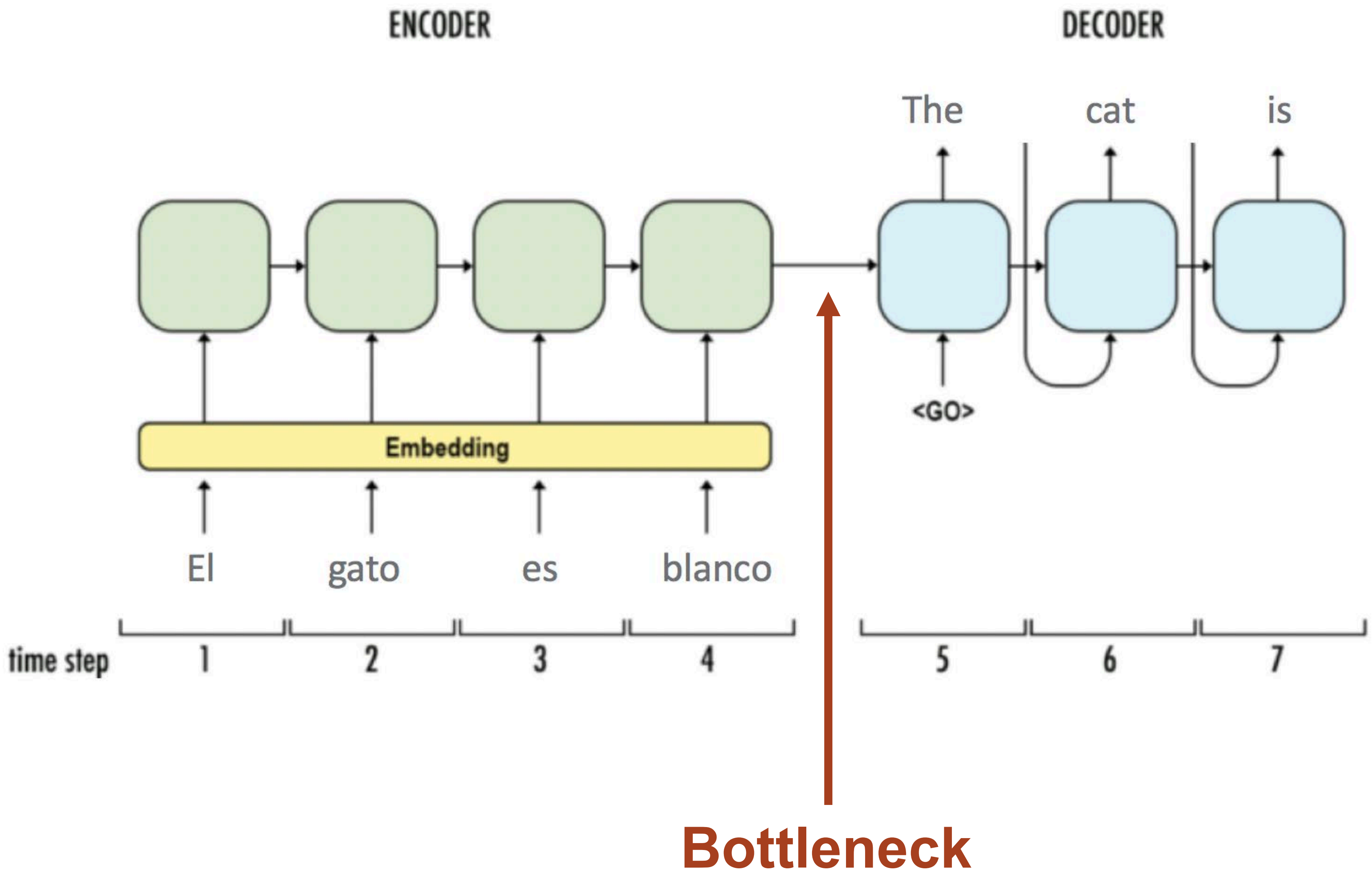
- **Exploding / Vanishing Gradients** (Bengio et al., 1994)
  - Norm of gradient approaches infinity/zero
  - Exponential over timesteps

- **Difficulty w/ Long-Term Dependencies**
  - All information is encoded in a single hidden state
  - How to assign credit to useful timesteps?

- **Long training times** due to dependencies on previous timesteps

- Solution: use **gating techniques** to regulate state updates (to *some* success)
  - Long Short-Term Memory (LSTM)
  - Gated Recurrent Unit (GRU)

Technical Difficulties… Dependency Problem



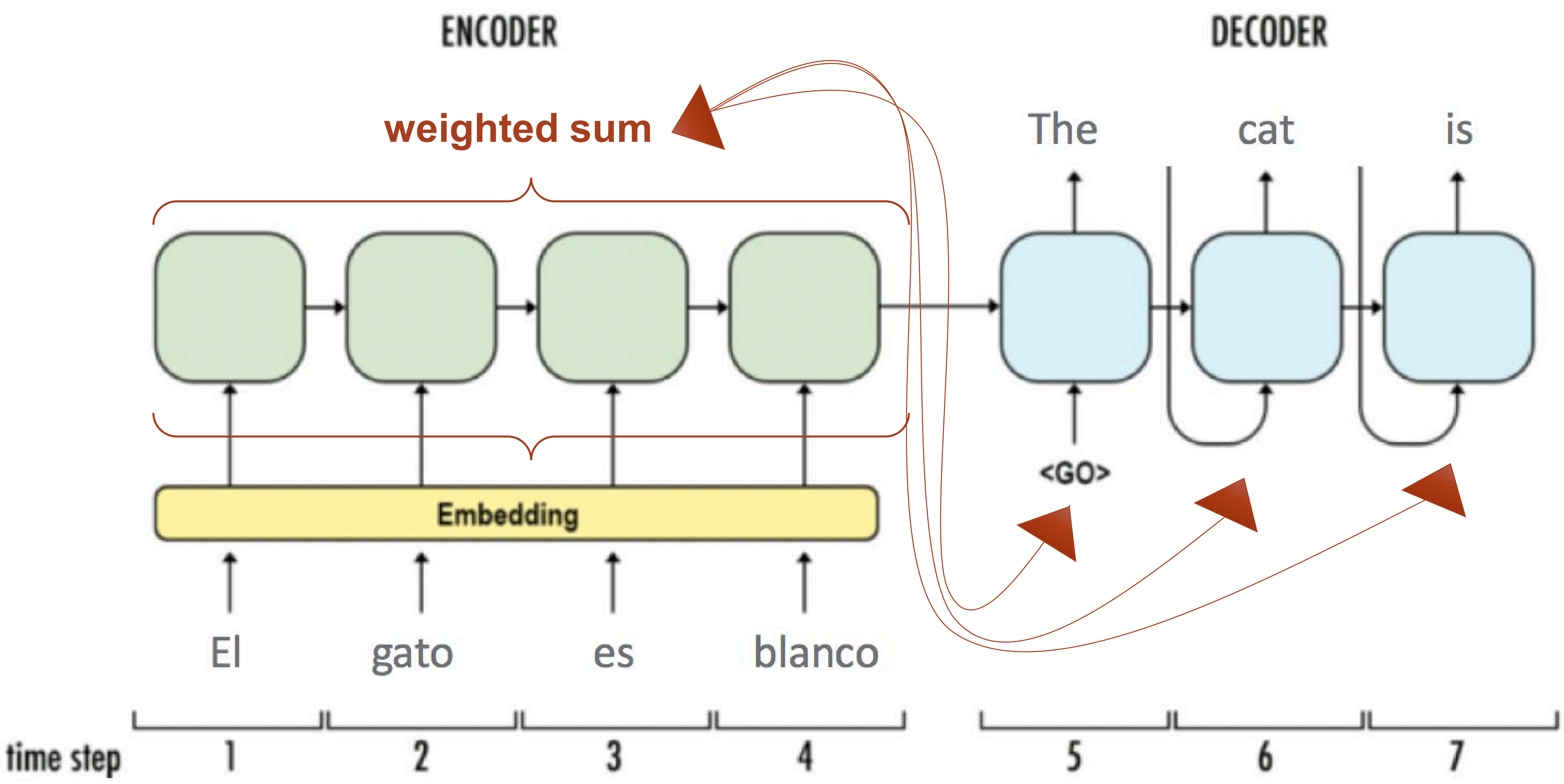
Attention Mechanisms: Overcoming the Bottleneck

- RNNs rely on internal dynamics to capture important timesteps
  - Via forget / update gates

- Attention mechanisms are a group of methods for capturing this explicitly
  - Typically, via **weighted sum over inputs**
  - We'll unpack this further in a few slides

Neural Translation By Jointly Learning to Align and Translate https://arxiv.org/pdf/1409.0473.pdf

Attention Mechanisms: Overcoming the Bottleneck

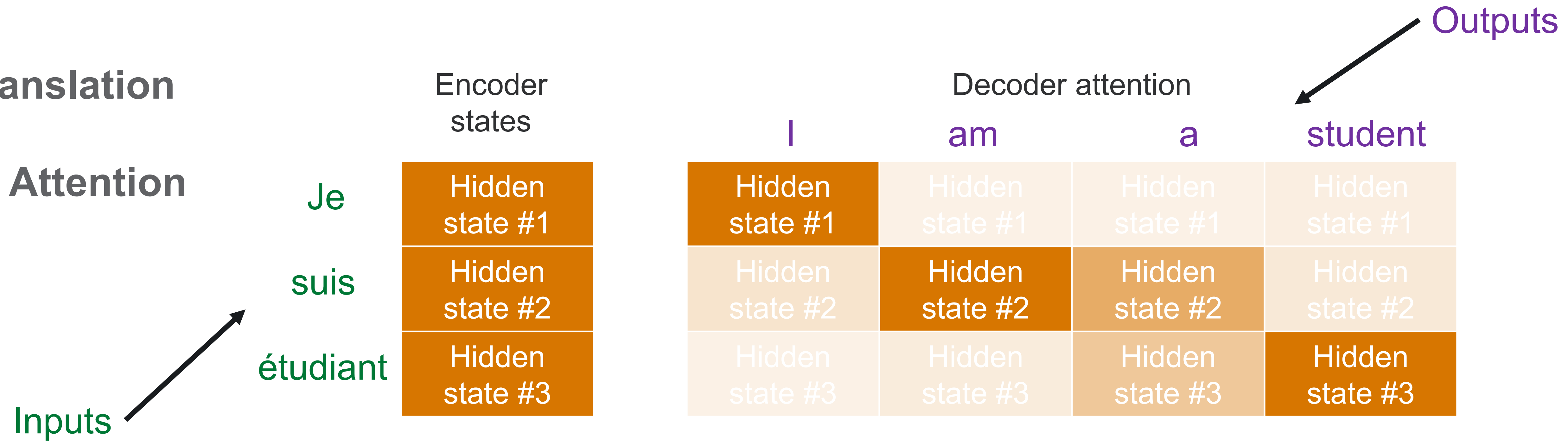


Seq2Seq + Attention

- To fix these issues, researchers introduced ***attention***
  - Especially common in seq2seq models, e.g. machine translation
- Attention creates an immediate mapping from inputs to outputs
  - Ignores timestep dimension that causes training issues for RNN

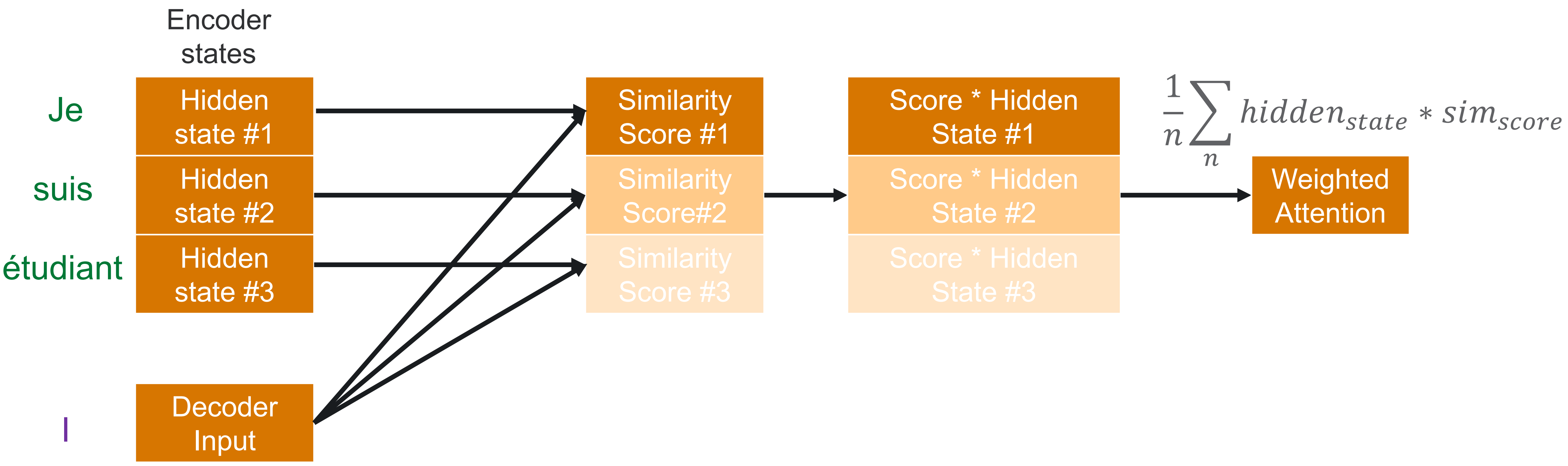
Machine Translation

Seq2Seq w/ Attention



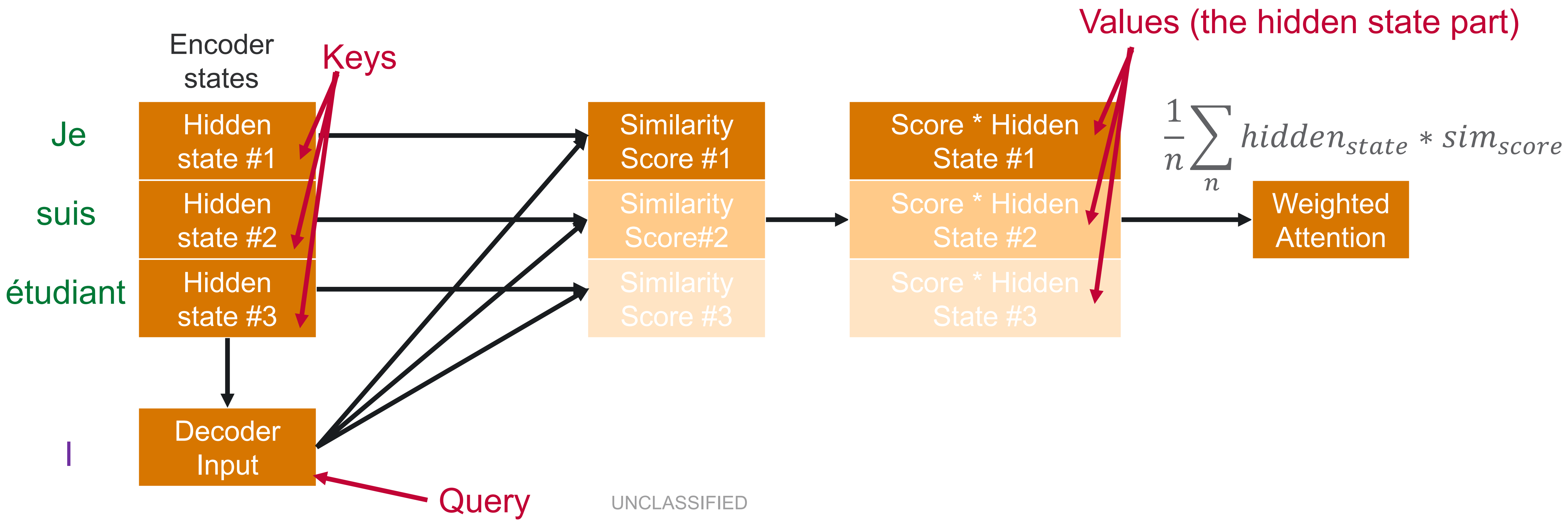
Seq2Seq – Additive Attention

- Attention – Weighting over inputs
  - Create an attention “score” for each input
  - Softmax over scores limits total amount of attention



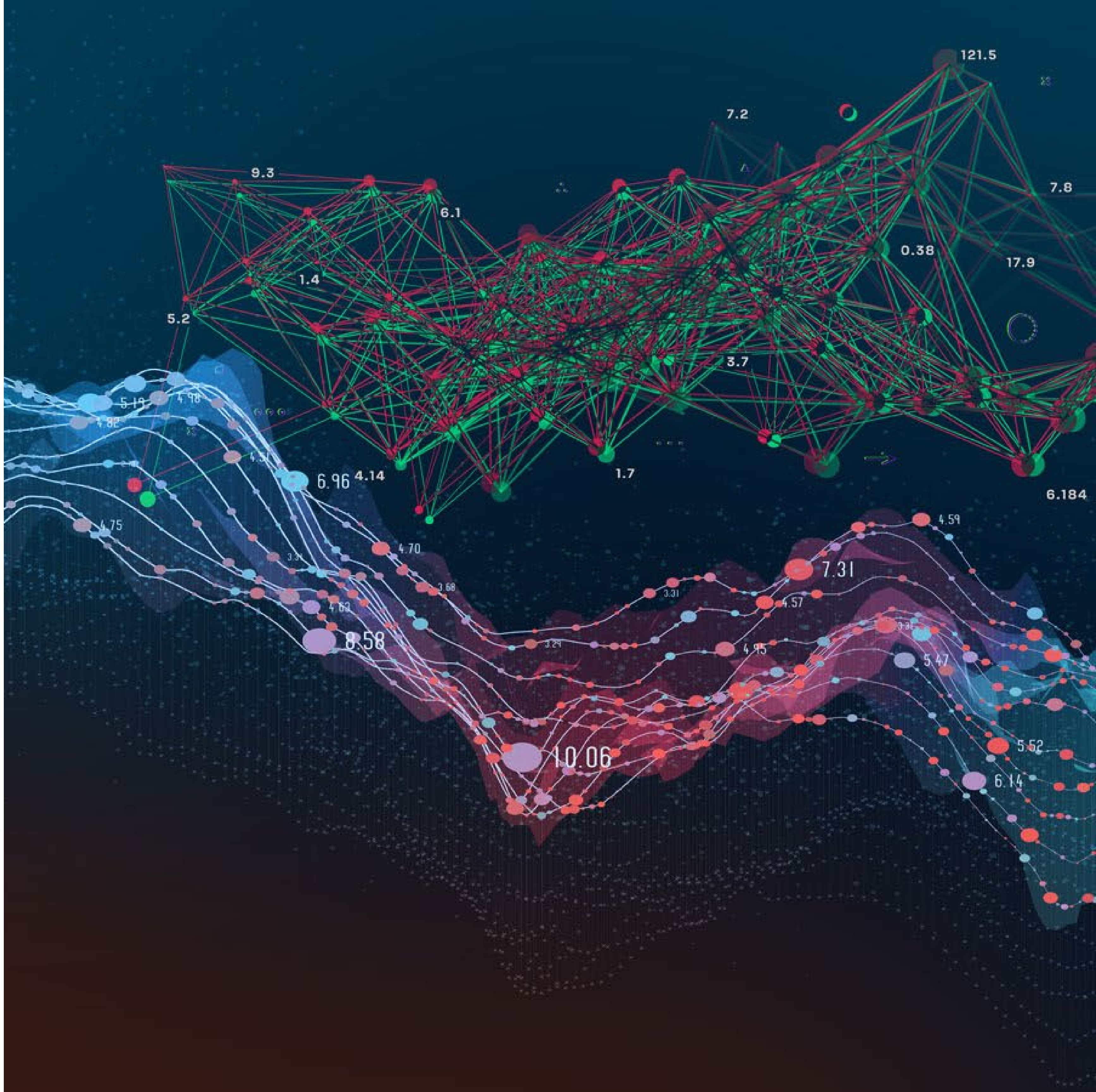
Attention – Queries/Keys/Values

- **Query** = Current state, tells us what to attend to (Decoder hidden state)
- **Key** = What is being attended over (Encoder hidden state)
- **Value** = What should be returned if the **Key** is matched (Encoder hidden state)



