Foundation Models & Embeddings

Lara J. Martin (she/they)

https://laramartin.net/interactive-fiction-class

Slides modified from Dr. Frank Ferraro

Learning Objectives

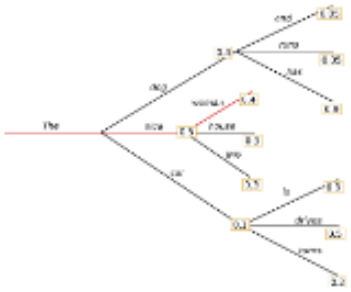
Recognize useful encoder-only, encoder-decoder, and decoder-only models

Understand the use & creation of dense vector embeddings

Calculate the distance between vector embeddings

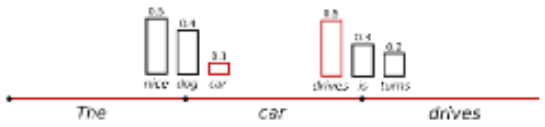
Differentiate between encoder model embeddings and older dense embeddings

Review: Sampling

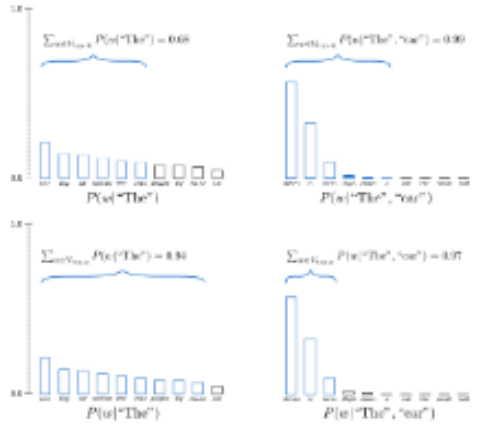


Greedy

Beam Search



Random Sampling



Top-K/P

Review

What’s the difference between finetuning and prompting?

What’s the difference between zero-shot and few-shot prompting?

Review: Tricks of the Trade

Instruction-tuned models like GPT-3.5 and Mistral-7B-Instruct like to be given a “role” first (e.g., “You are a helpful writing assistant.”)

The more defined the task, the better

- More details
- One thing to do at a time

LLMs are overly confident (like people on the internet)

- To “objectively” have the model evaluate something, you should have another instance judge

Chain-of-thought prompting helps models come up with better answers

They will “Yes and…” your prompt

LLM Vocabulary

Finetuning: Training an LLM more to adapt it to your task

Pre-training: The training before finetuning; creating a foundation model

Prompting: Getting the output you want by just changing the input; the model doesn't change

Zero-shot prompting: Prompting without examples

Few-shot prompting: Prompting with a few examples

Prompt engineering: Figuring out the right prompt for a task

Prompt tuning: Automated prompt engineering

Review:
What might go wrong with finetuning?

Underfitting – finetuning data is too different from what the foundational model was train on

Overfitting – overwrites what the model learned originally

Foundation Models

Types of Foundation Models

Encoder Only

Decoder Only

Encoder-Decoder Models

Denotes what they use during pre-training

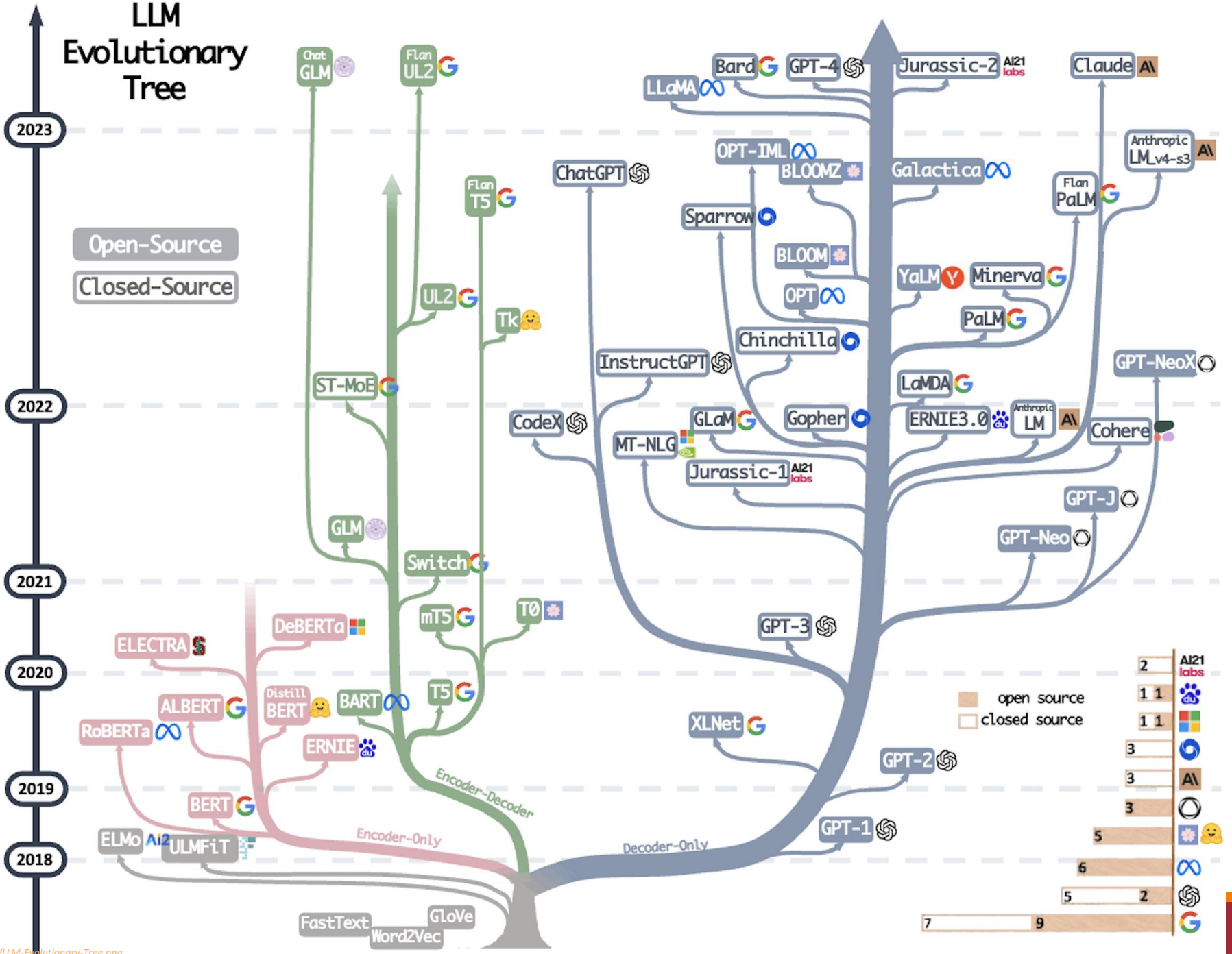
What is a foundation model?