Pacific Northwest
NATIONAL LABORATORY

Transformers



U.S. DEPARTMENT OF ENERGY BATTELLE

PNNL is operated by Battelle for the U.S. Department of Energy

UNCLASSIFIED

Pacific Northwest NATIONAL LABORATORY

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

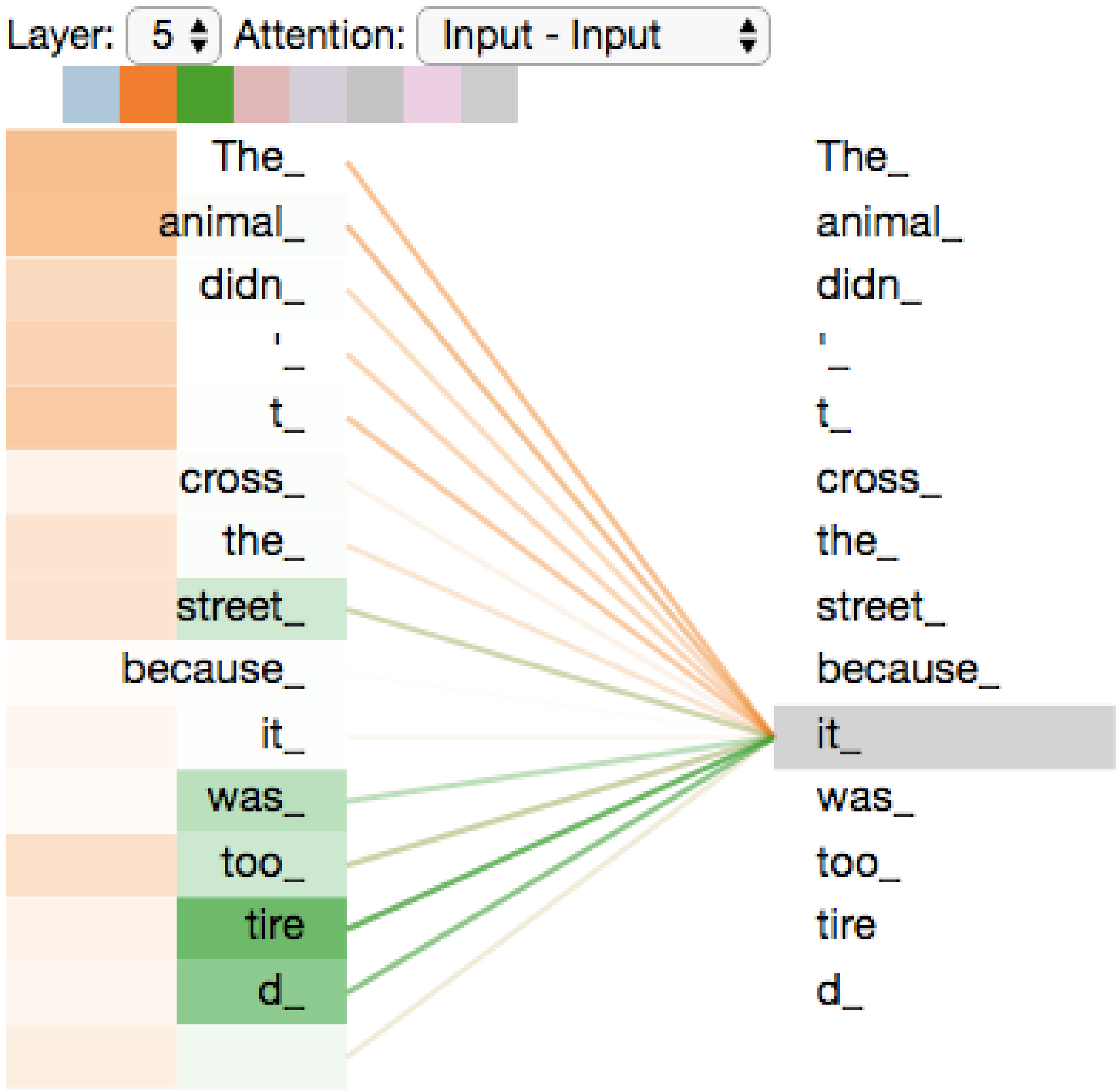
Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

Why are transformers more accurate?

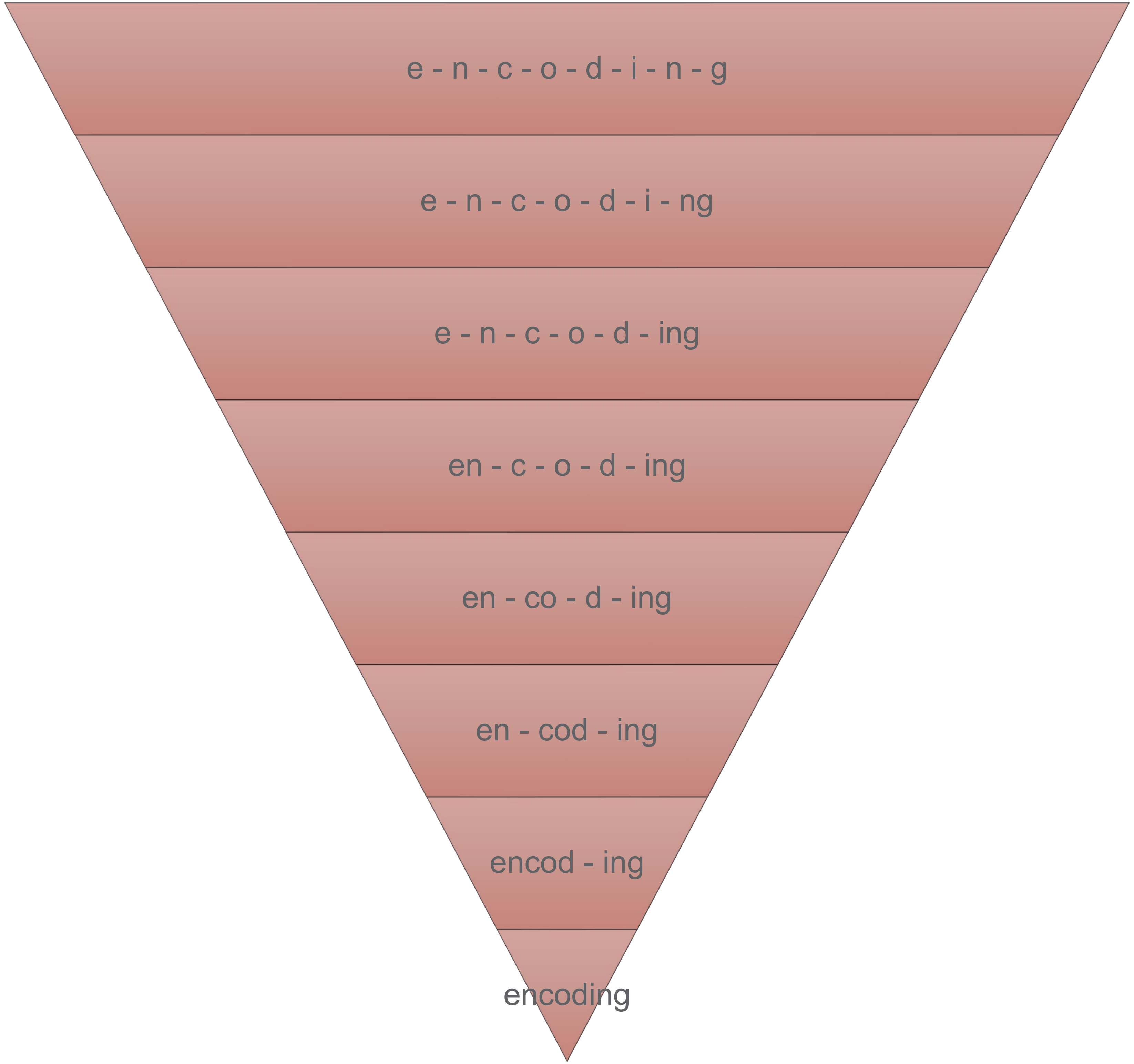
- Architecture
  - Context
  - Context!
  - Context!!
- Other factors
  - Data availability
  - Compute



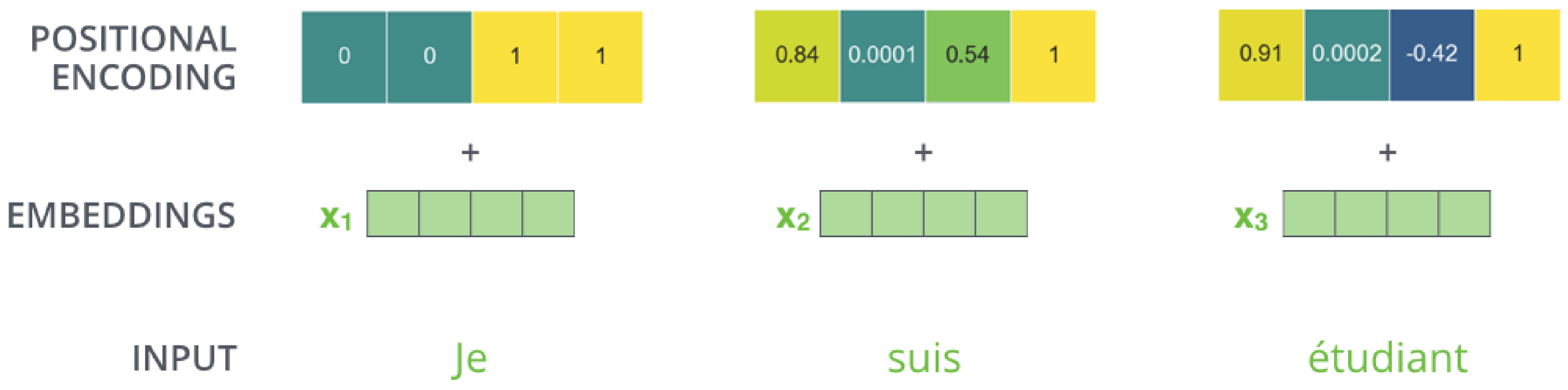
Remember BPEs?: Most transformers use them as a first step

Recap:

- BPE = byte pair encoding
- "Middle ground" between character- and word-level encodings
- Vocabulary-agnostic
- Probability-based

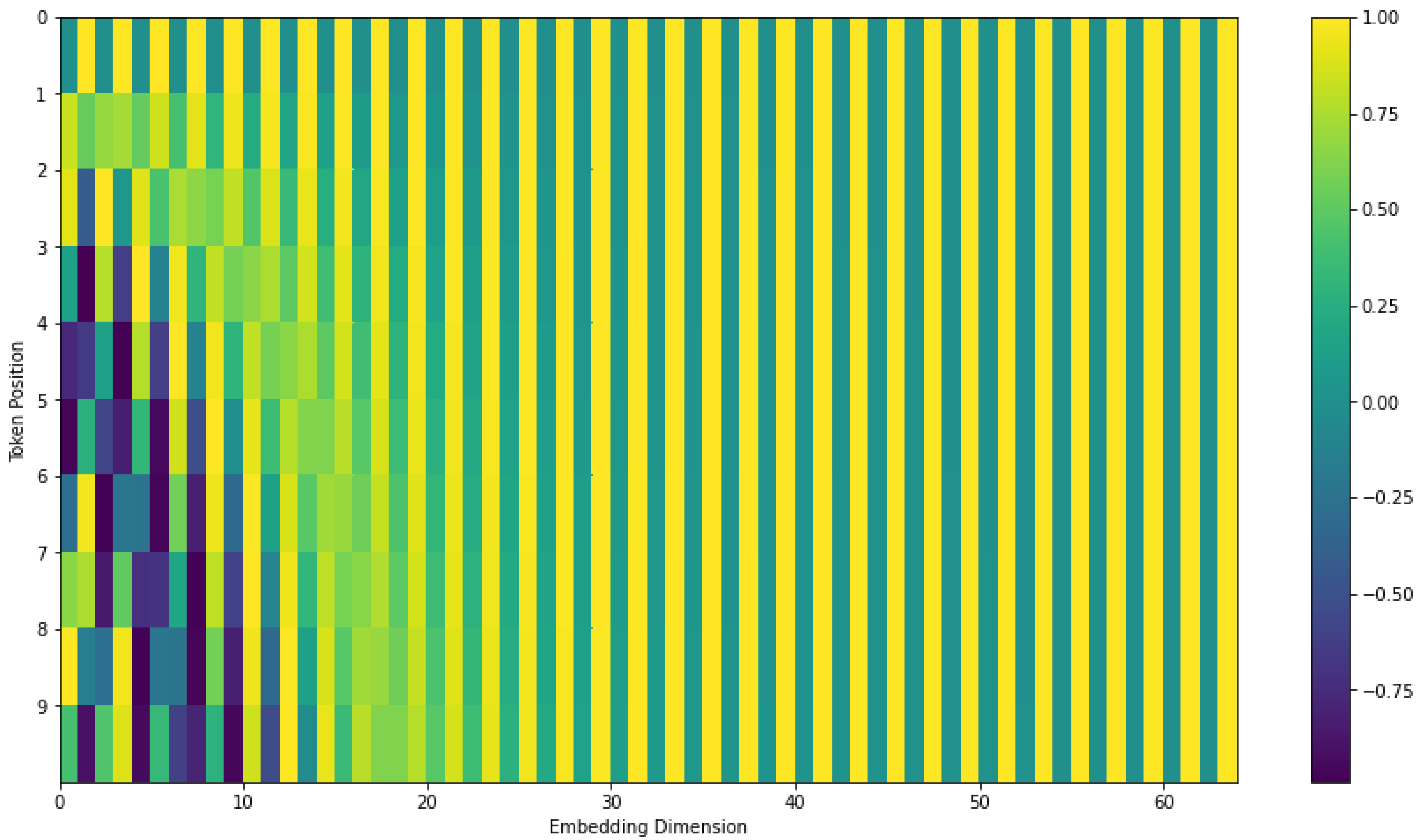


Sennrich, Rico, Barry Haddow, and Alexandra Birch. "Neural machine translation of rare words with subword units." *arXiv preprint arXiv:1508.07909* (2015).



Positional Encodings (PEs)

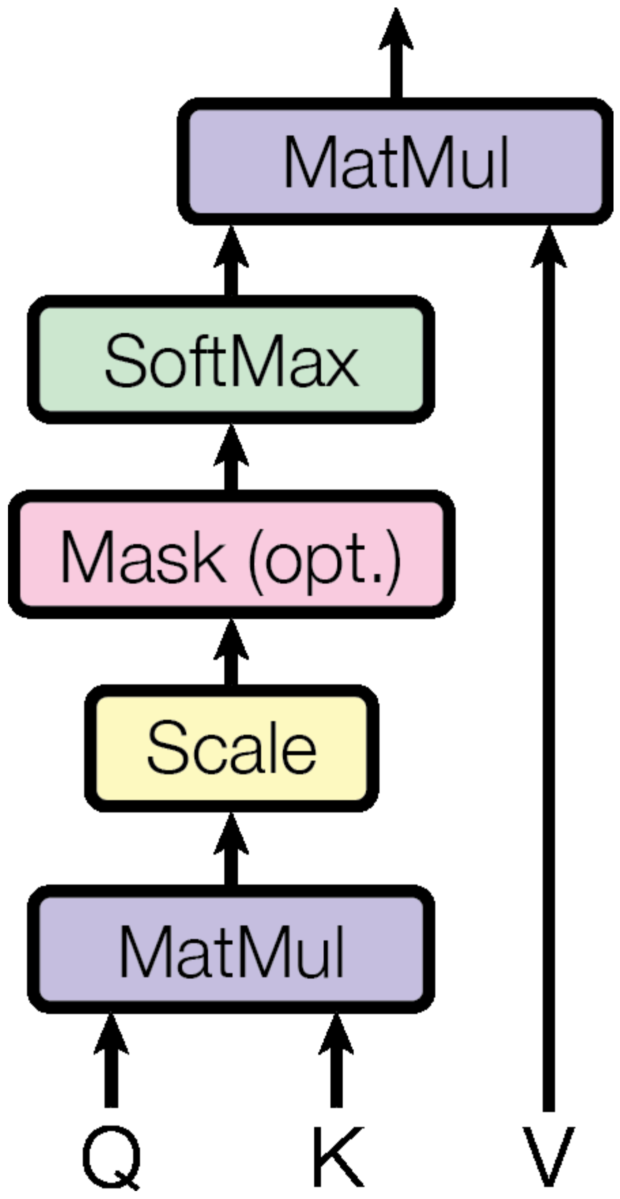
- Re-adds relative positioning to Transformer
- Cosine basis functions
- Added to BPEs



$$\mathrm{SDPA} = \mathrm{softmax}\left(\frac{\boldsymbol{QK}^T}{\sqrt{d_K}}\right)\boldsymbol{V}$$

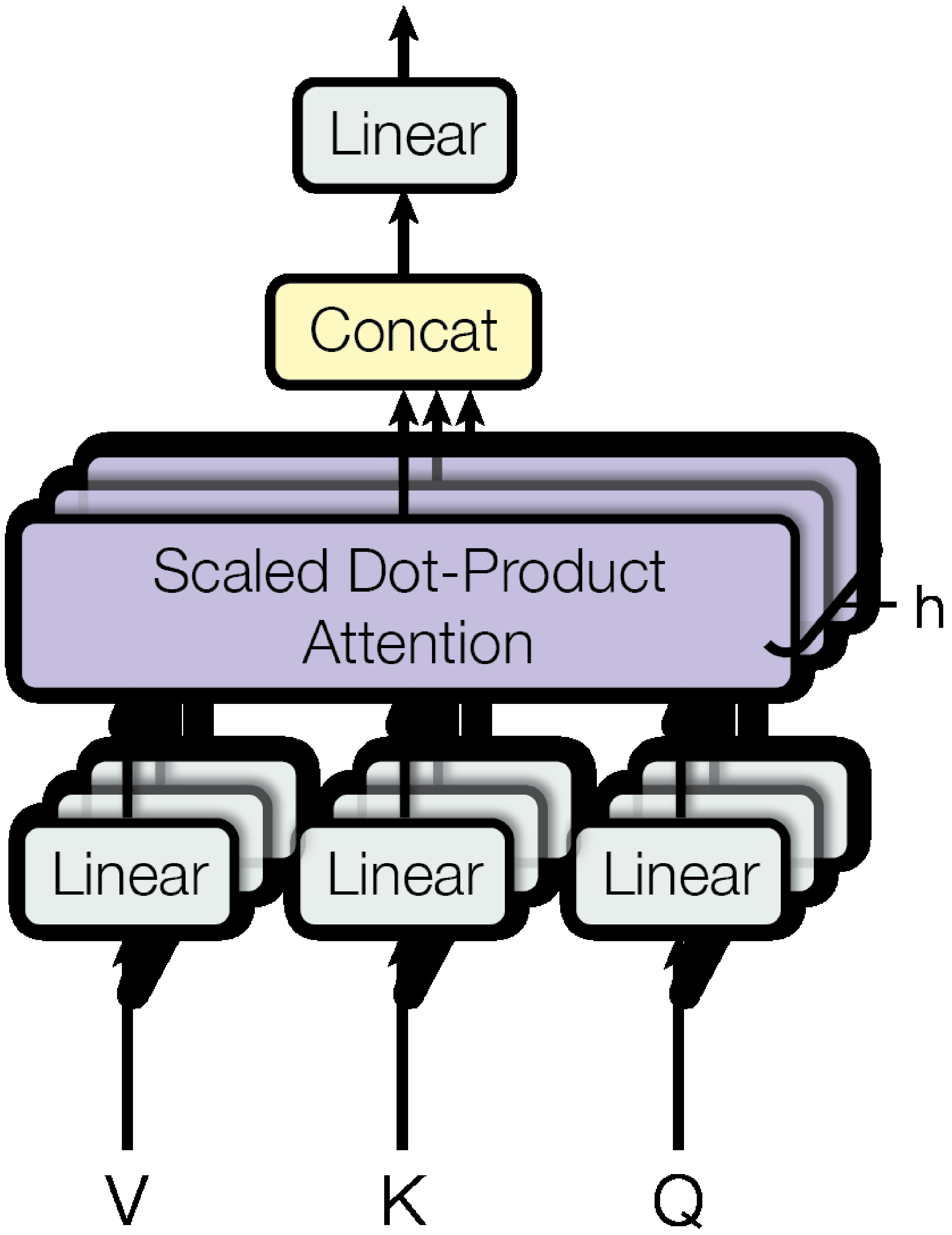
Scaled Dot-Product Attention (SDPA)

- d_k is the embedding dimensionality
- Allows for infinite-distance information to be retained
- Query, Key, and Values are all generated from linear combinations of the same input (Q and K are the same)



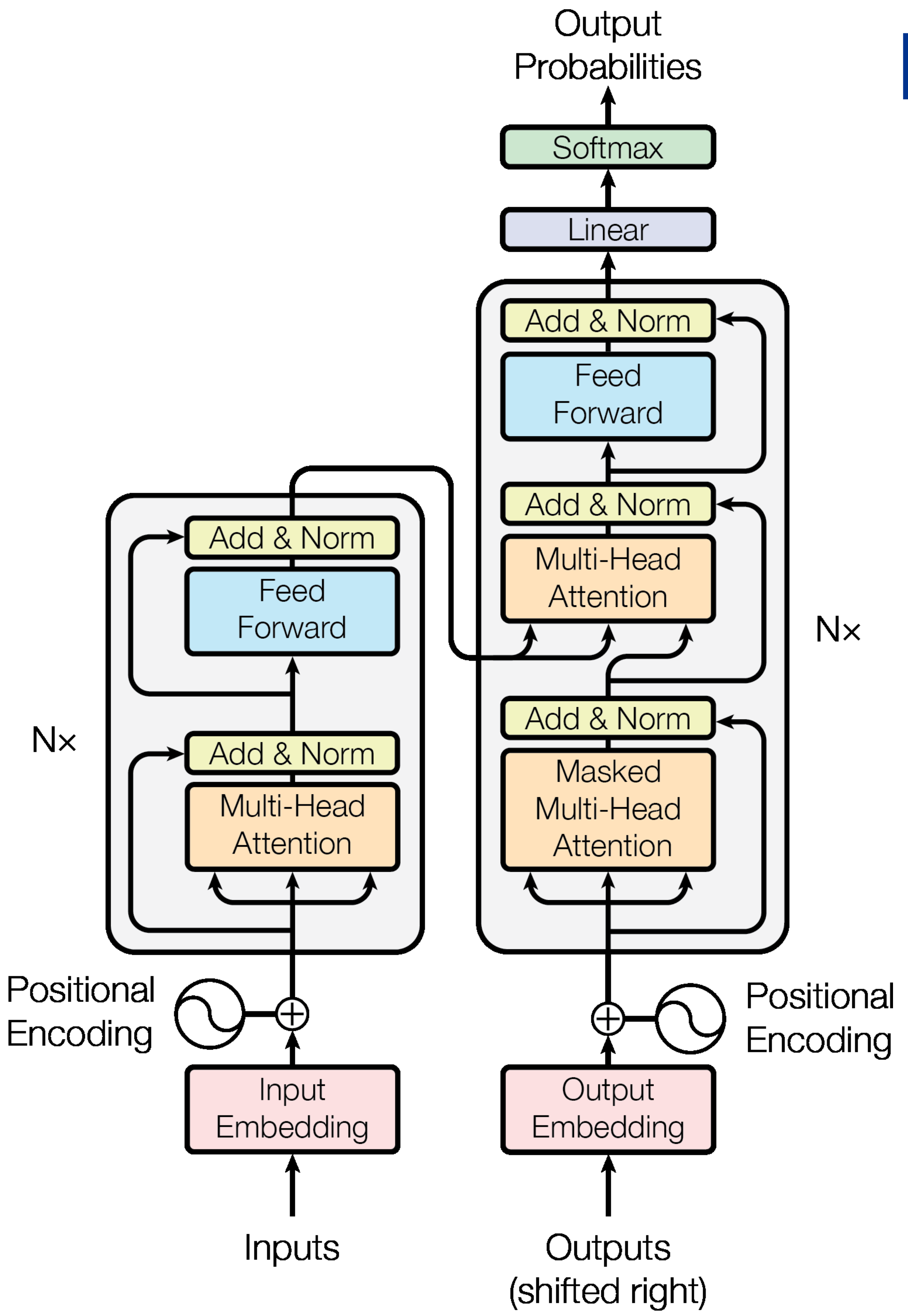
Multi-Headed Attention

- Performs SDPA operation in parallel across *h* heads
- In the original paper,
  - $h = 8$
  - $d_K = d_V = d_m/h = 68$



Encoder & Decoder

- Designed for translation, nowadays most use one or the other
- In the original paper,
  - $N = 6$
- Decoder uses *masking*, any position cannot be "seen" by later positions



APPENDIX

Performance

- n is the sequence length
- d is the encoding dimensionality
- k is the kernel size

Computational Complexity

- Self-Attention (SA) $\mathcal{O}(n^2 \cdot d)$
- RNNs $\mathcal{O}(n \cdot d^2)$
- CNNs $\mathcal{O}(k \cdot n \cdot d^2)$

Parallelized Operations

- SA $\mathcal{O}(1)$
- RNNs $\mathcal{O}(n)$
- CNNs $\mathcal{O}(1)$

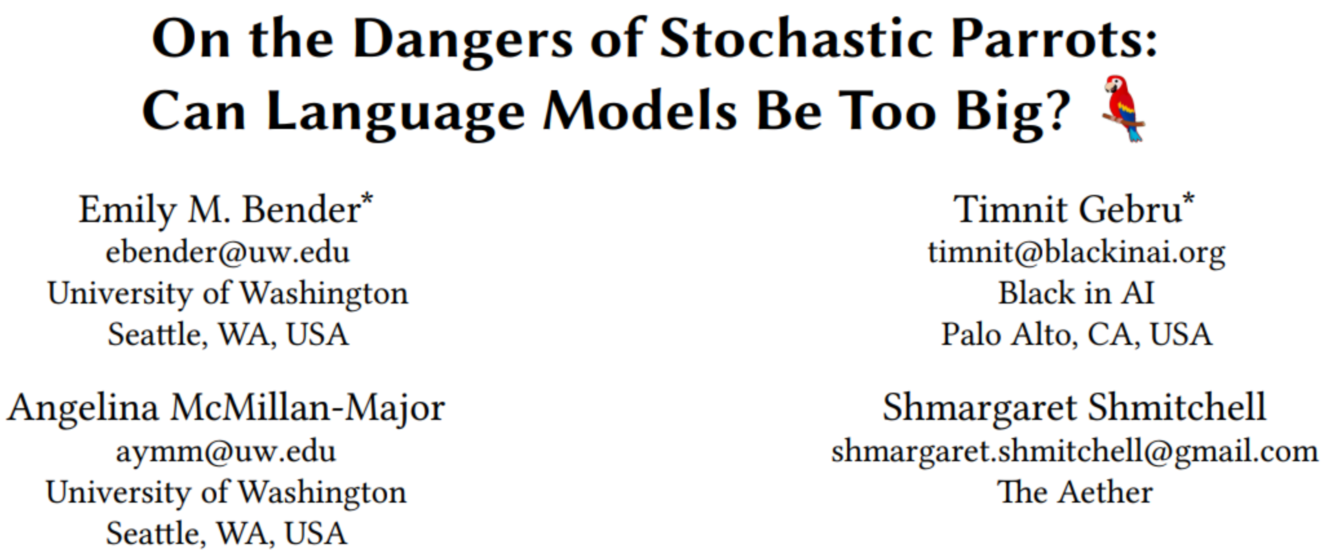
Path Length Between Long-Range Dependencies

- SA $\mathcal{O}(1)$
- RNNs $\mathcal{O}(n)$
- CNNs $\mathcal{O}(\log_k(n))$

Some commonly used Transformer models

- **General Purpose**
  - RoBERTa, XLNet, T5, LLaMA, Mixtral
- **Text Generation**
  - GPT-2, GPT-neo, T5, LLaMA, Falcon, Mixtral
  - Proprietary: GPT-3, GPT-4, ChatGPT, Claude 3, Gemini
- **Long Sequences**
  - Longformer, Reformer, *YaRN*
- **Limited Compute Resources**
  - DistilBERT, ELECTRA, *vLLM*

Transformer Concerns


On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? 🦜

Emily M. Bender*
ebender@uw.edu
University of Washington
Seattle, WA, USA

Timnit Gebru*
timnit@blackinai.org
Black in AI
Palo Alto, CA, USA

Angelina McMillan-Major
aymm@uw.edu
University of Washington
Seattle, WA, USA

Shmargaret Shmitchell
shmargaret.shmitchell@gmail.com
The Aether

- State-of-the-art performance comes with significant costs

- Pretraining data
  - Almost all transformers are trained on significant quantities of internet data which has never been closely inspected
  - Assume your model has inherited internet biases…

- Text generation is *fluent* but *nonsensical*
  - Cannot be relied on to generate facts/truthful statements
  - High performance != Language understanding

NLP Models

- Transformers have lots of hype!
  - Also concerns
  - And for simple tasks, TF-IDF and other features may be sufficient

- HuggingFace provides simple API for common tasks
  - But there are other options out there if you prefer

Questions? Comments?

Up Next: Wrap Up

Pacific Northwest
NATIONAL LABORATORY

What We Covered Today

Morning

- Text as a Modality
- Common NLP Tasks
- Ethics & Bias
- Preprocessing Basics

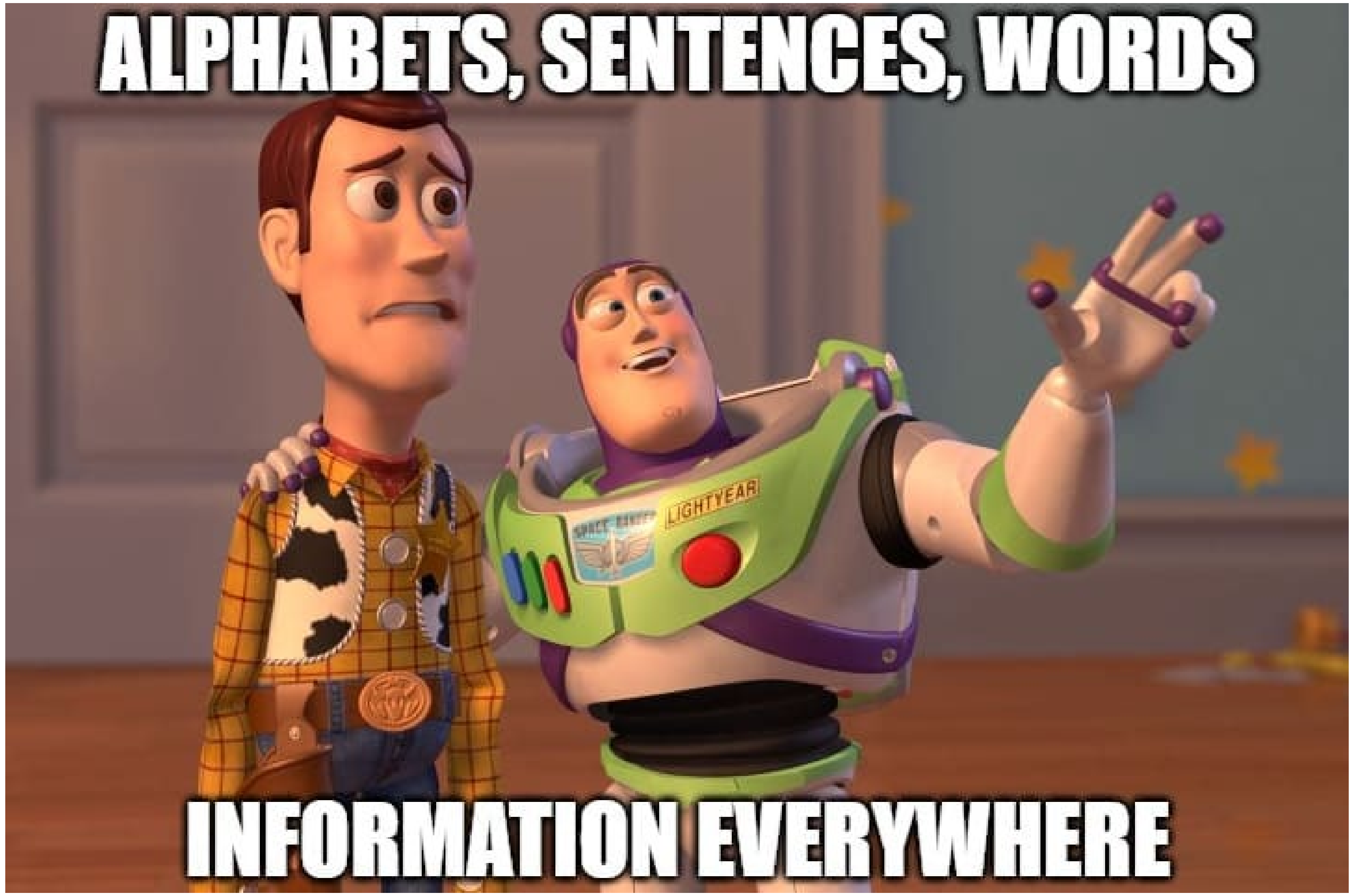
Afternoon

- Available Tools
- Text Representations
- Modeling
- Transformers

Pacific Northwest
NATIONAL LABORATORY

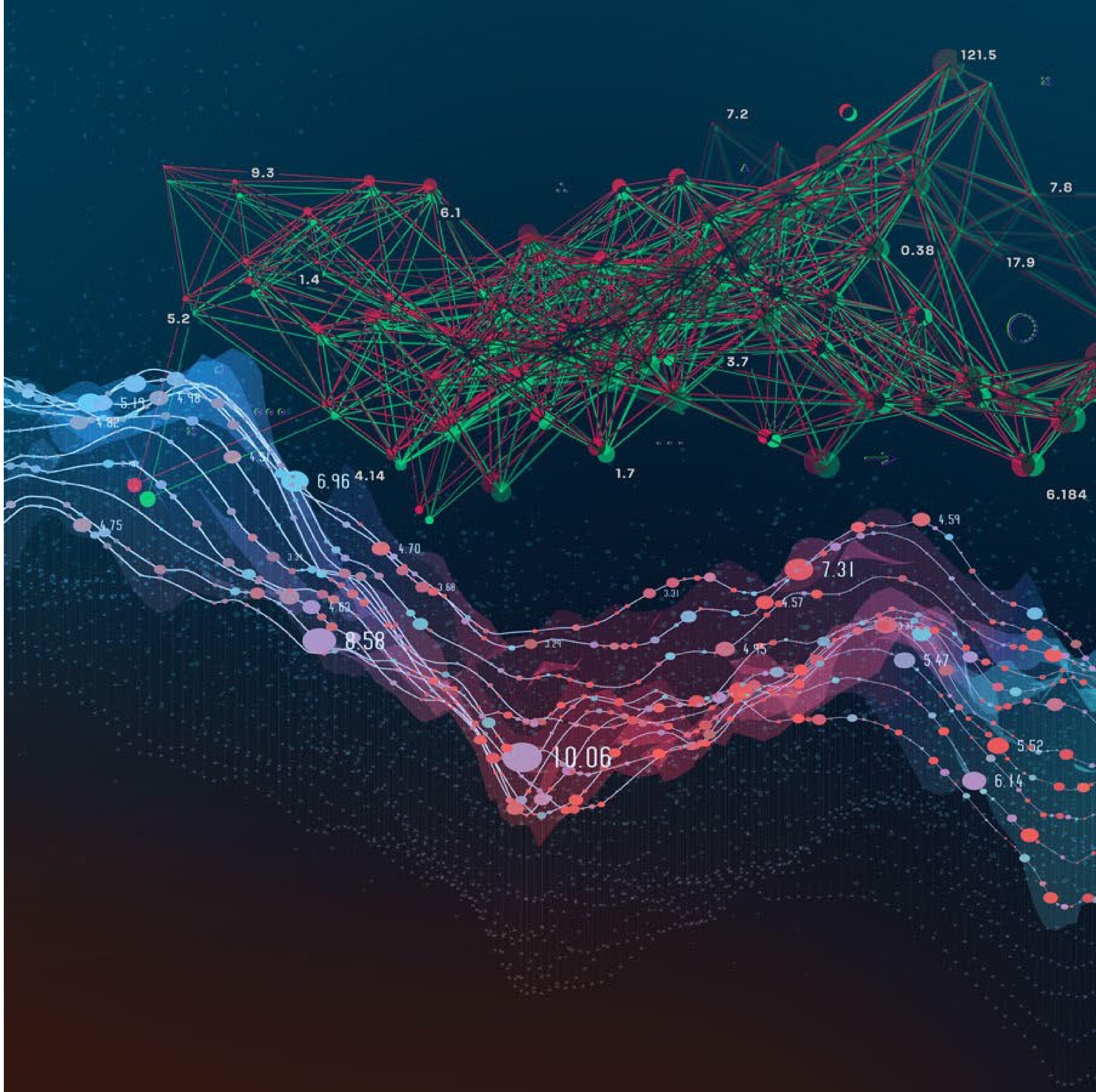
Thank You!