A model that captures “foundational” or core information about a modality (e.g., text, speech, images)

Pretrained on a large amount of data & able to be finetuned on a particular task

Self-supervised

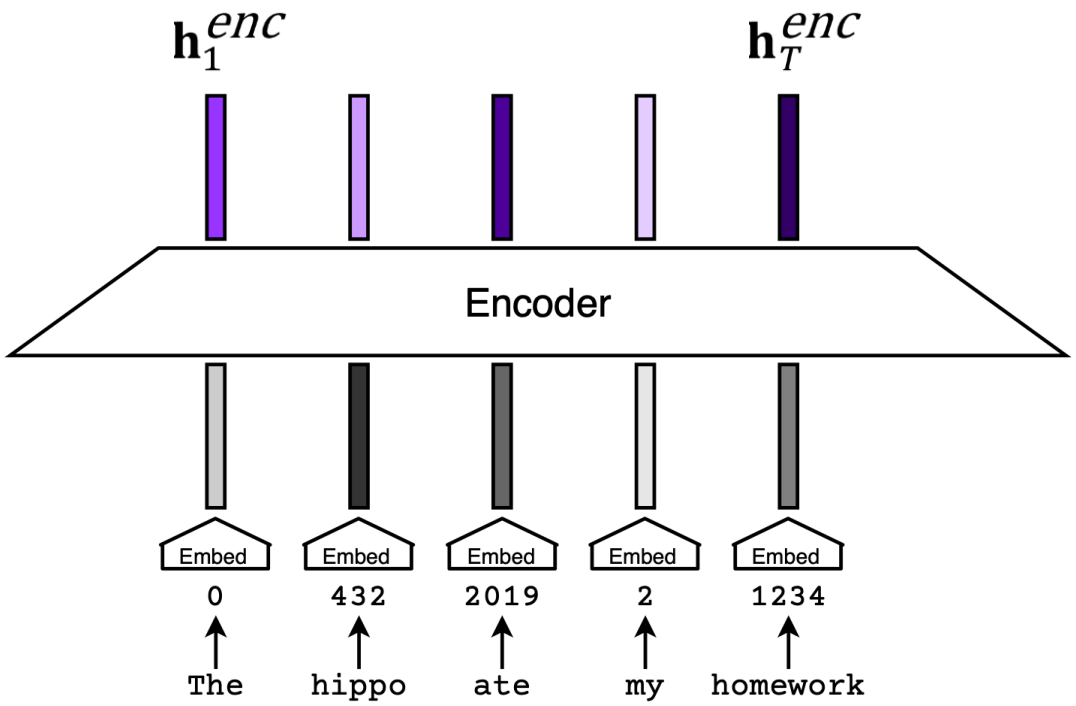
All non-finetuned large language models (LLMs) are foundational models

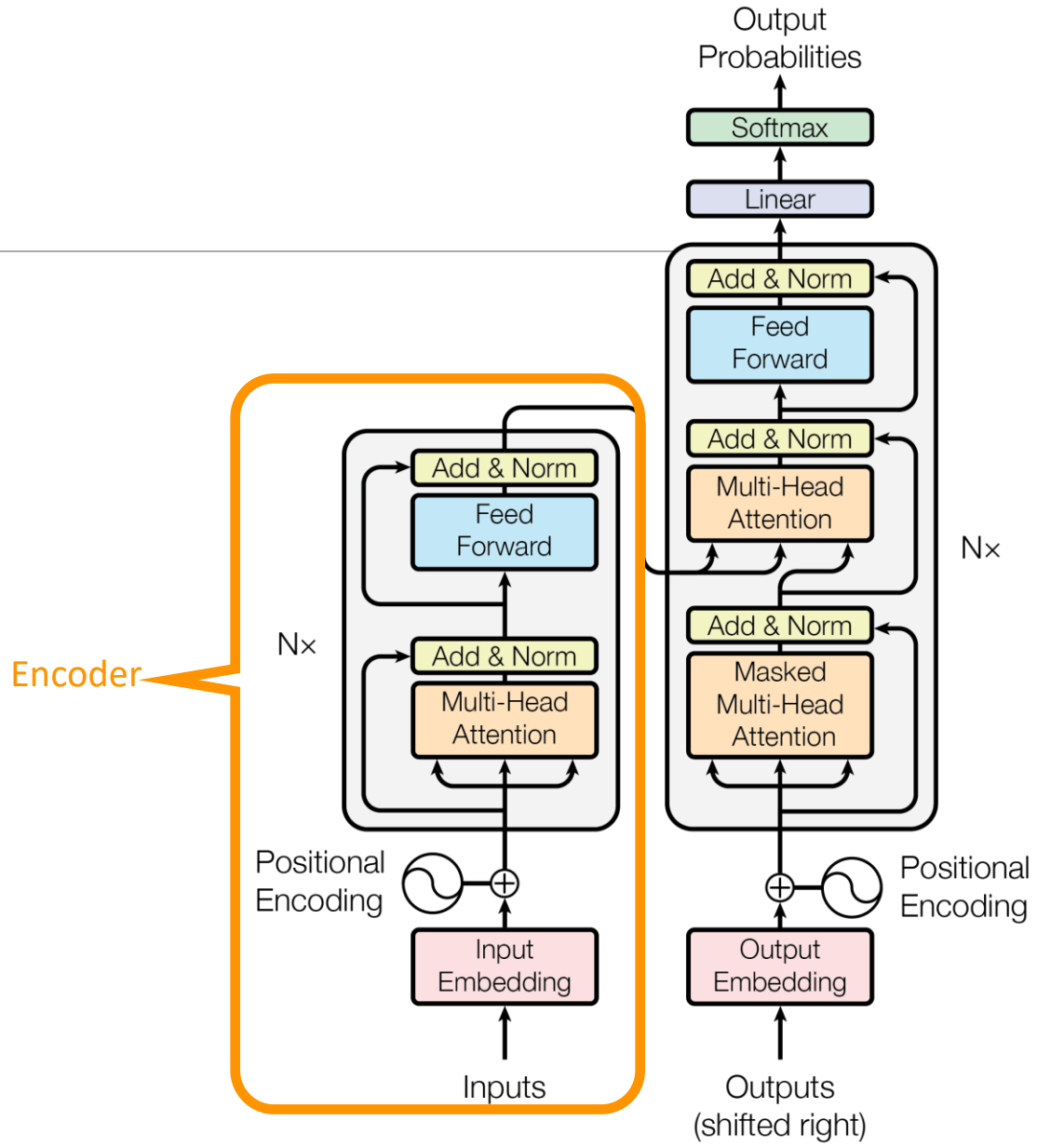


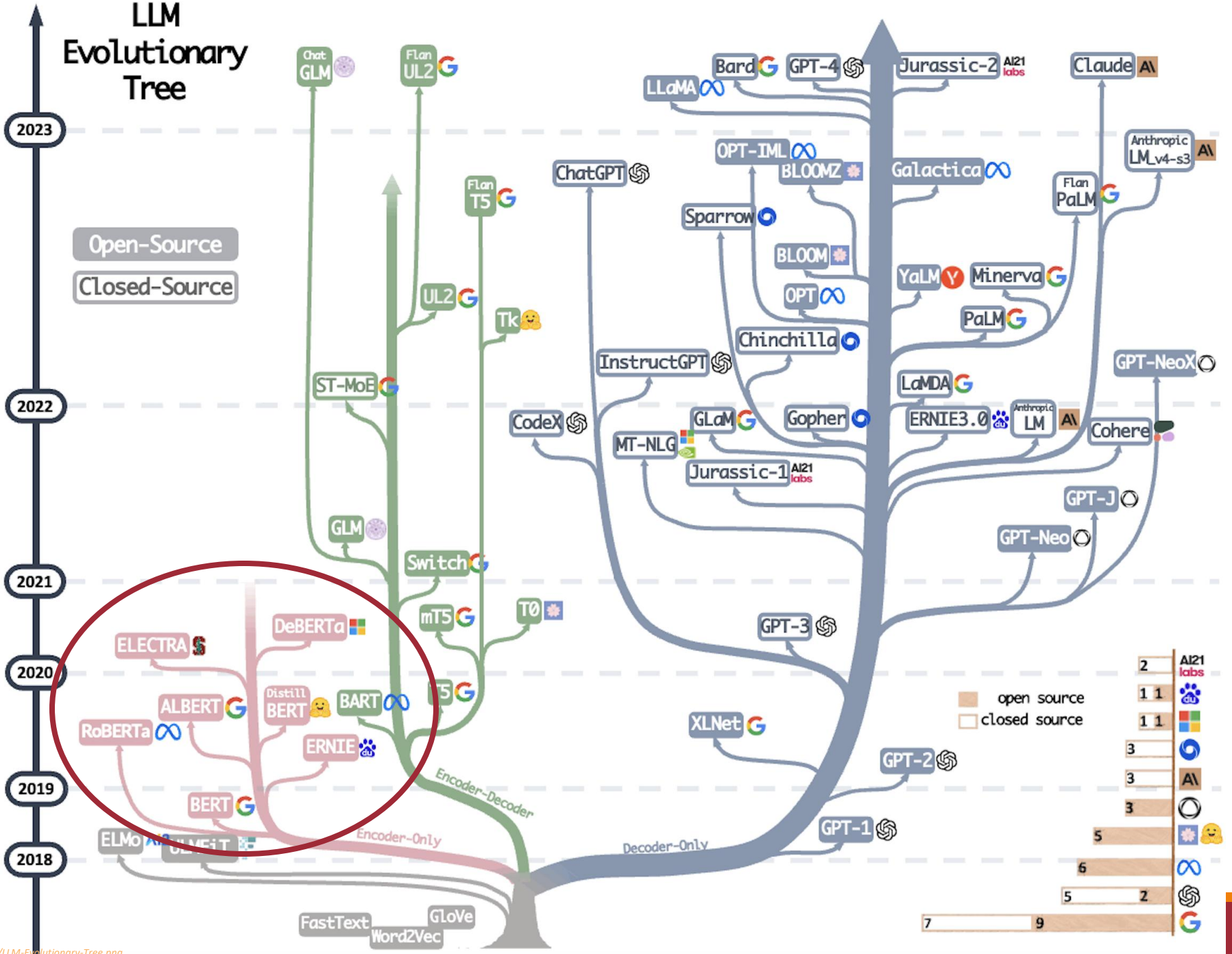
https://blog.biocomm.ai/wp-content/uploads/2023/05/LLM-Evolutionary-Tree.png


123
SESAME STREET

Encoder-only models

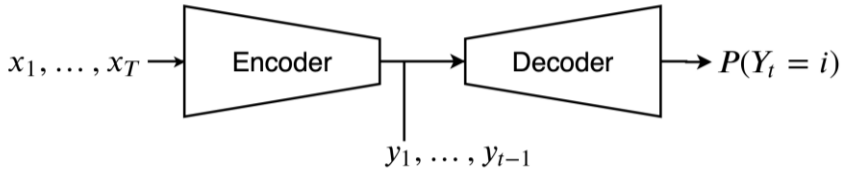






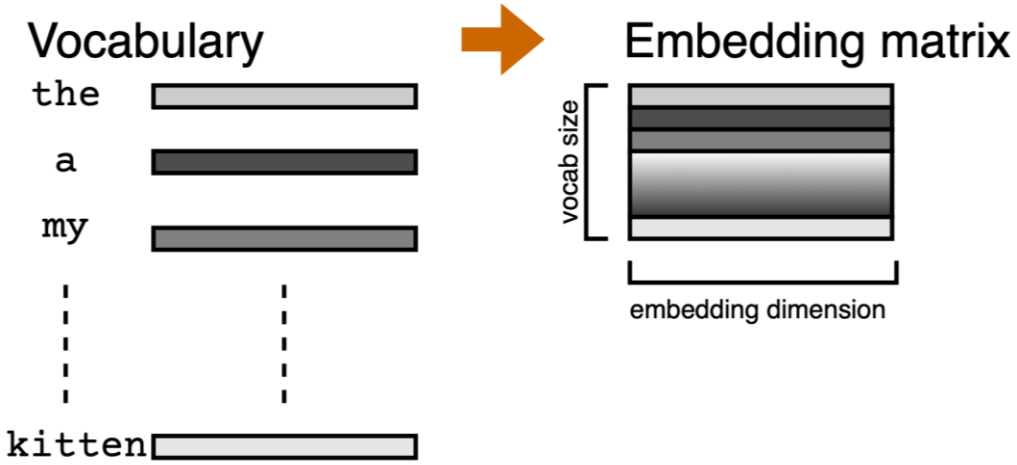
https://blog.biocomm.ai/wp-content/uploads/2023/05/LLM-Evolutionary-Tree.png

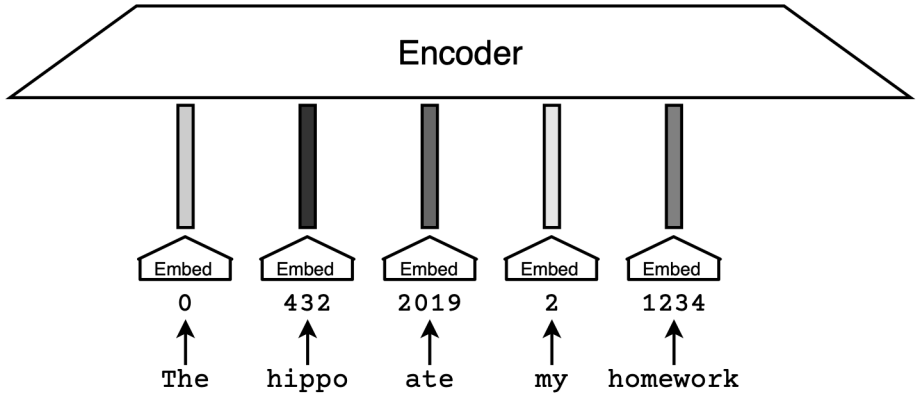
Word Embeddings



Review: Inputs to the Encoder

The encoder takes as input the embeddings corresponding to each token in the sequence.





How have we represented words?

Each word is a distinct item

- Bijection between the strings and unique integer ids:
- "cat" --> 3, "kitten" --> 792 "dog" --> 17394

- Are "cat" and "kitten" similar?

Equivalently: "One-hot" encoding

- Represent each word type w with a vector the size of the vocabulary
- This vector has V-1 zero entries, and 1 non-zero (one) entry

One-Hot Encoding Example

Let our vocab be {a, cat, saw, mouse, happy}

V = # types = 5

Assign:

a	4
cat	2
saw	3
mouse	0
happy	1

How do we represent "cat?"

$$e_{\text{cat}} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

How do we represent "happy?"

$$e_{\text{happy}} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

The Fragility of One-Hot Encodings
Case Study: Plagiarism Detector

Given two documents x_1, x_2, predict y = 1 (plagiarized) or y = 0 (not plagiarized)

What is/are the:

- Method/steps for predicting?
- General formulation?
- Features?

Image: http://3.bp.blogspot.com/_MWJDzvrnu7Y/TTcXpCWZerI/AAAAAAAAWY/zxu1GwrIBz0/s1600/excellent-mr-burns.gif

Case Study: Plagiarism Detector (Feature Example)

Given two documents x_1, x_2, predict y = 1 (plagiarized) or y = 0 (not plagiarized)

Intuition: documents are more likely to be plagiarized if they have words in common

$$f_{\text{any-common-word,Plag.}}(x_1, x_2) = ???$$
$$f_{\text{<word v>,Plag.}}(x_1, x_2) = ???$$

Image: http://3.bp.blogspot.com/_MWJDzvrnu7Y/TTcXpCWZerI/AAAAAAAAWY/zxu1GwrIBz0/s1600/excellent-mr-burns.gif

Case Study: Plagiarism Detector (Feature Example)

Given two documents x_1, x_2, predict y = 1 (plagiarized) or y = 0 (not plagiarized)