PNNL-SA-196905



Pacific Northwest NATIONAL LABORATORY

Deep Learning Appendix

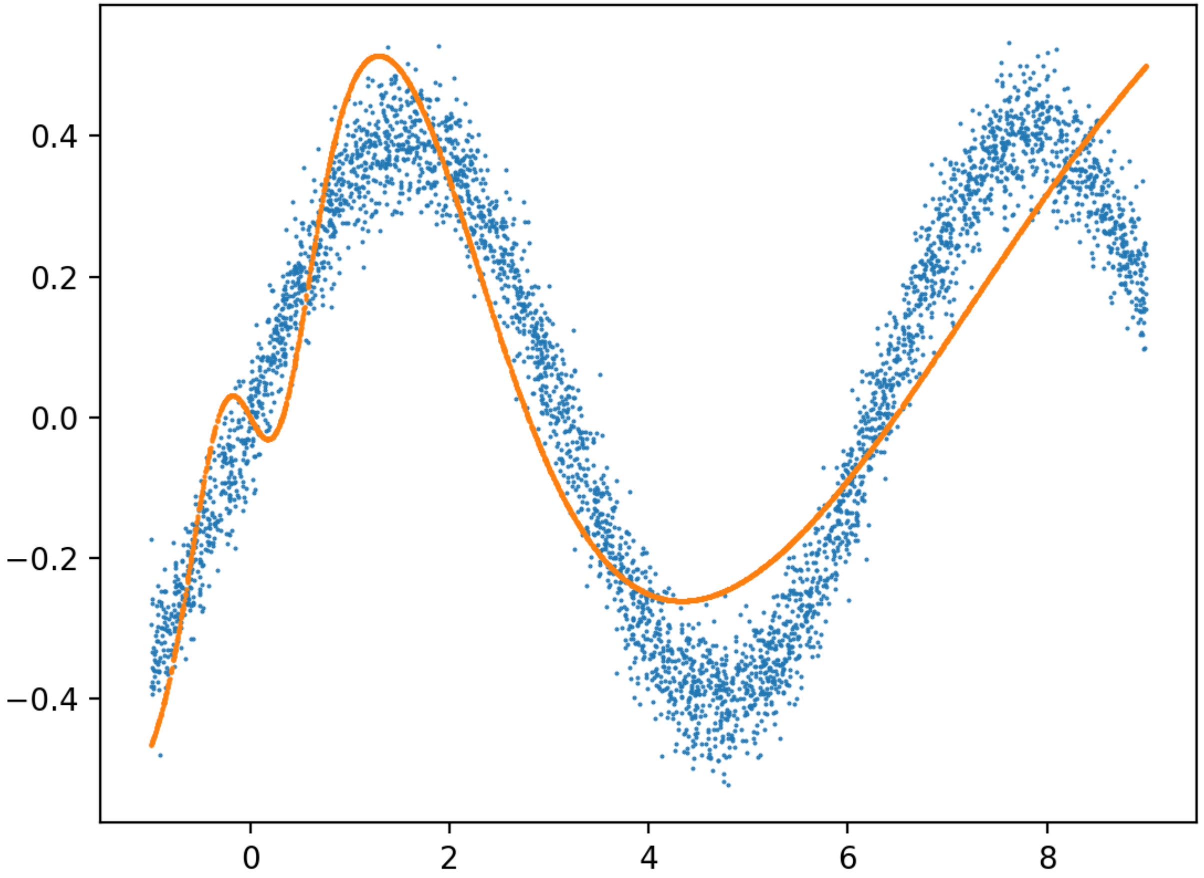


U.S. DEPARTMENT OF ENERGY BATTELLE

PNNL is operated by Battelle for the U.S. Department of Energy

Neural Networks Are Function Approximators

- The real world is complex, and it is impossible to capture every detail
- Neural network modeling is about function approximation
  - Model observed behavior as a function
  - Map inputs to expected outputs
- **Contrary to popular belief, neural networks are not brains!**



How well we can do this is based on:

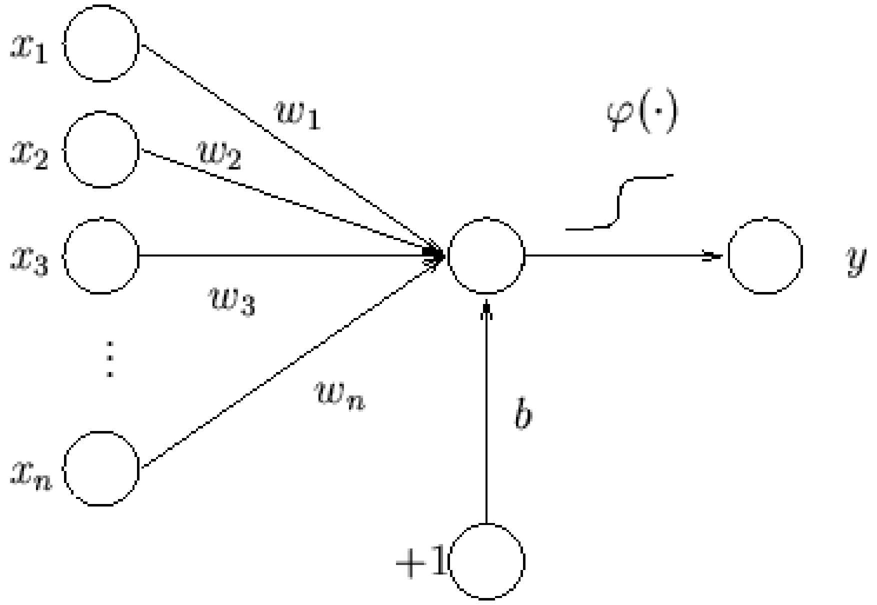
- **Available data** (quantity and quality)
  - Do we have enough data to describe the behavior of interest?
- **“Capacity” of the model**
  - How complex of a function can it represent?
  - How many learnable parameters does it have?
- **Confusing correlation and causation**
  - i.e., a connection between two variables that appears to be causal but is not

Neural Networks: What is a neuron?

- A neuron is a type of **computational unit**. Neural networks are a collection of neurons with **weighted connections** to each other

$$y = \varphi(W\vec{x} + b)$$

Inputs ($\vec{x}$)

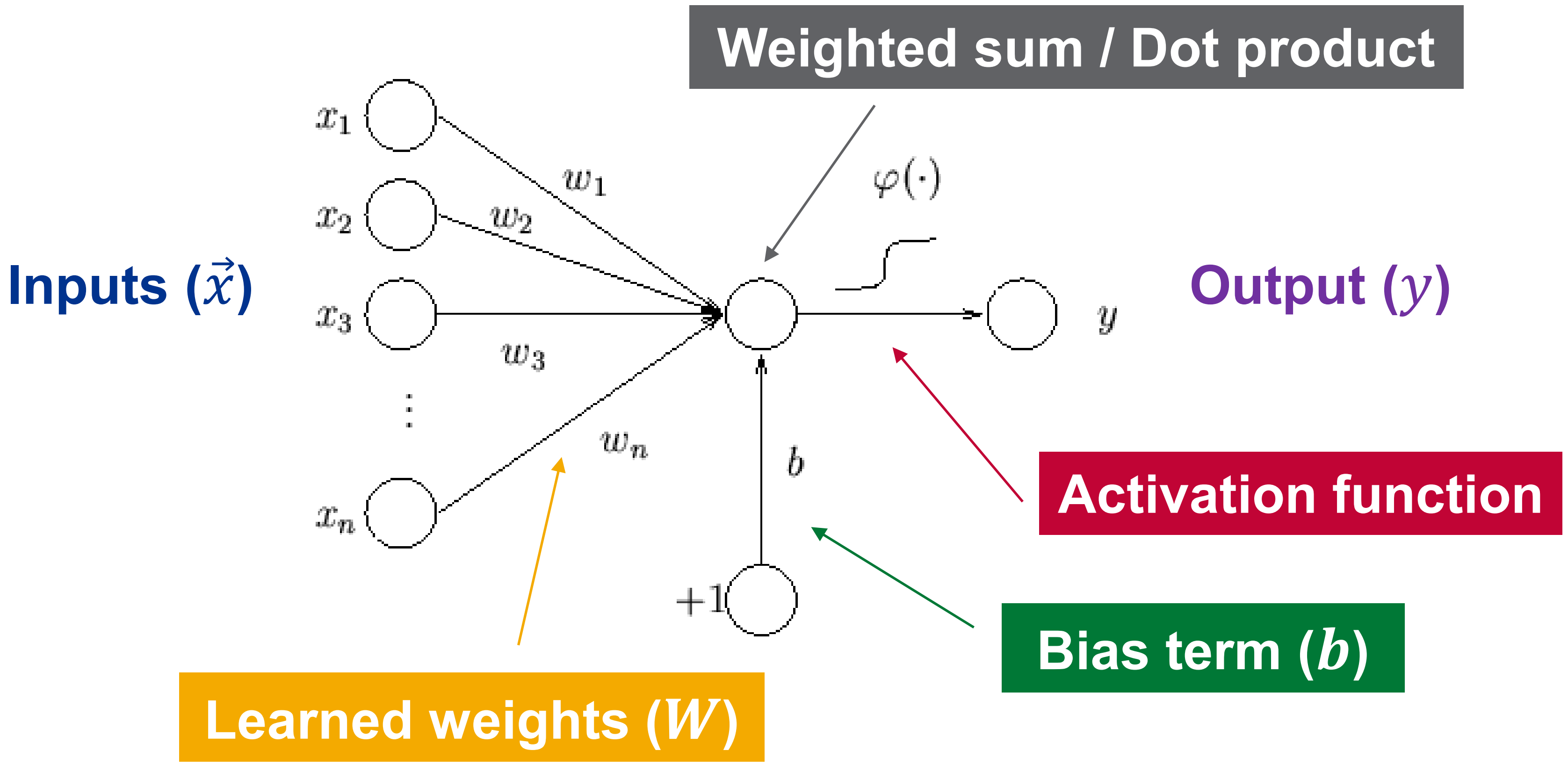


Output (y)

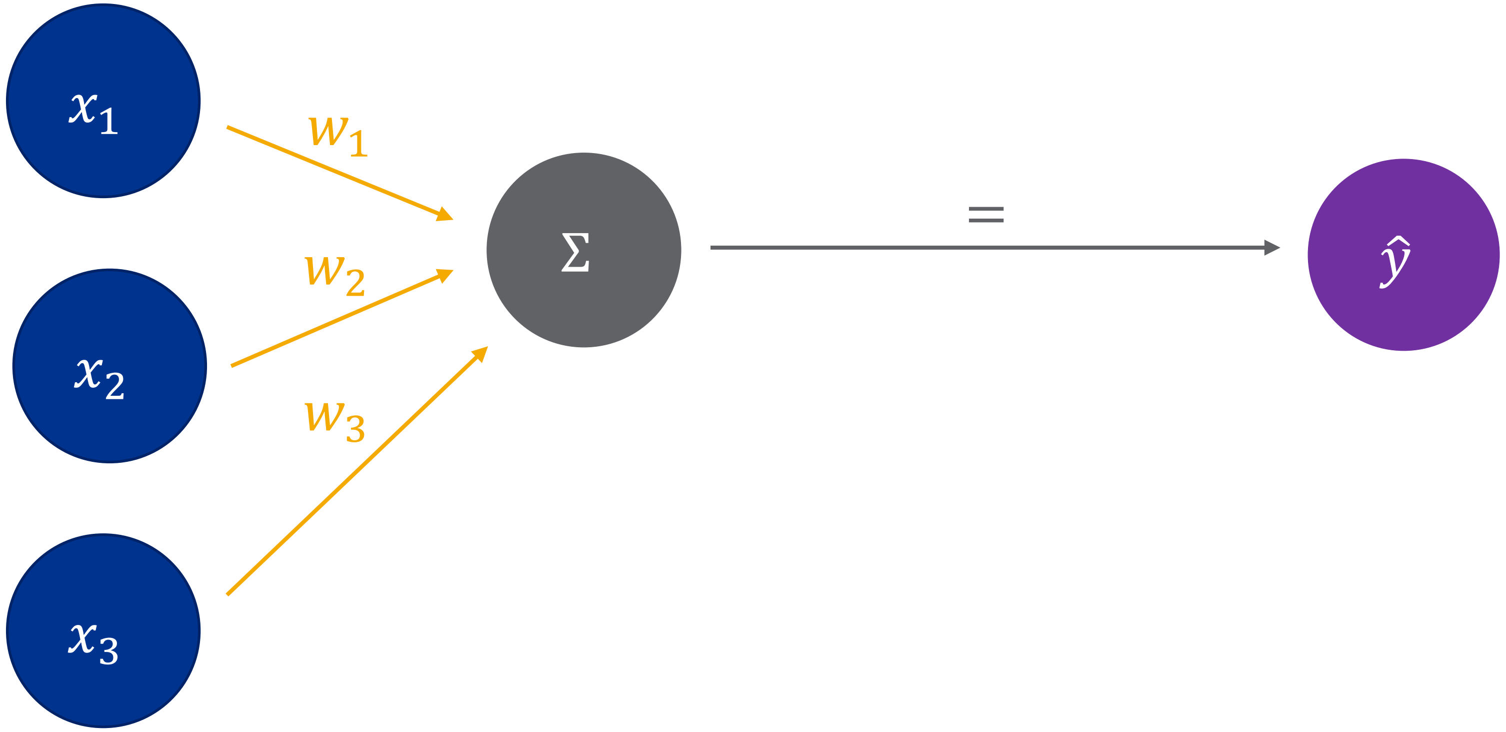
Neural Networks: What is a neuron?