Intuition: documents are more likely to be plagiarized if they have words in common

n # adjacent words

$$f_{\text{any-common-word,Plag.}}(x_1, x_2) = ???$$

$$f_{\text{<word v>,Plag.}}(x_1, x_2) = ???$$

$$f_{\text{<ngram Z>,Plag.}}(x_1, x_2) = ???$$

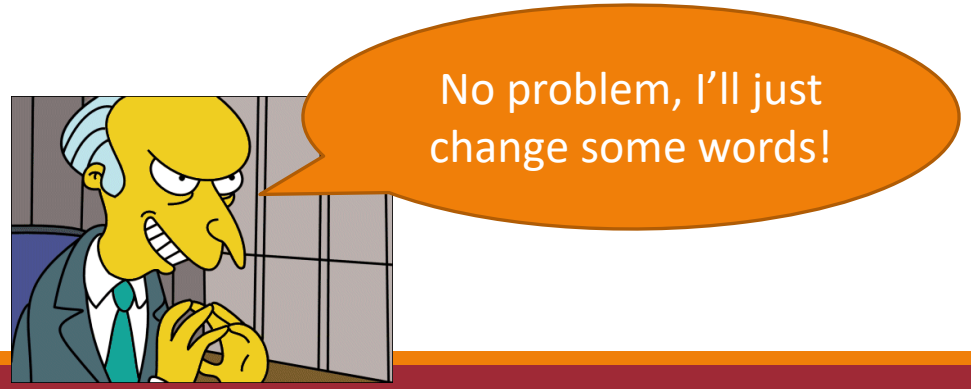


image: http://3.bp.blogspot.com/_MWJDzvrnu7Y/TTcXpCWZerI/AAAAAAAAWY/zxu1GwrIBz0/s1600/excellent-mr-burns.gif

Case Study: Plagiarism Detector (Feature Example)

Given two documents x_1, x_2, predict y = 1 (plagiarized) or y = 0 (not plagiarized)

Intuition: documents are more likely to be plagiarized if they have words in common

$$f_{\text{any-common-word,Plag.}}(x_1, x_2) = ???$$
$$f_{\text{<word v>,Plag.}}(x_1, x_2) = ???$$
$$f_{\text{<ngram Z>,Plag.}}(x_1, x_2) = ???$$
$$f_{\text{synonym-of-<word v>,Plag.}}(x_1, x_2) = ???$$

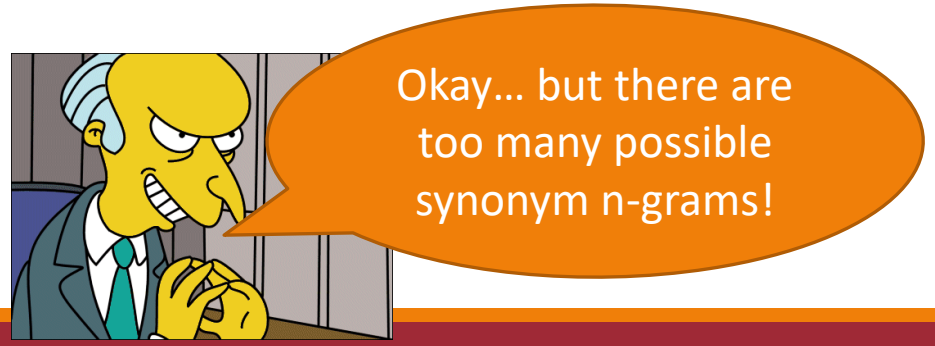


Image: http://3.bp.blogspot.com/_MWJDzvrnu7Y/TTcXpCWZerI/AAAAAAAAWY/zxu1GwrIBz0/s1600/excellent-mr-burns.gif

Case Study: Plagiarism Detector (Feature Example)

Given two documents x_1, x_2, predict y = 1 (plagiarized) or y = 0 (not plagiarized)

Intuition: documents are more likely to be plagiarized if they have words in common

$$f_{\text{any-common-word,Plag.}}(x_1, x_2) = ???$$
$$f_{\text{<word v>,Plag.}}(x_1, x_2) = ???$$
$$f_{\text{<ngram Z>,Plag.}}(x_1, x_2) = ???$$
$$f_{\text{synonym-of-<word v>,Plag.}}(x_1, x_2) = ???$$
$$f_{\text{synonym-of-<ngram Z>,Plag.}}(x_1, x_2) = ???$$

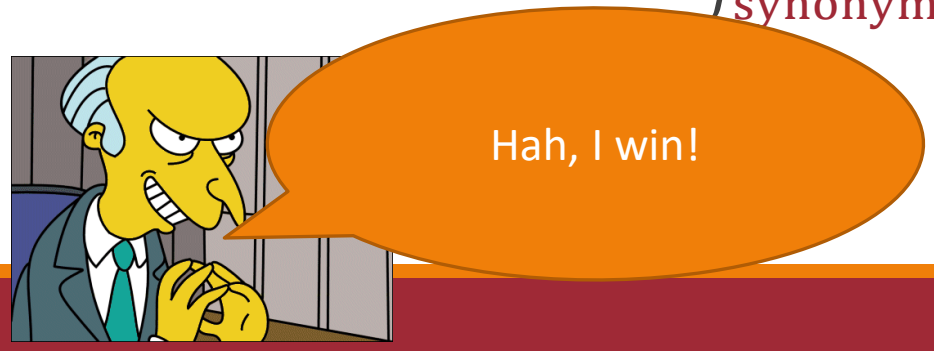


image: http://3.bp.blogspot.com/_MWJDzvrnu7Y/TTcXpCWZerI/AAAAAAAAWY/zxu1GwrIBz0/s1600/excellent-mr-burns.gif

Plagiarism Detection: Word Similarity?

MAINFRAMES

Mainframes are primarily referred to large computers with rapid, advanced processing capabilities that can execute and perform tasks equivalent to many Personal Computers (PCs) machines networked together. It is characterized with high quantity Random Access Memory (RAM), very large secondary storage devices, and high-speed processors to cater for the needs of the computers under its service.

Consisting of advanced components, mainframes have the capability of running multiple large applications required by many and most enterprises and organizations. This is one of its advantages. Mainframes are also suitable to cater for those applications (programs) or files that are of very high demand by its users (clients). Examples of such organizations and enterprises using mainframes are online shopping websites such as

MAINFRAMES

Mainframes usually are referred those computers with fast, advanced processing capabilities that could perform by itself tasks that may require a lot of Personal Computers (PC) Machines. Usually mainframes would have lots of RAMs, very large secondary storage devices, and very fast processors to cater for the needs of those computers under its service.

Due to the advanced components mainframes have, these computers have the capability of running multiple large applications required by most enterprises, which is one of its advantage. Mainframes are also suitable to cater for those applications or files that are of very large demand by its users (clients). Examples of these include the large online shopping websites -i.e. : Ebay, Amazon, Microsoft, etc.

Word Embeddings

A dense, “low”-dimensional vector representation

Many values are not 0 (or at least less sparse than one-hot)