- A neuron is a type of **computational unit**. Neural networks are a collection of neurons with **weighted connections** to each other

$$y = \varphi(W\vec{x} + b)$$



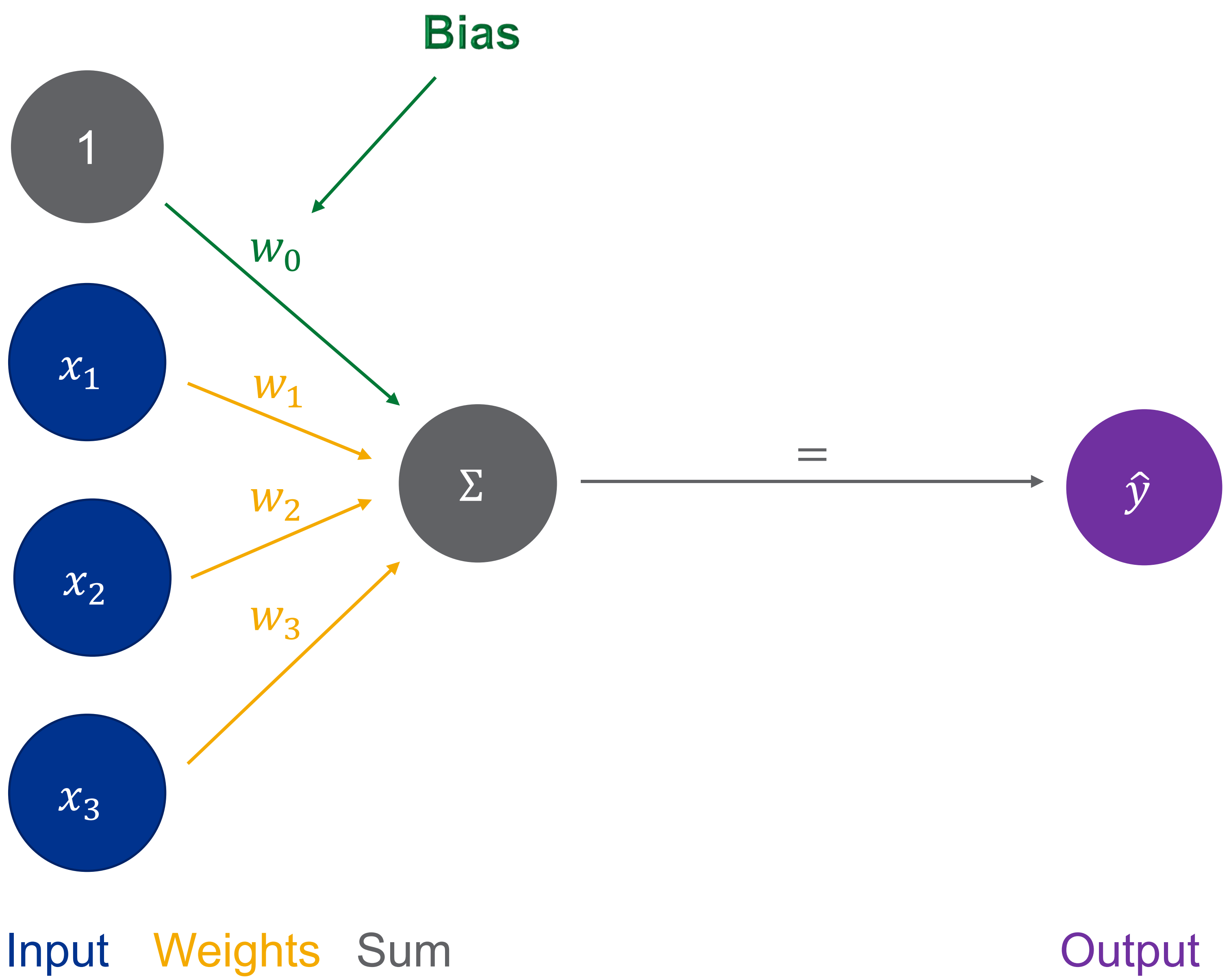
Function Approximation: Linear



$$\hat{y} = \left(\sum_{i=1}^{3} w_i x_i \right)$$

Input Weights Sum Output Equation

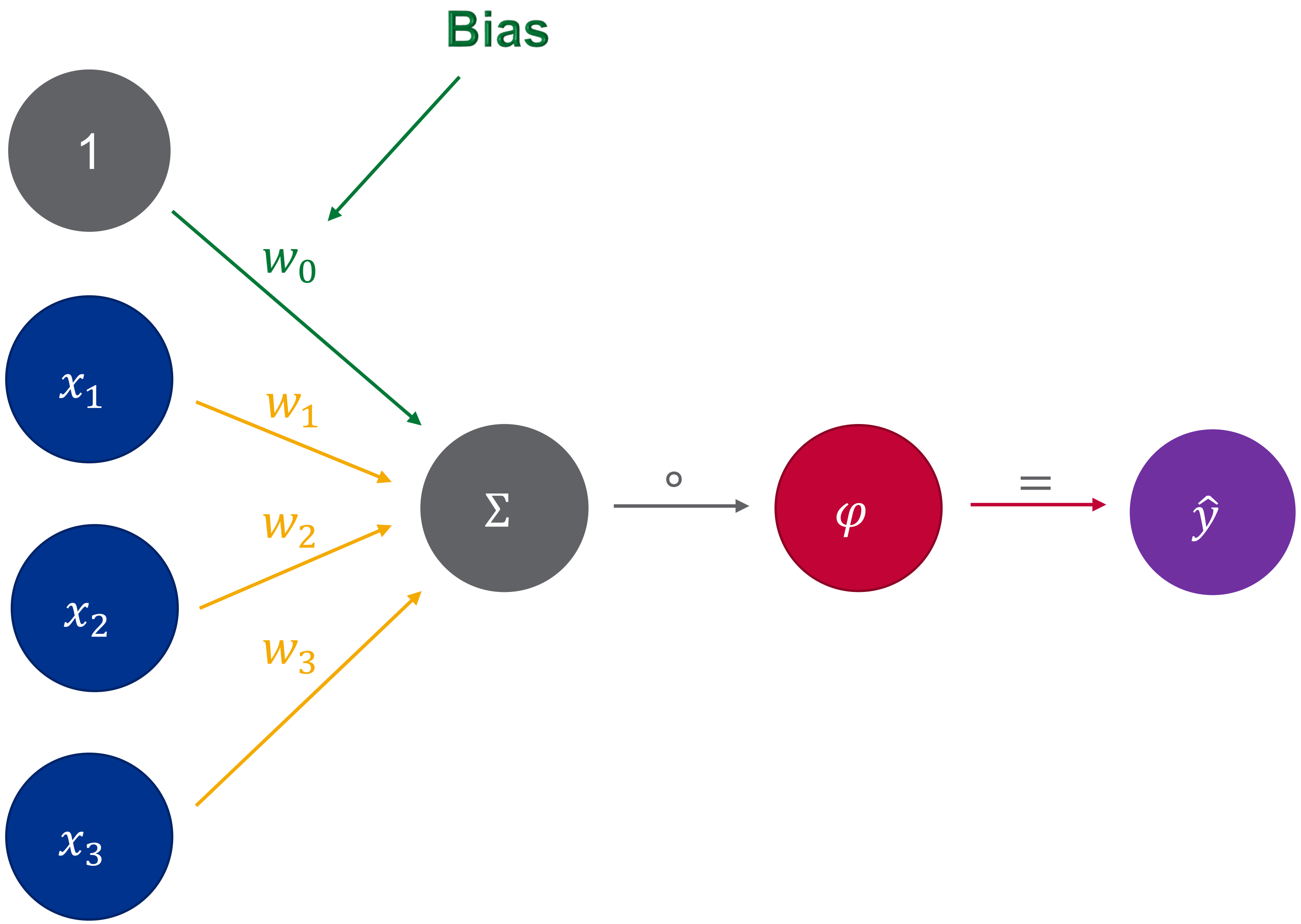
Function Approximation: Affine



$$\hat{y} = \left(w_0 + \sum_{i=1}^{3} w_i x_i \right)$$

Equation

Function Approximation: Nonlinear

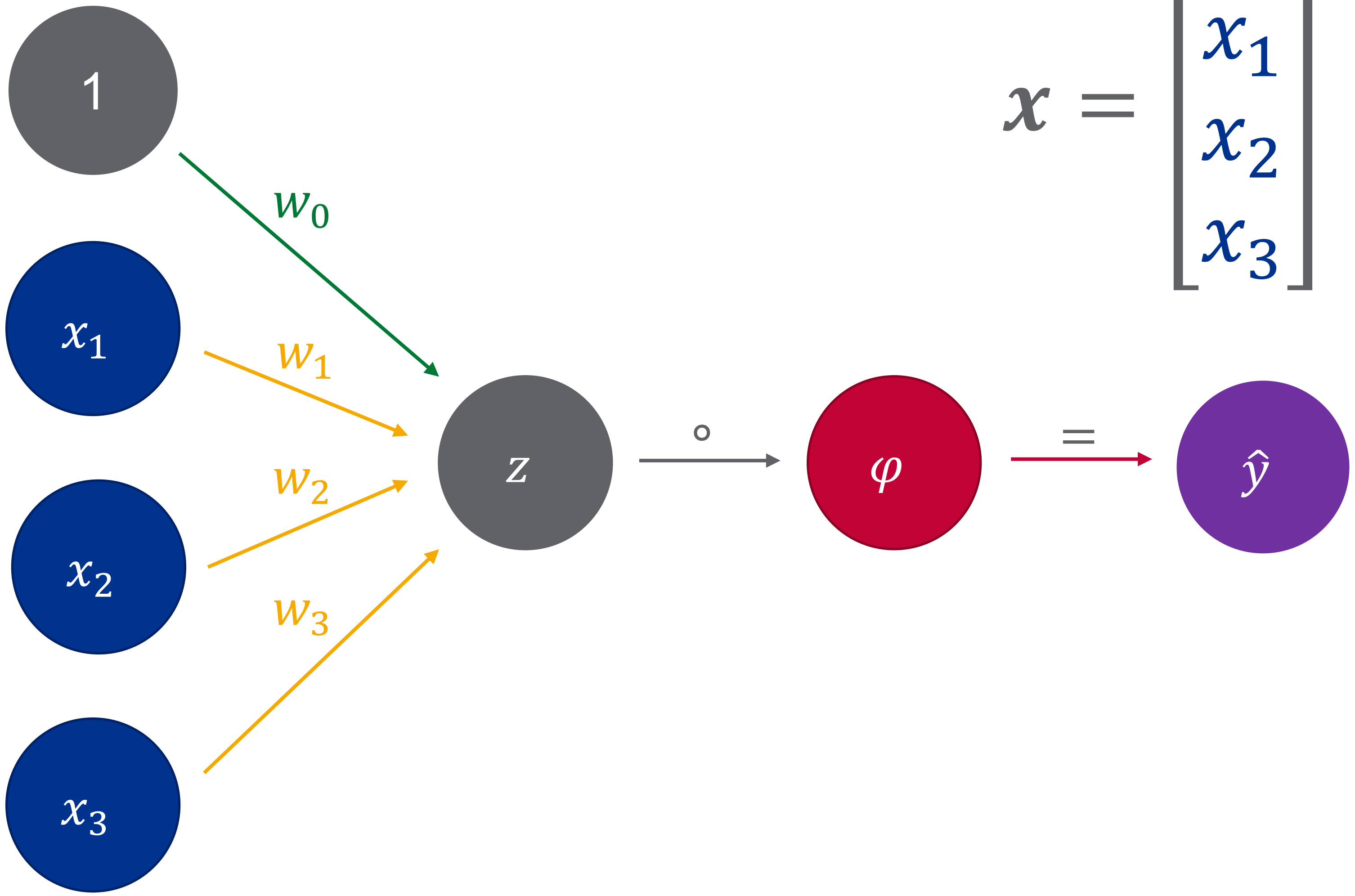


$$\hat{y} = \varphi\left(w_0 + \sum_{i=1}^{3} w_i x_i\right)$$

Input Weights Sum Composition Nonlinearity Output

Equation

Function Approximation: Matrix Notation



$$\boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} \quad z = \boldsymbol{w}^T \boldsymbol{x}$$

$$\hat{y} = \varphi(z)$$

Pacific Northwest NATIONAL LABORATORY

Neural Networks: What does a neuron do?

$$y = \varphi(\boldsymbol{W}\vec{\boldsymbol{x}} + \boldsymbol{b})$$

- **Features / inputs (vector)**
  - The inputs to the neural network. Contains the features of the data
  - e.g., image of a cat or dog as a vector
- **Weights (matrix)**
  - Apply to input vector using matrix multiplication
  - Typically initialized randomly, then updated ("learned") during training
- **Bias (vector)**
  - Added to the result of weight / input multiplication
  - Typically initialized randomly, then updated ("learned") during training

Neural Networks: What does a neuron do?

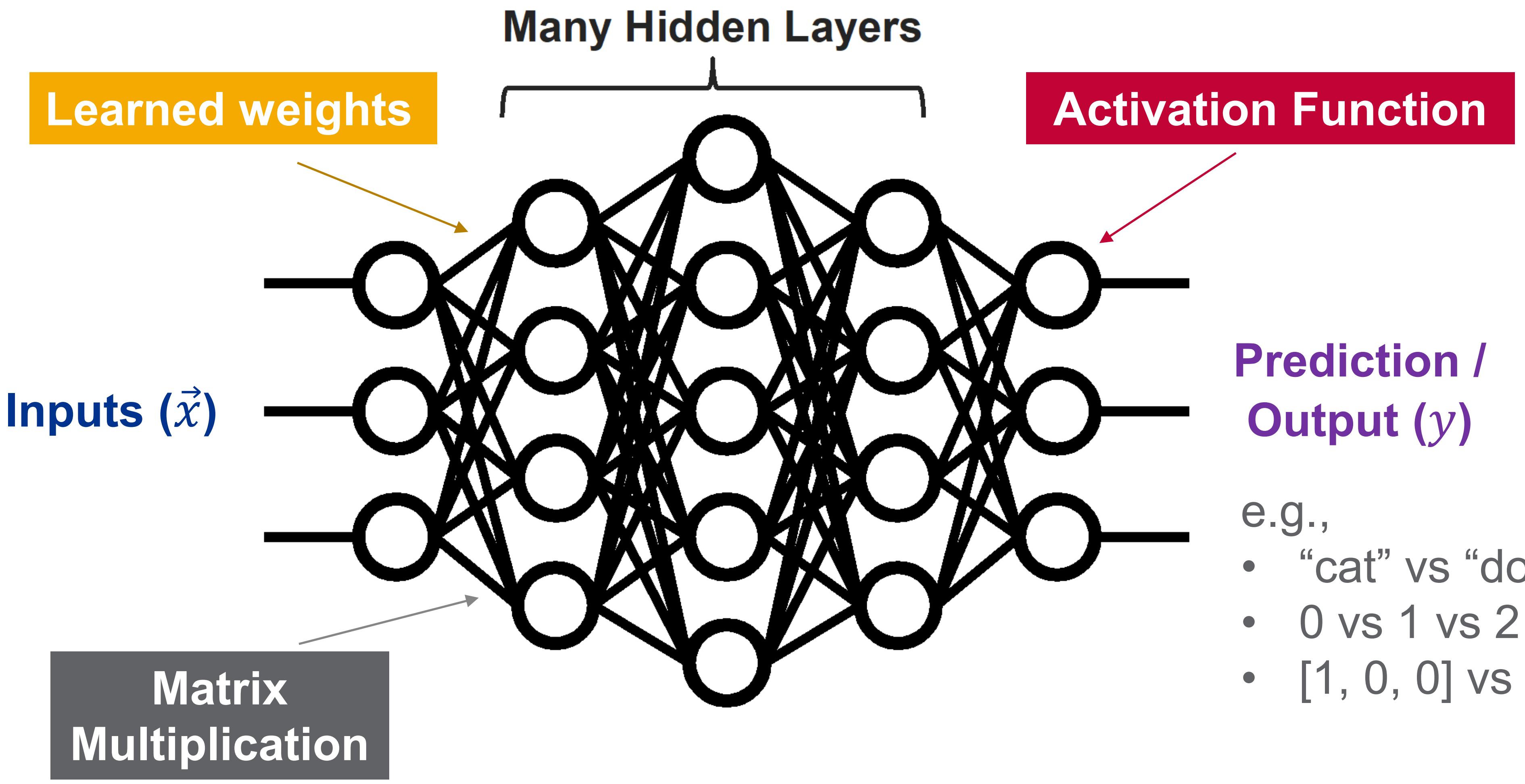
$$y = \varphi(W\vec{x} + b)$$

- **Activation function**
  - Applied to the result of the weight / input / bias computation
  - Typically nonlinear, to increase model capacity / expressivity

- **Output**
  - Result of entire neuron computation
  - A single neuron produces a single number
  - Classification problems use 1-hot vectors
  - Often consider a set of activations as a vector

Deep Neural Networks

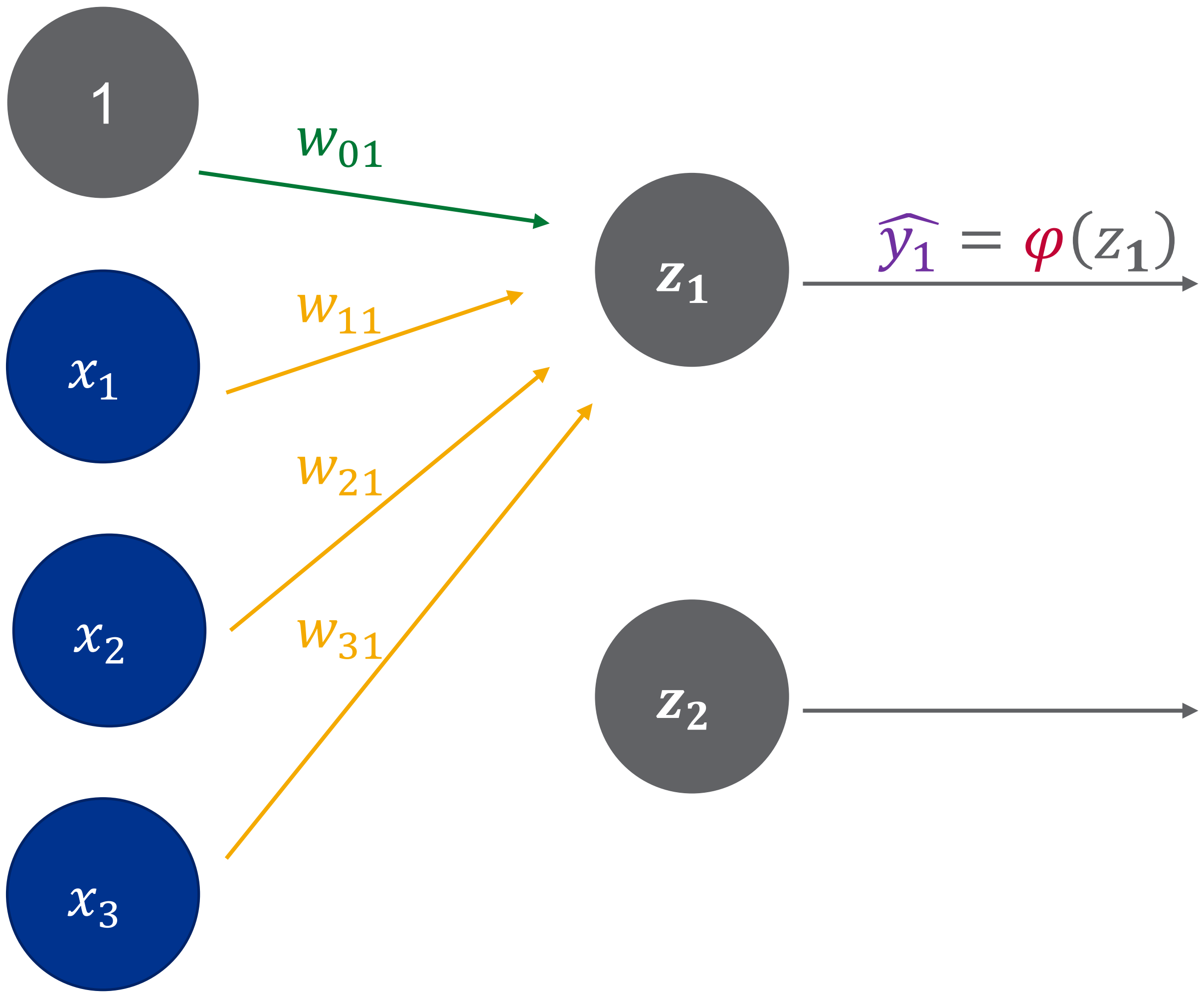
- Made up of **multiple layers of neurons** which learn a hierarchical data representation