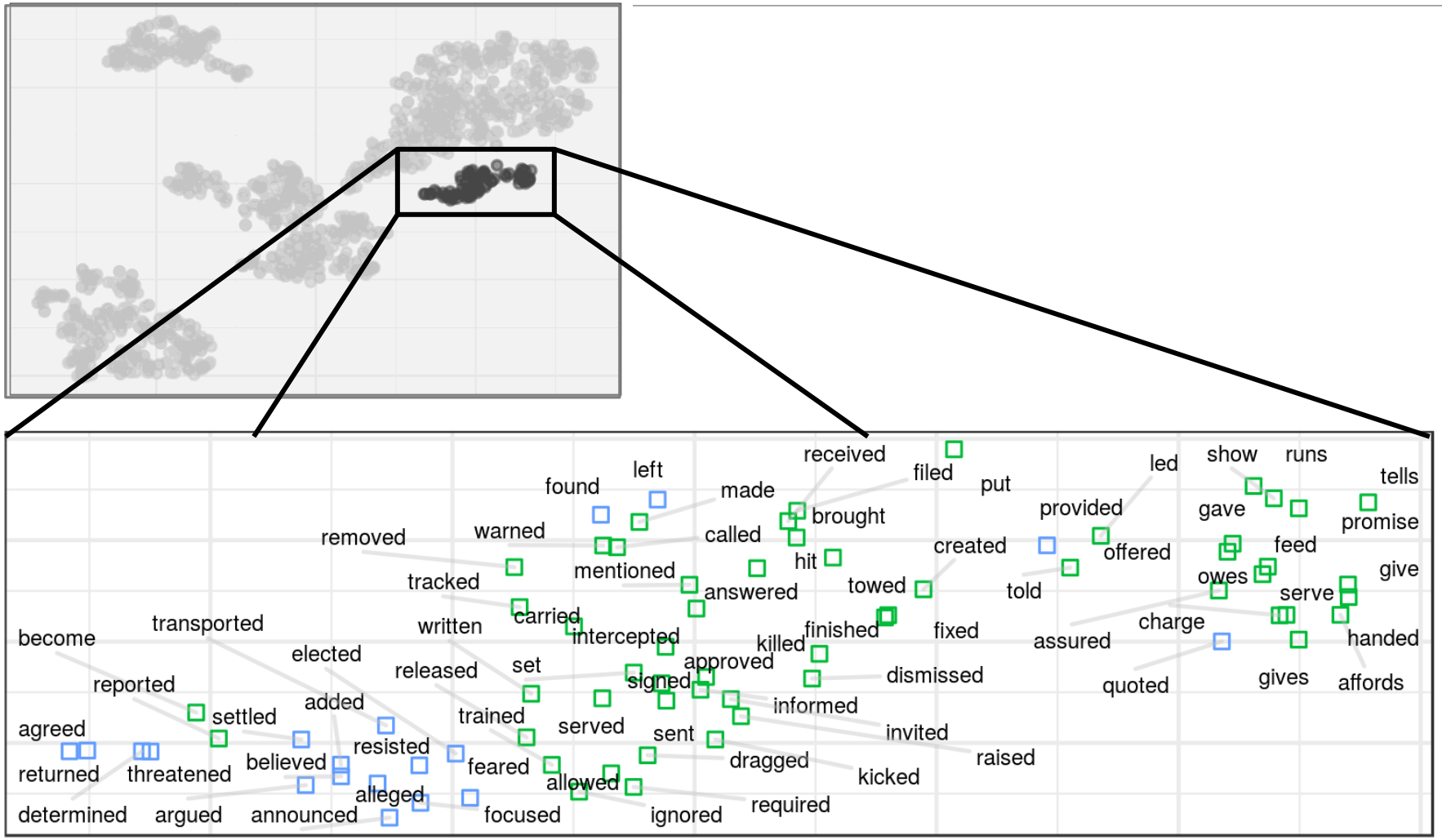
Up till ~2013: E could be any size
2013-present: E << vocab

An E-dimensional vector, often (but not always) real-valued

These are also called

- **embeddings**
- **Continuous representations**
- **(word/sentence/…) vectors**
- **Vector-space models**

A Dense Representation

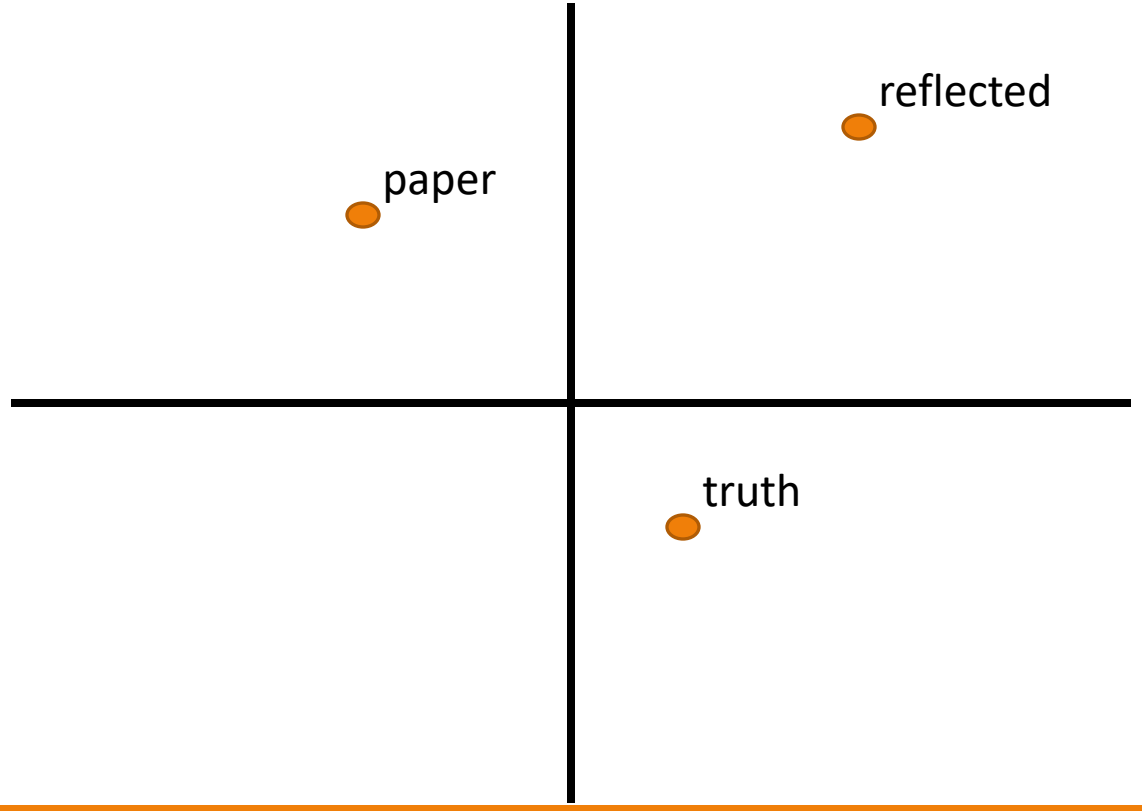


Continuous Meaning

The paper reflected the truth.

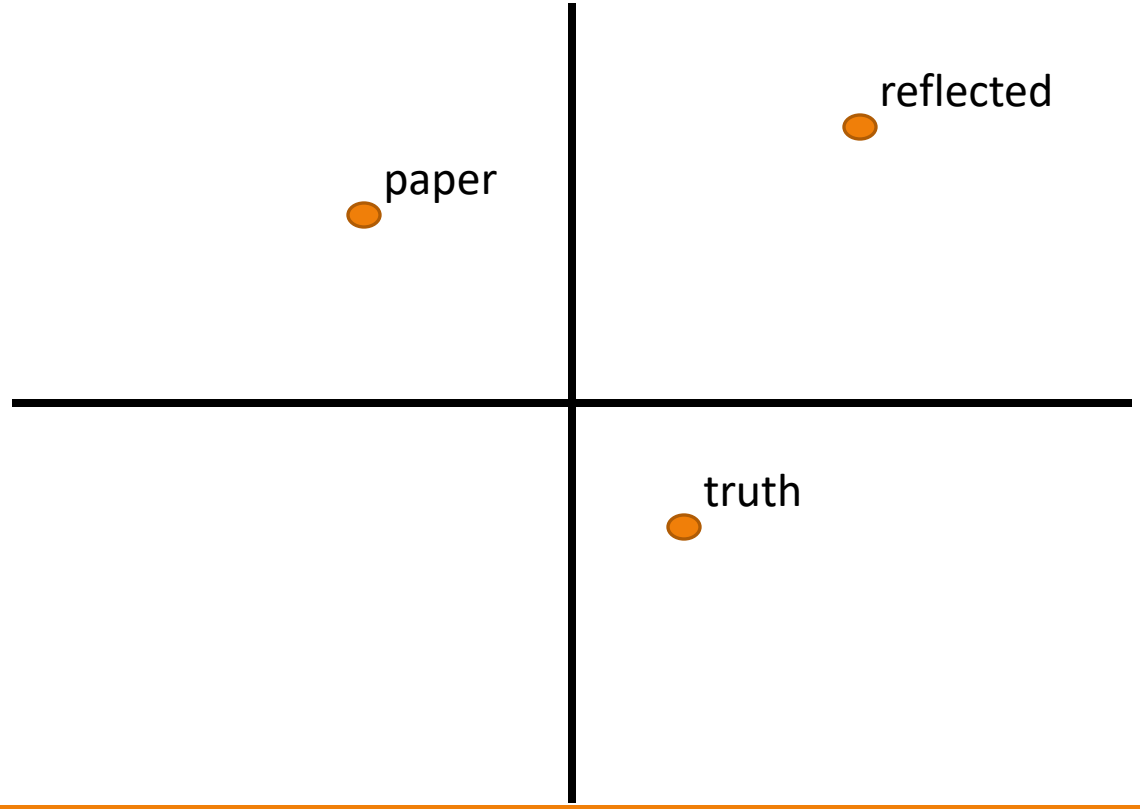
Continuous Meaning

The paper reflected the truth.