e.g.,

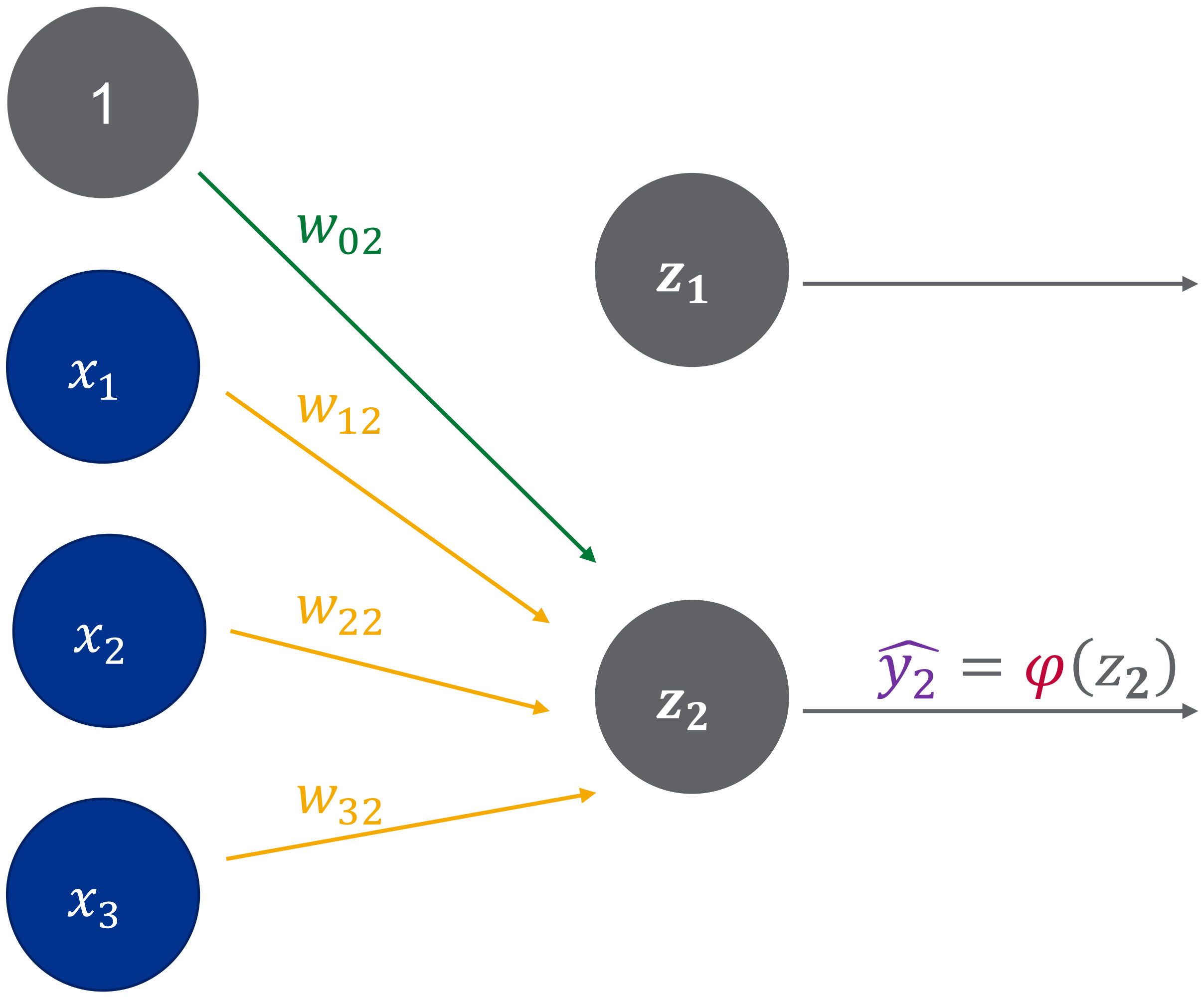
- "cat" vs "dog" vs "fish"
- 0 vs 1 vs 2
- [1, 0, 0] vs [0, 1, 0] vs [0, 0, 1]

Function Approximation: Multiple Neurons



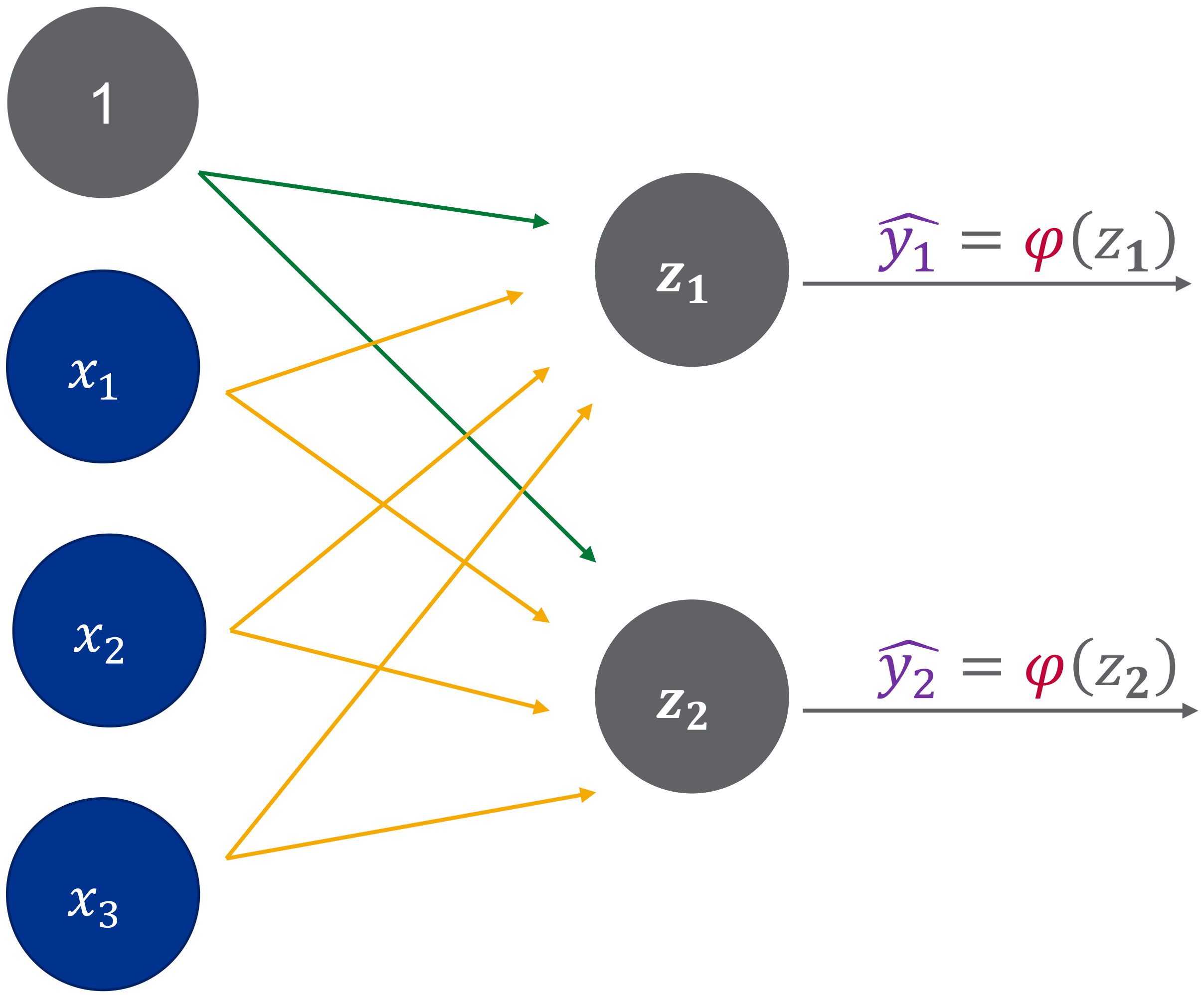
$$\boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}, \boldsymbol{w}_1 = \begin{bmatrix} w_{01} \\ w_{11} \\ w_{21} \\ w_{31} \end{bmatrix}, z_1 = \boldsymbol{w}_1^T \boldsymbol{x}$$

Function Approximation: Multiple Neurons



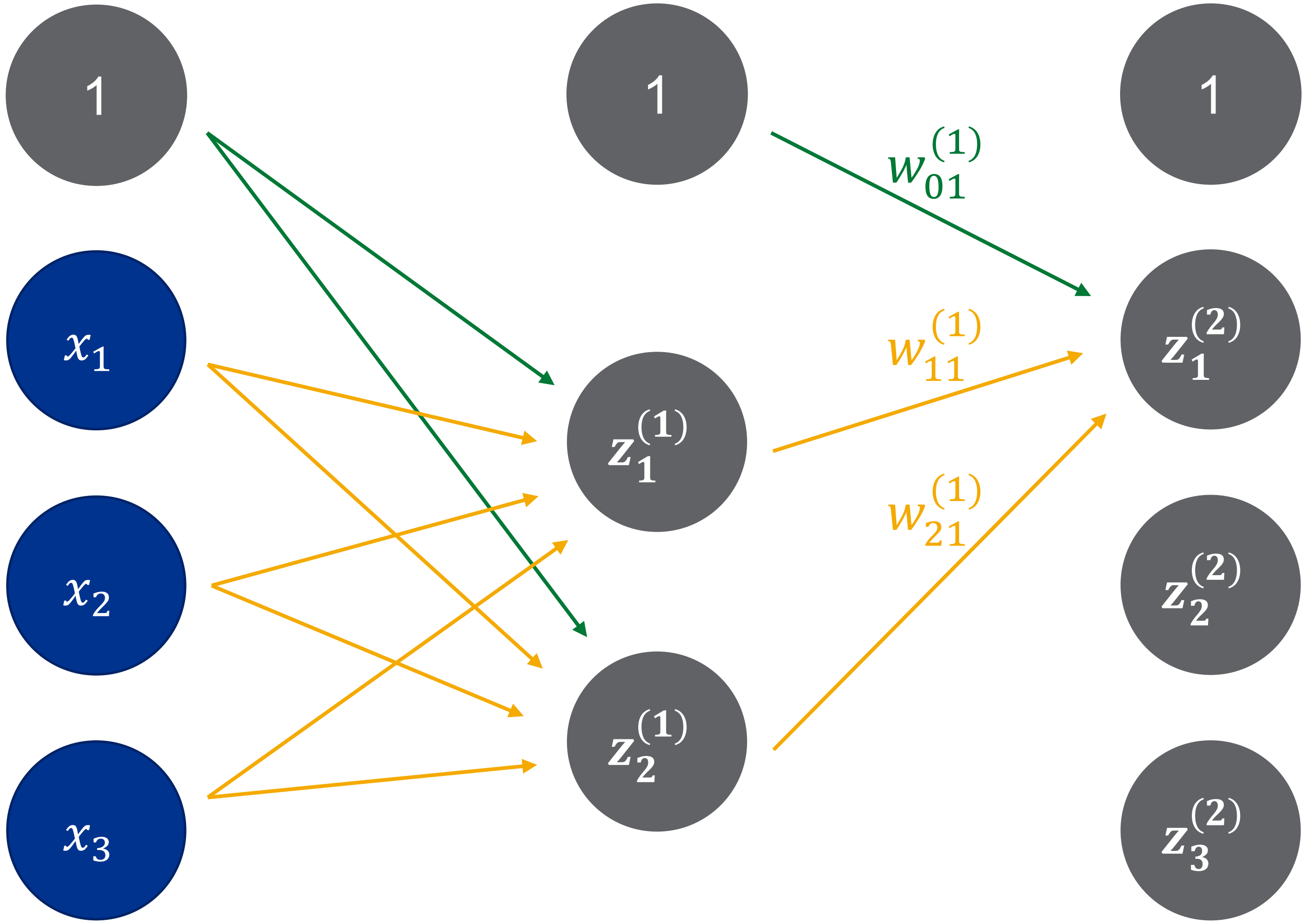
$$\boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}, \; \boldsymbol{w}_2 = \begin{bmatrix} w_{02} \\ w_{12} \\ w_{22} \\ w_{32} \end{bmatrix}, \; z_2 = \boldsymbol{w}_2^T \boldsymbol{x}$$

Function Approximation: Multiple Neurons



$$\boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \boldsymbol{w}_i = \begin{bmatrix} w_{0i} \\ w_{1i} \\ w_{2i} \\ w_{3i} \end{bmatrix}, \quad z_i = \boldsymbol{w}_i^T \boldsymbol{x}$$

Function Approximation: Multiple Layers

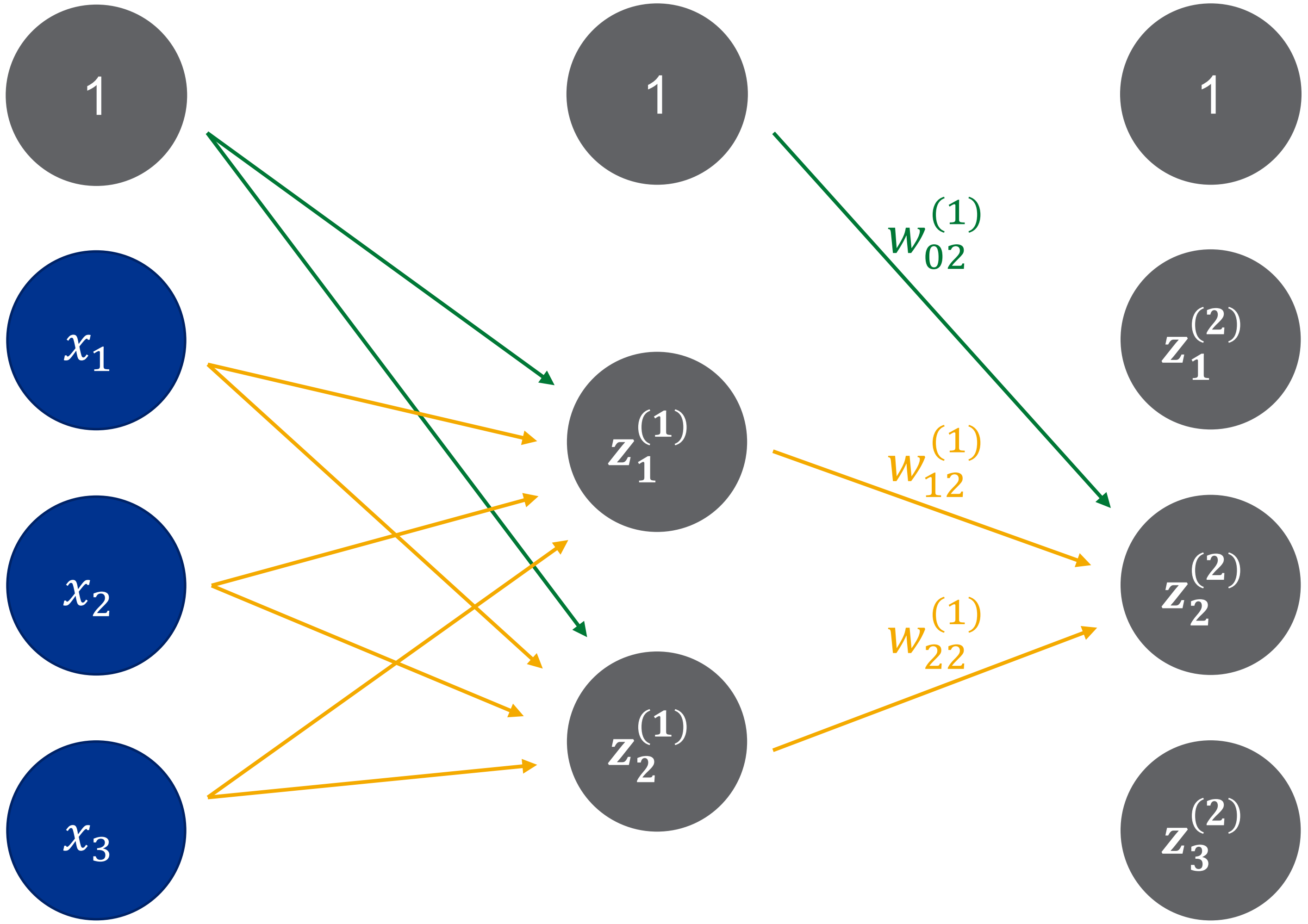


$$z_1^{(1)} = w_1^{(1)T} x, \quad w_1^{(1)} = \begin{bmatrix} w_{01}^{(1)} \\ w_{11}^{(1)} \\ w_{21}^{(1)} \end{bmatrix}$$

$$\widehat{y_1} = \varphi\left(w_1^{(1)T} \varphi\left(z_1^{(1)}\right)\right)$$

Function Approximation: Multiple Layers

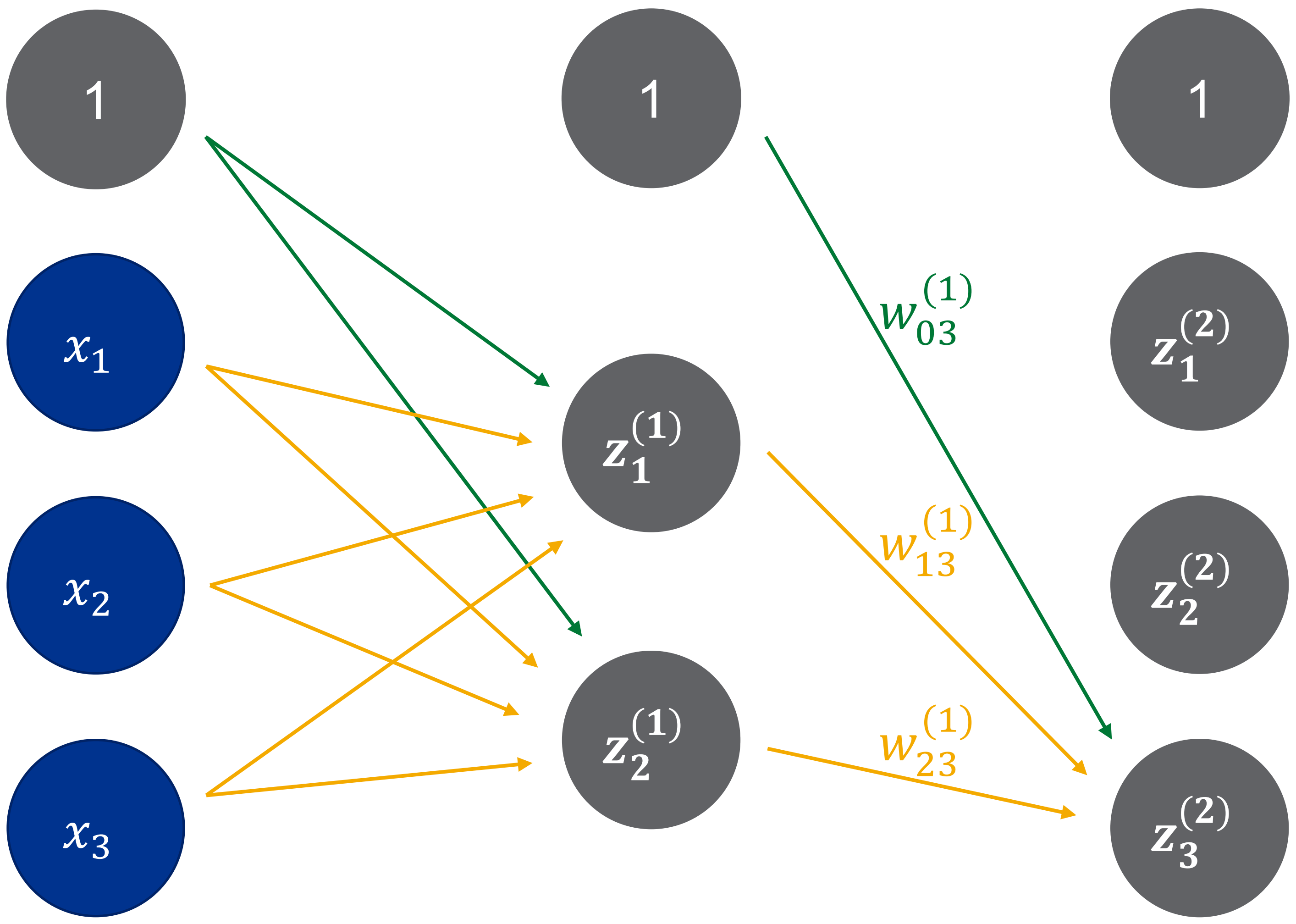
$$z_2^{(1)} = w_2^{(1)T} x, \quad w_2^{(1)} = \begin{bmatrix} w_{02}^{(1)} \\ w_{12}^{(1)} \\ w_{22}^{(1)} \end{bmatrix}$$



$$\widehat{y_2} = \varphi\left(w_2^{(1)T} \varphi\left(z_2^{(1)}\right)\right)$$

Function Approximation: Multiple Layers

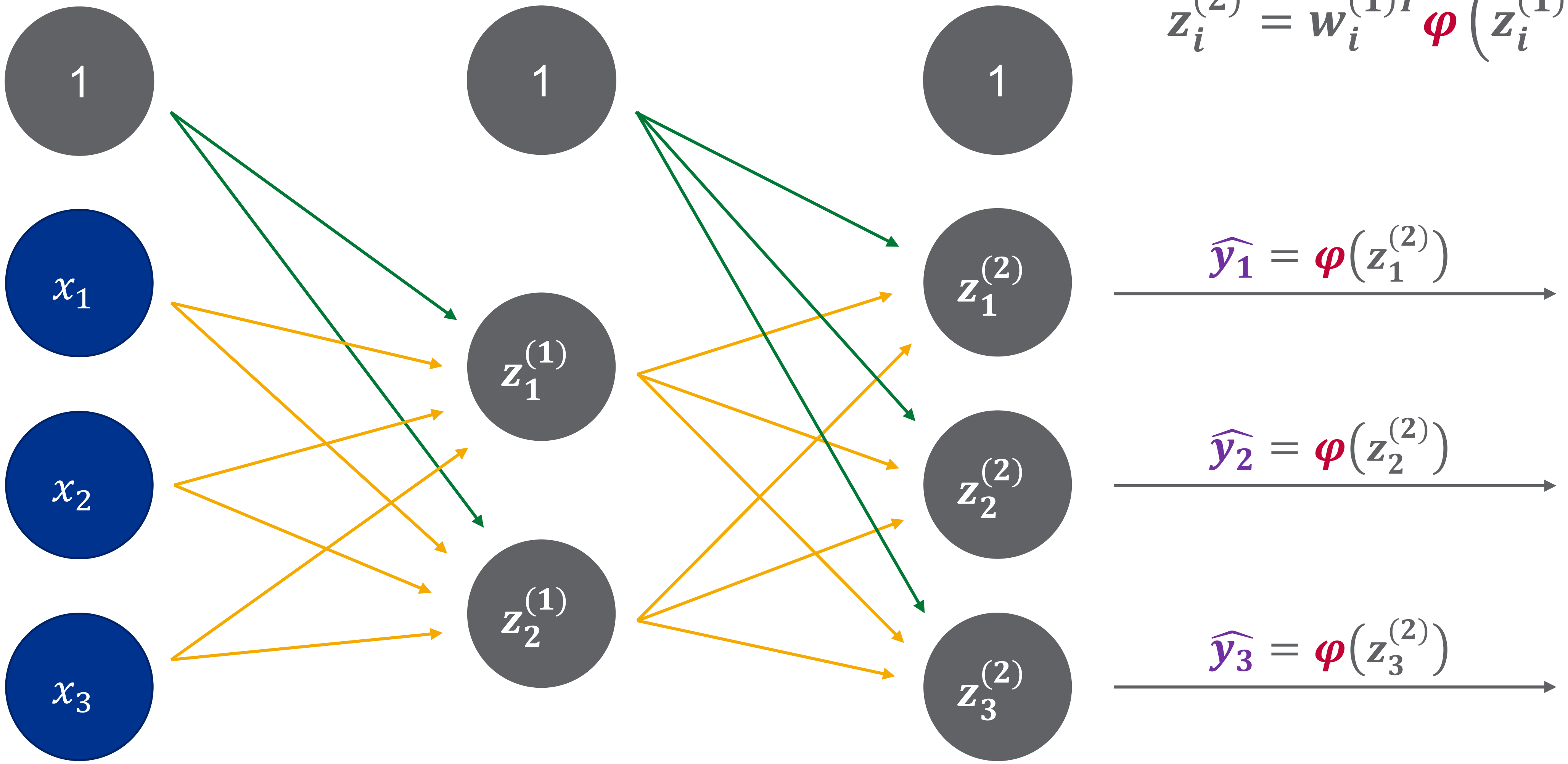
$$z_3^{(1)} = w_3^{(1)T} x, \quad w_3^{(1)} = \begin{bmatrix} w_{03}^{(1)} \\ w_{13}^{(1)} \\ w_{23}^{(1)} \end{bmatrix}$$



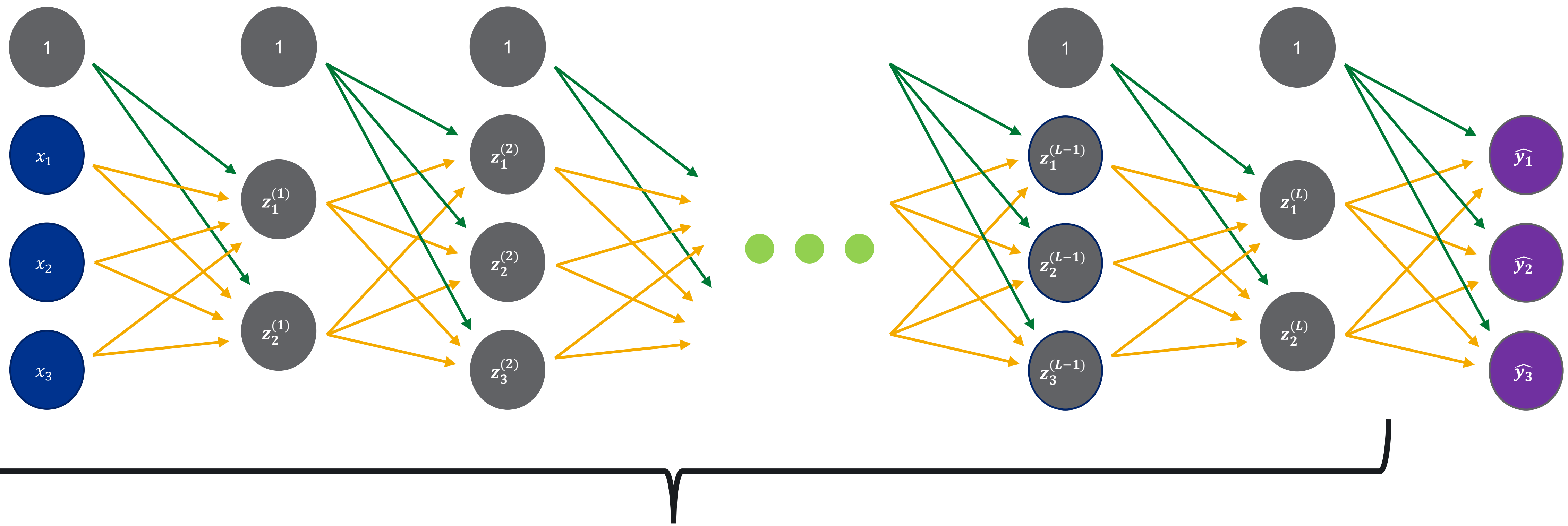
$$\widehat{y_3} = \varphi\left(w_3^{(1)T} \varphi\left(z_3^{(1)}\right)\right)$$

Function Approximation: Multiple Layers

$$z_i^{(2)} = w_i^{(1)T} \varphi\left(z_i^{(1)}\right), \quad w_i^{(1)} = \begin{bmatrix} w_{0i}^{(1)} \\ w_{1i}^{(1)} \\ w_{2i}^{(1)} \end{bmatrix}$$



Function Approximation: Multiple Layers



The final output can be defined recursively using the composition of all previous layers,

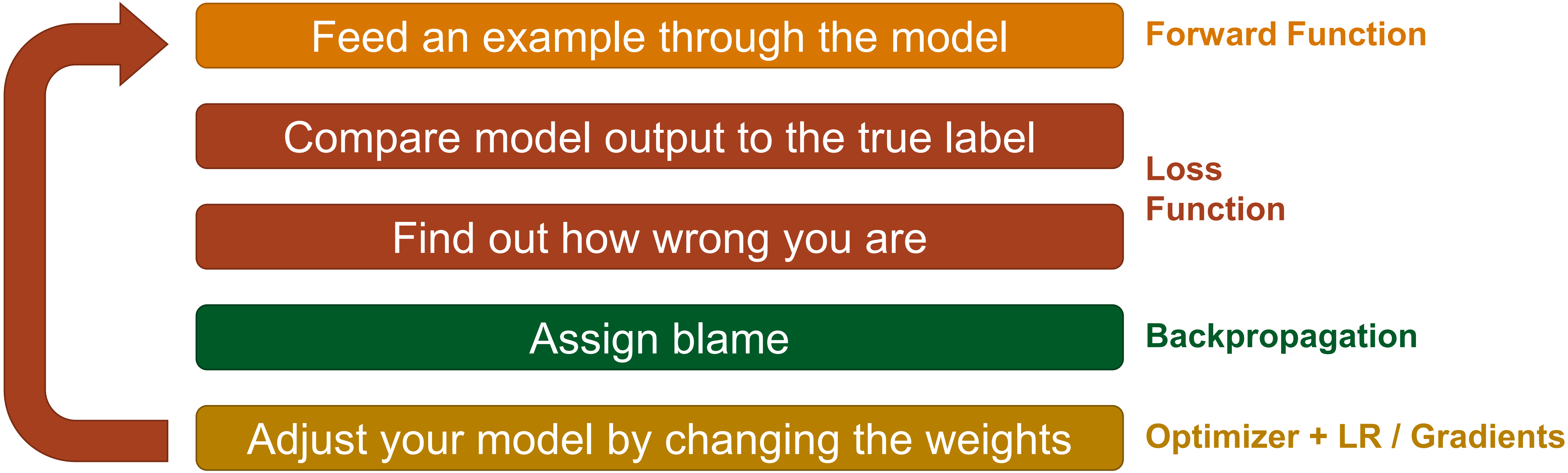
$\hat{y}_i = \boldsymbol{\varphi}\left(\boldsymbol{w}_i^{(L)T}\boldsymbol{\varphi}\left(\boldsymbol{z}_i^{(L)}\right)\right)$ where $\boldsymbol{z}_i^{(j)} = \boldsymbol{w}_i^{(j-1)T}\boldsymbol{\varphi}\left(\boldsymbol{z}_i^{(j-1)}\right)$ for $j = 2, \ldots L$.

We can write the whole network as the function, $\hat{\boldsymbol{y}}^{(\boldsymbol{n})} = \boldsymbol{f}\left(\boldsymbol{x}^{(n)}; \boldsymbol{W}\right)$, where $\boldsymbol{W}$ represents all the weights

DNNs are series of learned functions

- Given an input the model makes a prediction (e.g., text → class)
- Different types of intermediate functions are like building blocks we can customize to fit a problem
- We choose the architecture of the function, but the functions themselves are learned from training data
- **The goal is for the function to correctly predict the output for an example it has never seen before**

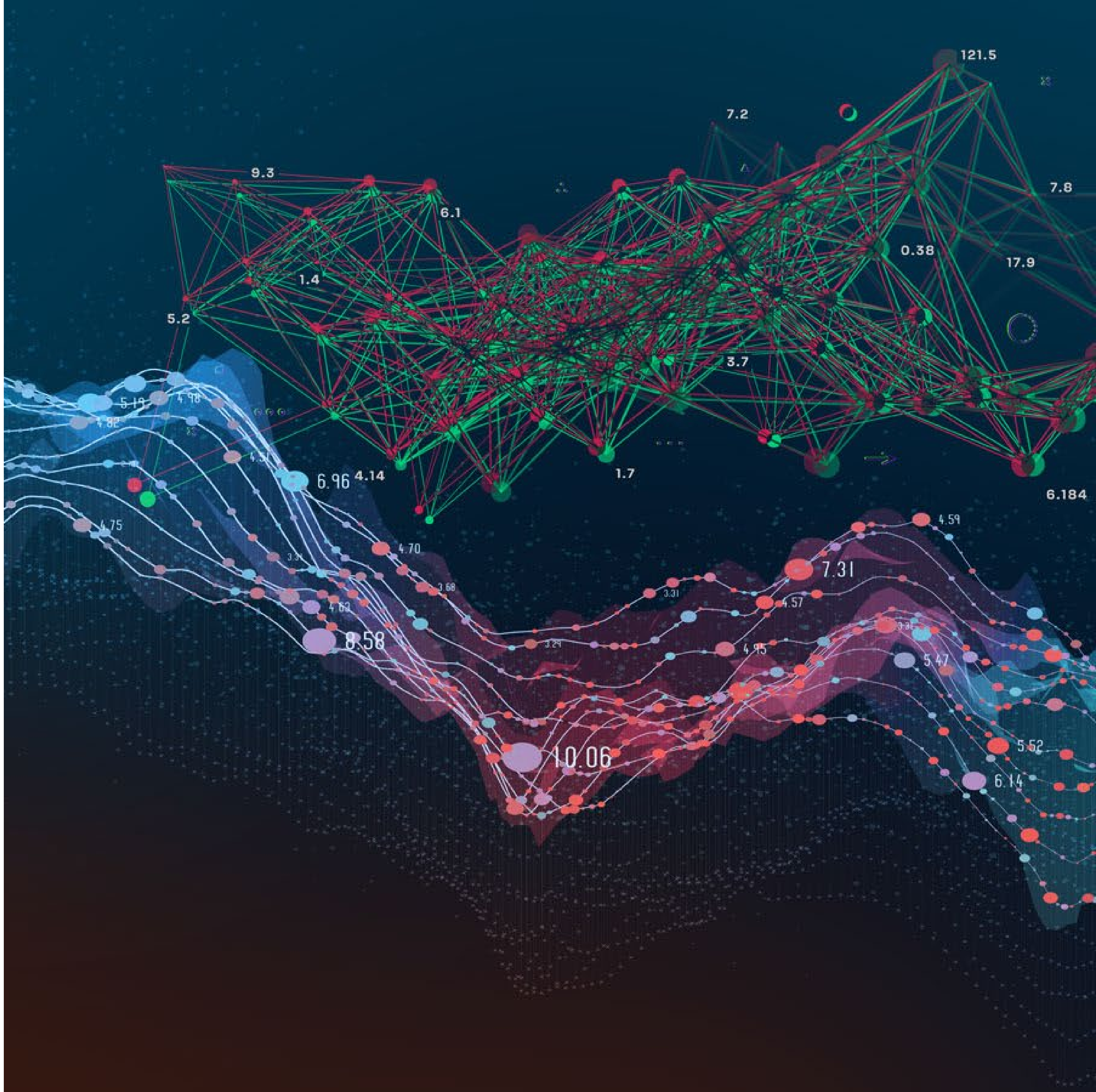
Neural Network Training



- Once we've completed these steps for every example in the training data, we've completed one ***epoch*** of training