Continuous Meaning

The paper reflected the truth.



glean

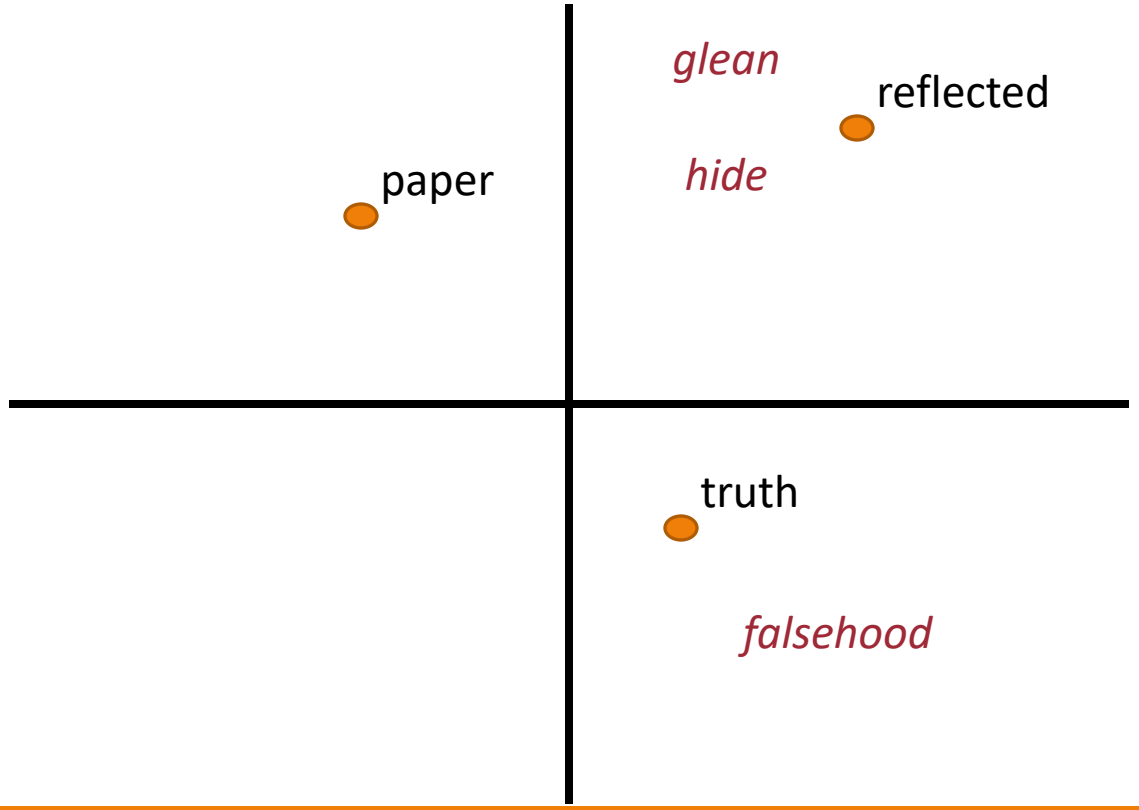
hide

falsehood

where might these go in this space?

Continuous Meaning

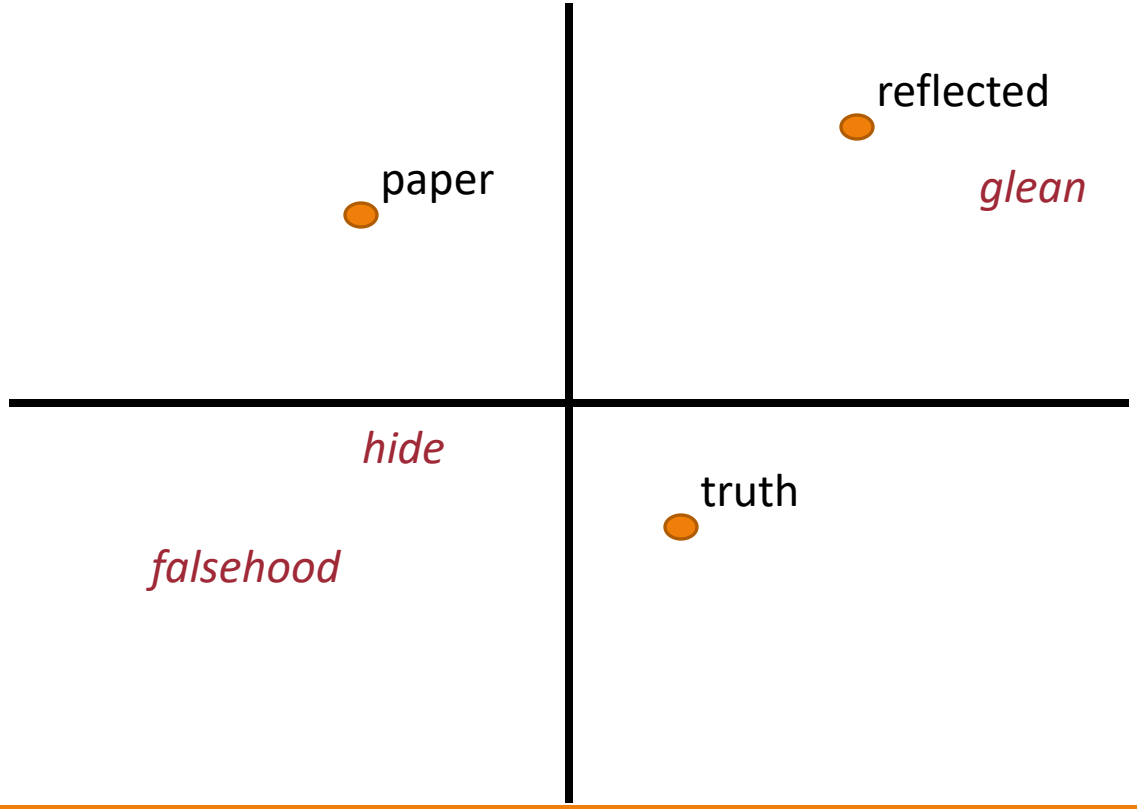
The paper reflected the truth.



One option

Continuous Meaning

The paper reflected the truth.



Another option

(Some) Properties of Embeddings

https://media3.giphy.com/media/3orif0M8U1E7NfpFzq/200_s.gif

Capture “like” (similar) words

target:	Redmond	Havel	ninjutsu	graffiti	capitulate
	Redmond Wash.	Vaclav Havel	ninja	spray paint	capitulation
	Redmond Washington	president Vaclav Havel	martial arts	grafitti	capitulated
	Microsoft	Velvet Revolution	swordsmanship	taggers	capitulating