Pacific Northwest
NATIONAL LABORATORY

Transformer Appendix



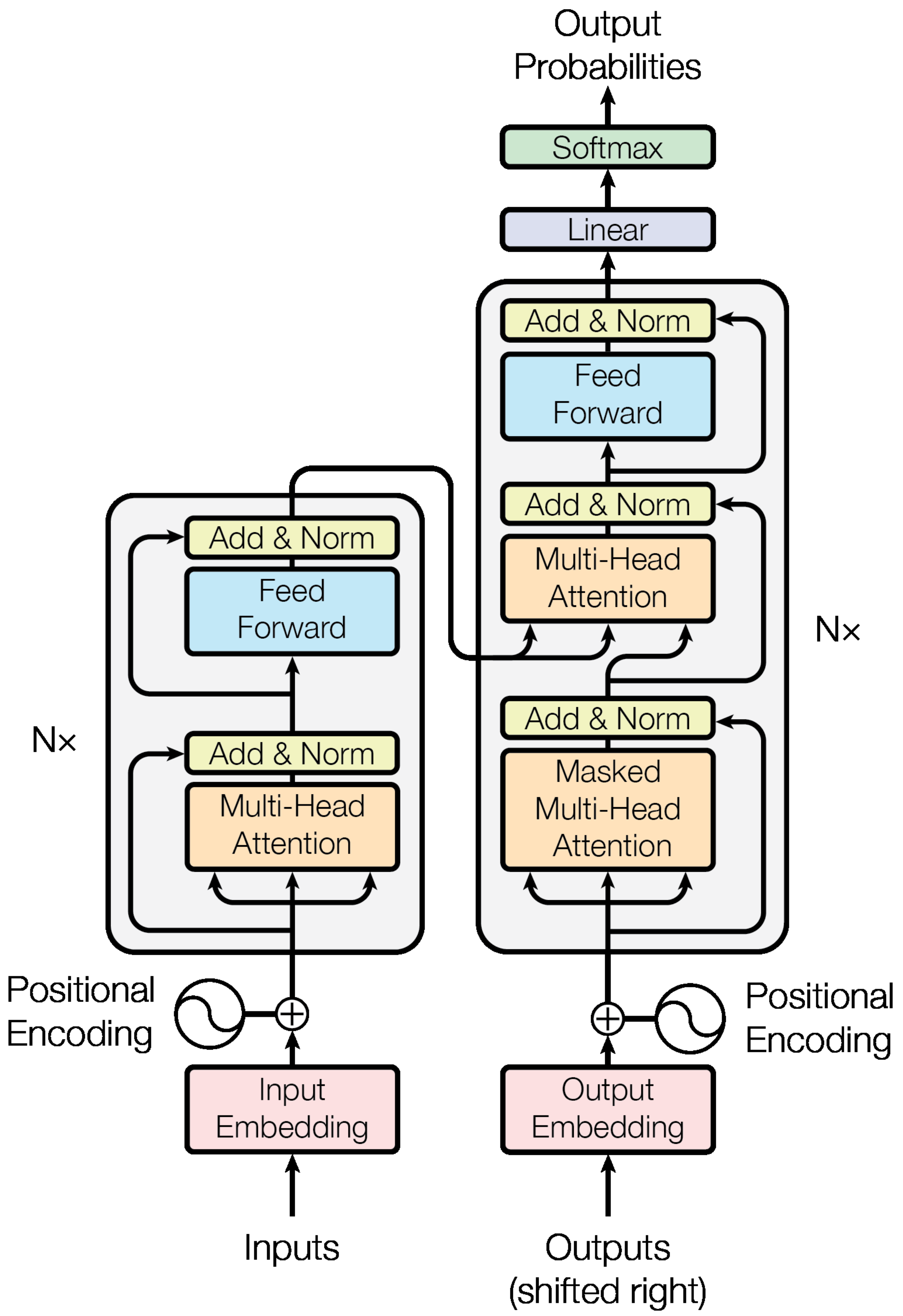
U.S. DEPARTMENT OF ENERGY BATTELLE

PNNL is operated by Battelle for the U.S. Department of Energy

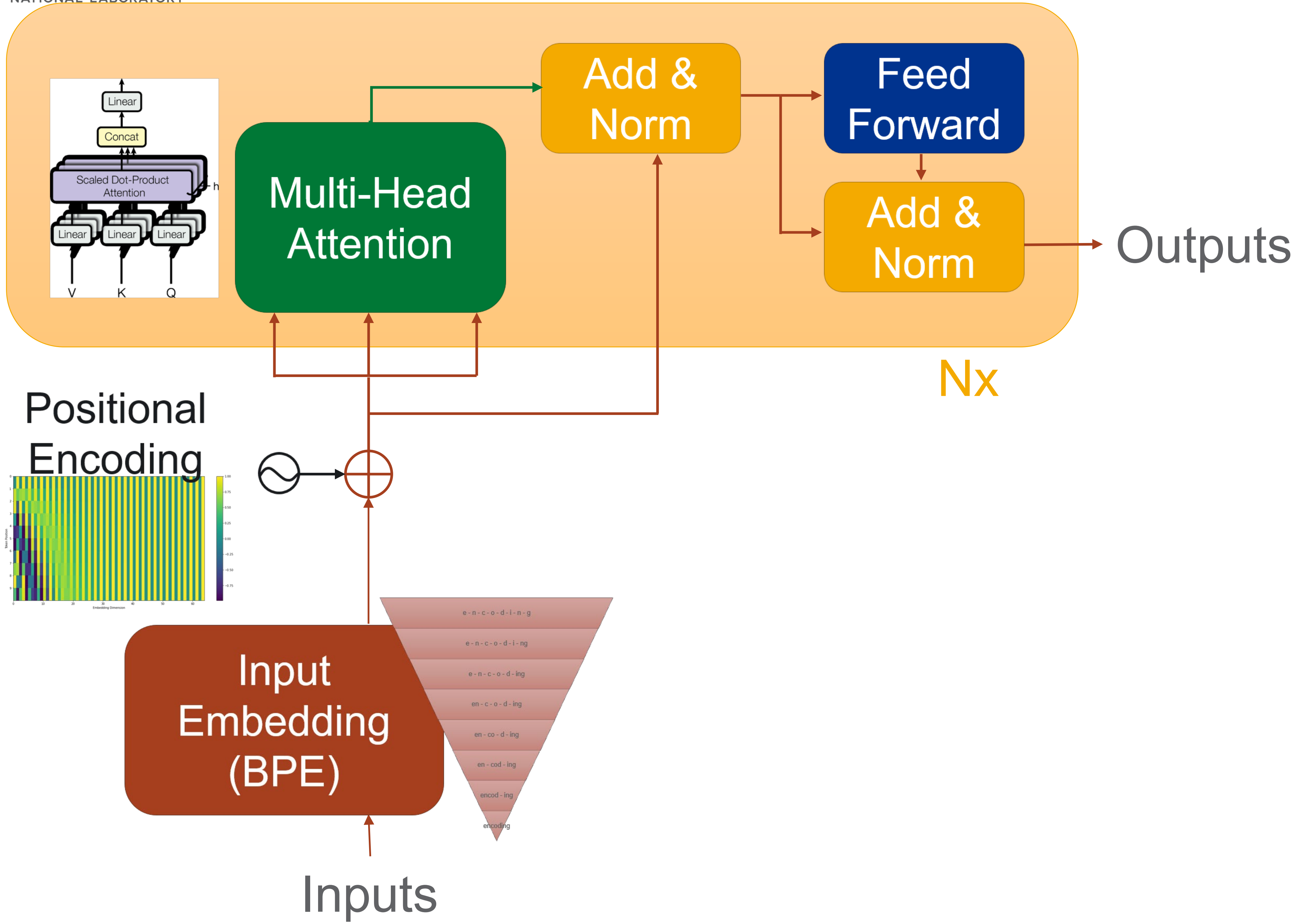
UNCLASSIFIED

Encoder & Decoder

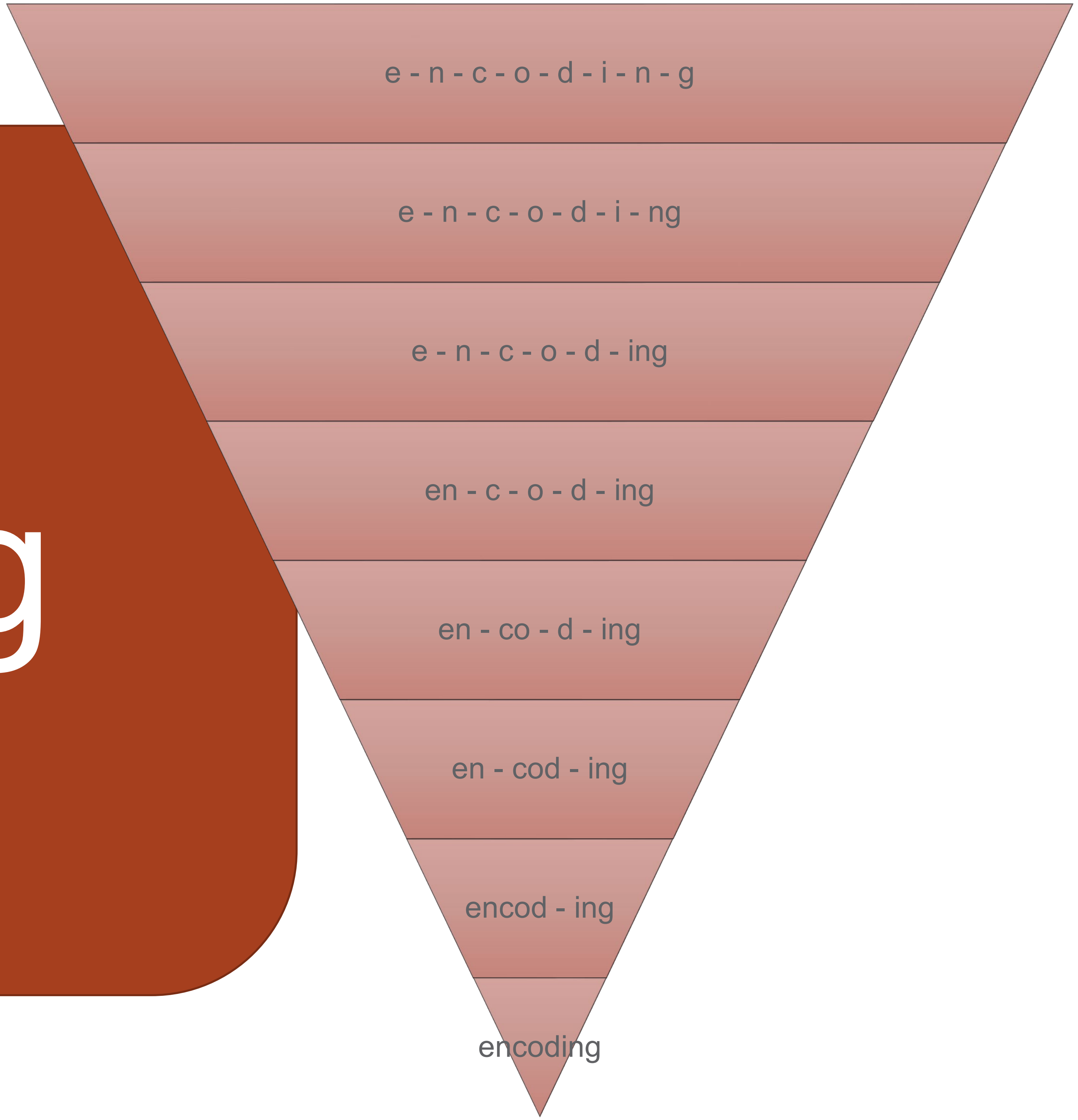
- Designed for translation, nowadays most use one or the other
- In the original paper,
  - $N = 6$
- Decoder uses *masking*, any position cannot be "seen" by later positions



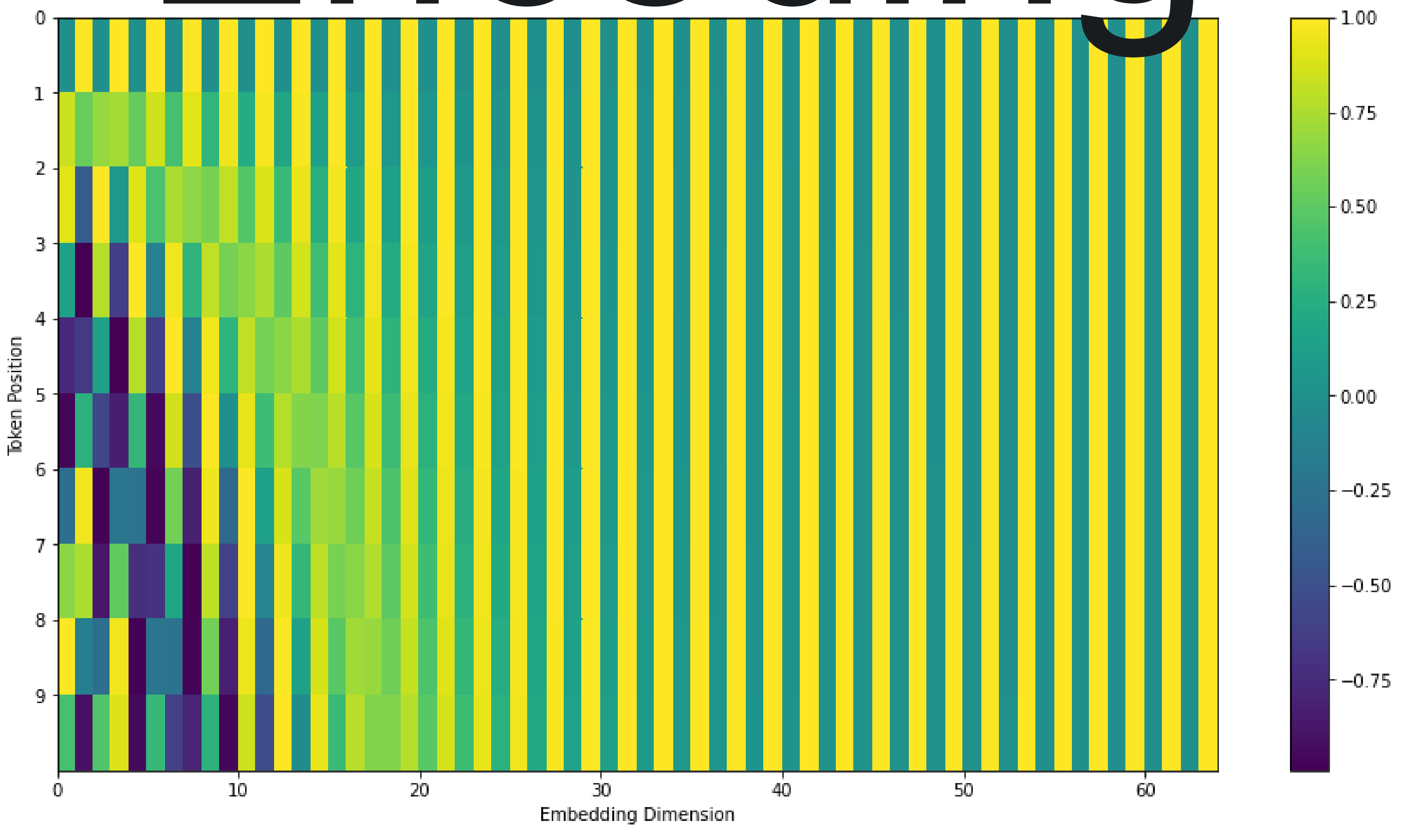
Encoder Architecture

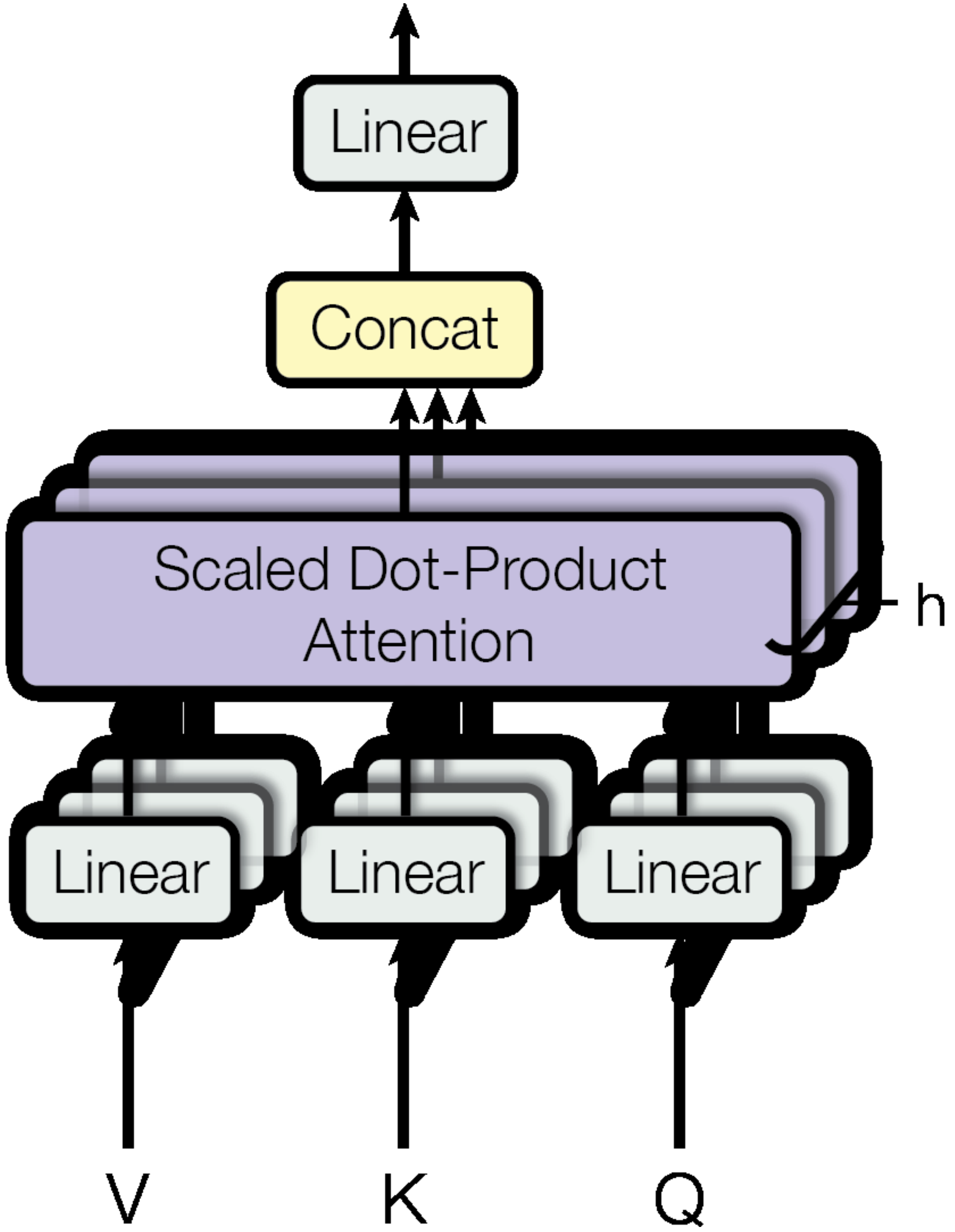


Input Embedding (BPE)



Positional Encoding





Multi-Head Attention

RETURN

Decoder Architecture