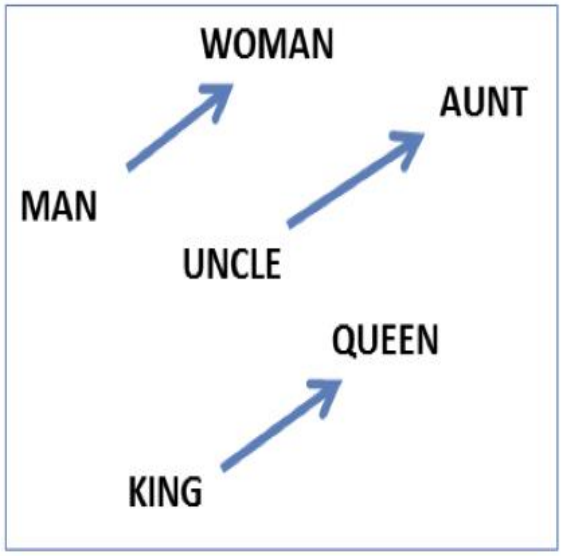
(Some) Properties of Embeddings

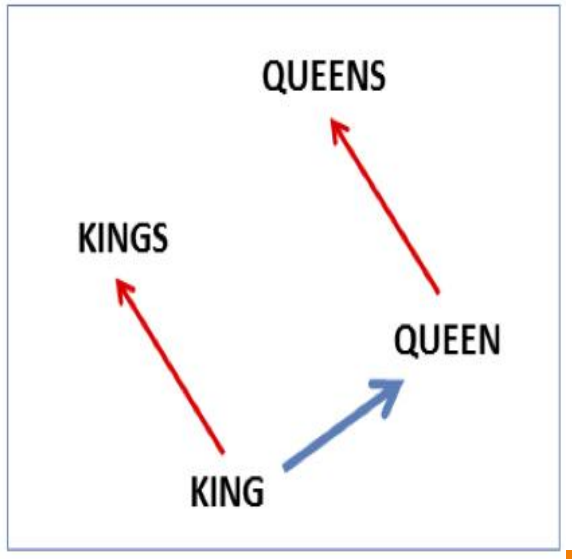
https://media3.giphy.com/media/3orif0M8U1E7NfpFzq/200_s.gif

Capture "like" (similar) words

target:	Redmond	Havel	ninjutsu	graffiti	capitulate
	Redmond Wash.	Vaclav Havel	ninja	spray paint	capitulation
	Redmond Washington	president Vaclav Havel	martial arts	grafitti	capitulated
	Microsoft	Velvet Revolution	swordsmanship	taggers	capitulating

Capture relationships





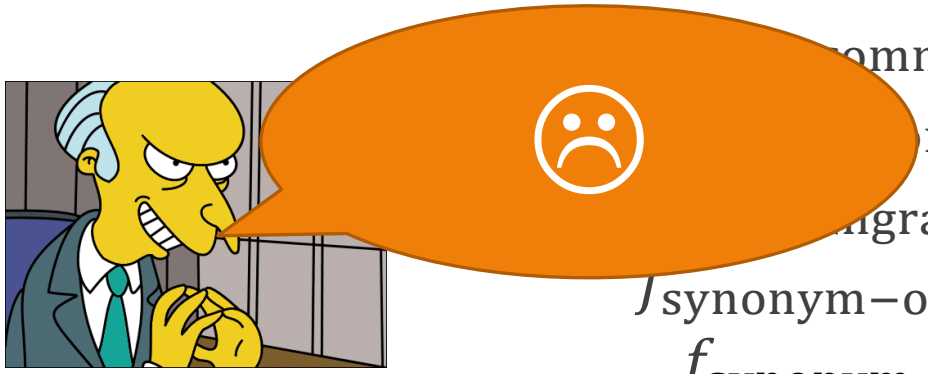
vector(*'king'*) – vector(*'man'*) + vector(*'woman'*) ≈ vector('queen')

vector(*'Paris'*) - vector(*'France'*) + vector(*'Italy'*) ≈ vector('Rome')

Case Study: Plagiarism Detector (Feature Example)

Given two documents x_1, x_2, predict y = 1 (plagiarized) or y = 0 (not plagiarized)

Intuition: documents are more likely to be plagiarized if they have words in common



$f_{\text{common-word,Plag.}}(x_1, x_2) = ???$

$f_{\text{rd v>,Plag.}}(x_1, x_2) = ???$

$f_{\text{ngram Z>,Plag.}}(x_1, x_2) = ???$

$f_{\text{synonym-of-<word v>,Plag.}}(x_1, x_2) = ???$

$f_{\text{synonym-of-<ngram Z>,Plag.}}(x_1, x_2) =$

get_similarity_with_embeddings()

Image: http://3.bp.blogspot.com/_MWJDzvrnu7Y/TTcXpCWZerI/AAAAAAAAAWY/zxu1GwrIBz0/s1600/excellent-mr-burns.gif

Think-Pair-Share: Embedding Similarity

https://vectors.nlpl.eu/explore/embeddings/en/

These embeddings are created from Wikipedia. Consider how the words that are similar to them might have been calculated. That is, what articles do you think the words were found in?

Vector Representations

Key Ideas

Vector embeddings can be used for phrases, paragraphs, or even whole documents!

1. Acquire basic contextual statistics (often counts) for each word type v

2. Extract a real-valued vector e_v for each word v from those statistics

[0.00315225, 0.00315225, 0.00547597, 0.00741556, 0.00912817, 0.01068435, 0.01212381, 0.01347162, 0.01474487, 0.0159558]

3. Use the vectors to represent each word in later tasks

Shared Intuition Across Common Embedding Types

Model the meaning of a word by “embedding” in a vector space

The meaning of a word is a vector of numbers

Contrast: word meaning is represented in many computational linguistic applications by a vocabulary index (“word number 545”) or the string itself

Three Common Kinds of Embedding Models

1. Co-occurrence matrices
2. ~~Matrix Factorization: Singular value decomposition/Latent Semantic Analysis, Topic Models~~
3. Neural-network-inspired models (skip-grams, CBOW)

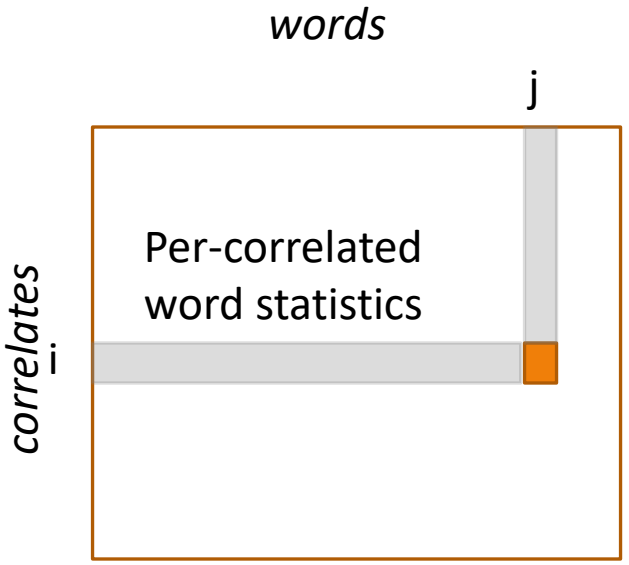
Three Common Kinds of Embedding Models

1. Co-occurrence matrices
2. ~~Matrix Factorization: Singular value decomposition/Latent Semantic Analysis, Topic Models~~
3. Neural-network-inspired models (skip-grams, CBOW)

Co-occurrence matrices can be used in their own right, but they're most often used as inputs (directly or indirectly) to the matrix factorization or neural approaches

Co-occurrence Matrix

Acquire basic contextual statistics (often counts) for each word type v via *correlate*.



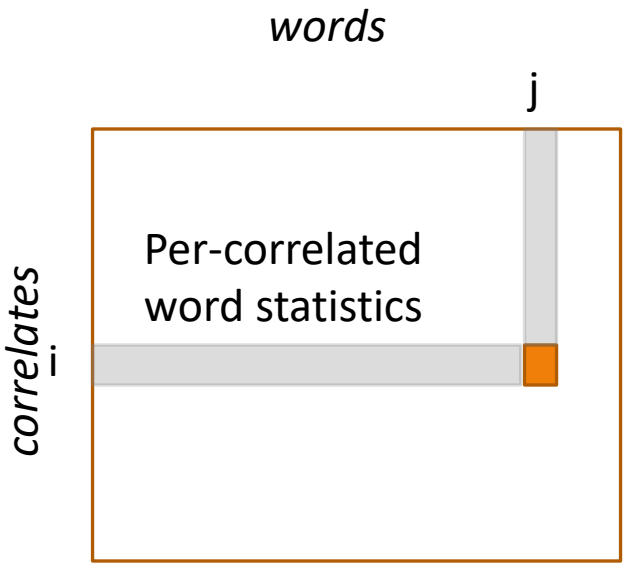
Co-occurrence Matrix

Acquire basic contextual statistics (often counts) for each word type v via *correlate*:

For example:

documents

- Record how often a word occurs in each document



correlates =
documents

Co-occurrence Matrix

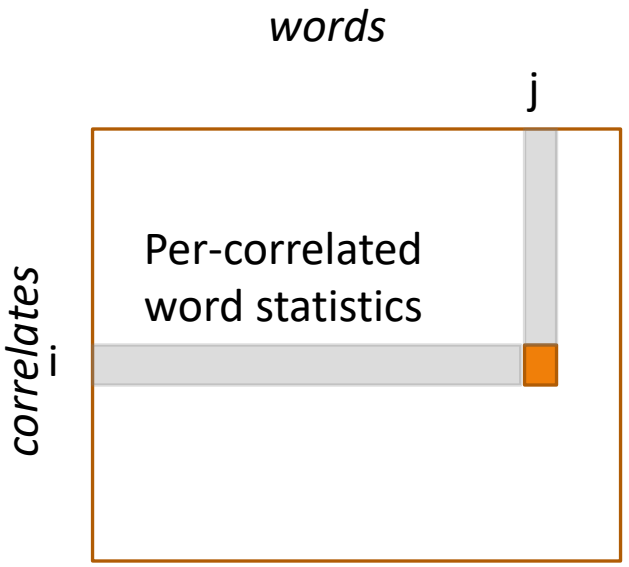
Acquire basic contextual statistics (often counts) for each word type v via *correlate*:

For example:

documents

surrounding context words

- Record how often v occurs with other word types u



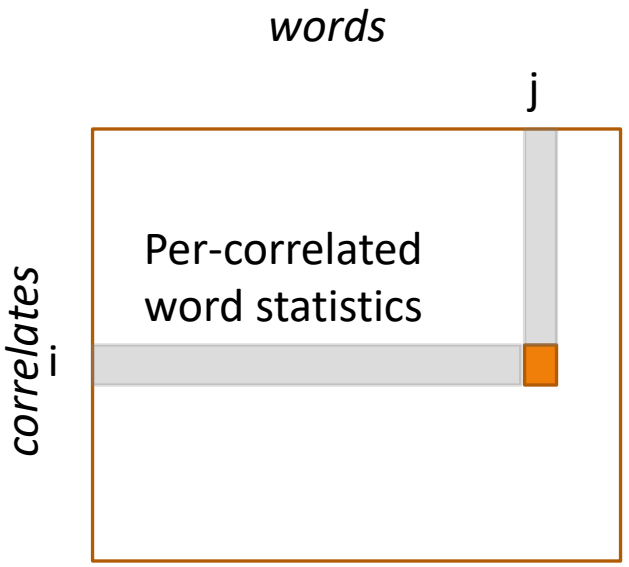
correlates =
word types

Co-occurrence Matrix

Acquire basic contextual statistics (often counts) for each word type v via *correlate*:

For example:

- documents
- surrounding context words
- linguistic annotations (POS tags, syntax)
- …



Assumption: Two words are similar if their vectors are similar

“Acquire basic contextual statistics (often counts) for each word type v”

Two basic, initial counting approaches

- Record which words appear in which documents
- Record which words appear together

These are good first attempts, but with some large downsides

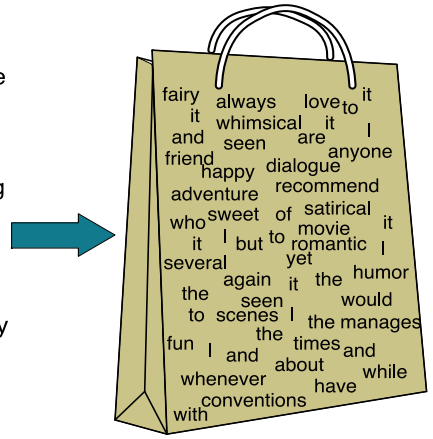
Co-Occurrence Vectors

document (↓)-word (→) count matrix

	battle	soldier	fool	clown
As You Like It	1	2	37	6
Twelfth Night	1	2	58	117
Julius Caesar	8	12	1	0
Henry V	15	36	5	0

basic bag-of-words counting

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

Co-Occurrence Vectors

document (↓)-word (→) count matrix

	battle	soldier	fool	clown
As You Like It	1	2	37	6
Twelfth Night	1	2	58	117
Julius Caesar	8	12	1	0
Henry V	15	36	5	0

Assumption: Two documents are similar if their vectors are similar

Co-Occurrence Vectors

document (↓)-word (→) count matrix

	battle	soldier	fool	clown
As You Like It	1	2	37	6
Twelfth Night	1	2	58	117
Julius Caesar	8	12	1	0
Henry V	15	36	5	0

Assumption: Two words are similar if their vectors are similar???

Co-Occurrence Vectors

document (↓)-word (→) count matrix

	battle	soldier	fool	clown
As You Like It	1	2	37	6
Twelfth Night	1	2	58	117
Julius Caesar	8	12	1	0
Henry V	15	36	5	0

Assumption: Two words are similar if their vectors are similar

Issue: Count word vectors are very large, sparse, and skewed!

Co-Occurrence Vectors

context (↓)-word (→) count matrix

	apricot	pineapple	digital	information
aardvark	0	0	0	0
computer	0	0	2	1
data	0	10	1	6
pinch	1	1	0	0
result	0	0	1	4
sugar	1	1	0	0

Context: those other words within a small "window" of a target word

Co-Occurrence Vectors

context (↓)-word (→) count matrix

	apricot	pineapple	digital	information
aardvark	0	0	0	0
computer	0	0	2	1
data	0	10	1	6
pinch	1	1	0	0
result	0	0	1	4
sugar	1	1	0	0

Context: those other words within a small "window" of a target word

a cloud computer stores digital data on a remote computer

Co-Occurrence Vectors

context (↓)-word (→) count matrix

	apricot	pineapple	digital	information
aardvark	0	0	0	0
computer	0	0	2	1
data	0	10	1	6
pinch	1	1	0	0
result	0	0	1	4
sugar	1	1	0	0

Context: those other words within a small "window" of a target word

Assumption: Two words are similar if their vectors are similar

Issue: Count word vectors are very large, sparse, and skewed!

Pointwise Mutual Information (PMI): Dealing with Problems of Raw Counts

Raw word frequency is not a great measure of association between words

- It's very skewed: "the" and "of" are very frequent, but maybe not the most discriminative

We'd rather have a measure that asks whether a context word is **particularly informative** about the target word.

Pointwise mutual information:

Do events x and y co-occur more than if they were independent?

probability words x and y occur together (in the same context/window)

$$\text{PMI}(x, y) = \log \frac{p(x, y)}{p(x)p(y)}$$

probability that word x occurs

probability that word y occurs

Three Common Kinds of Embedding Models

1. Co-occurrence matrices
2. ~~Matrix Factorization: Singular value decomposition/Latent Semantic Analysis, Topic Models~~
3. Neural-network-inspired models (skip-grams, CBOW)

Word2Vec

Mikolov et al. (2013; NeurIPS): “Distributed Representations of Words and Phrases and their Compositionality”

Revisits the context-word approach

Learn a model p(c | w) to predict a context word from a target word

Word2Vec

Mikolov et al. (2013; NeurIPS): “Distributed Representations of Words and Phrases and their Compositionality”

Revisits the context-word approach

Learn a model p(c | w) to predict a context word from a target word

Learn two types of vector representations

- $h_c \in \mathbb{R}^E$: vector embeddings for each context word
- $v_w \in \mathbb{R}^E$: vector embeddings for each target word

$$p(c \mid w) \propto \exp(h_c^T v_w)$$

Word2Vec

context (↓)-word (→) count matrix

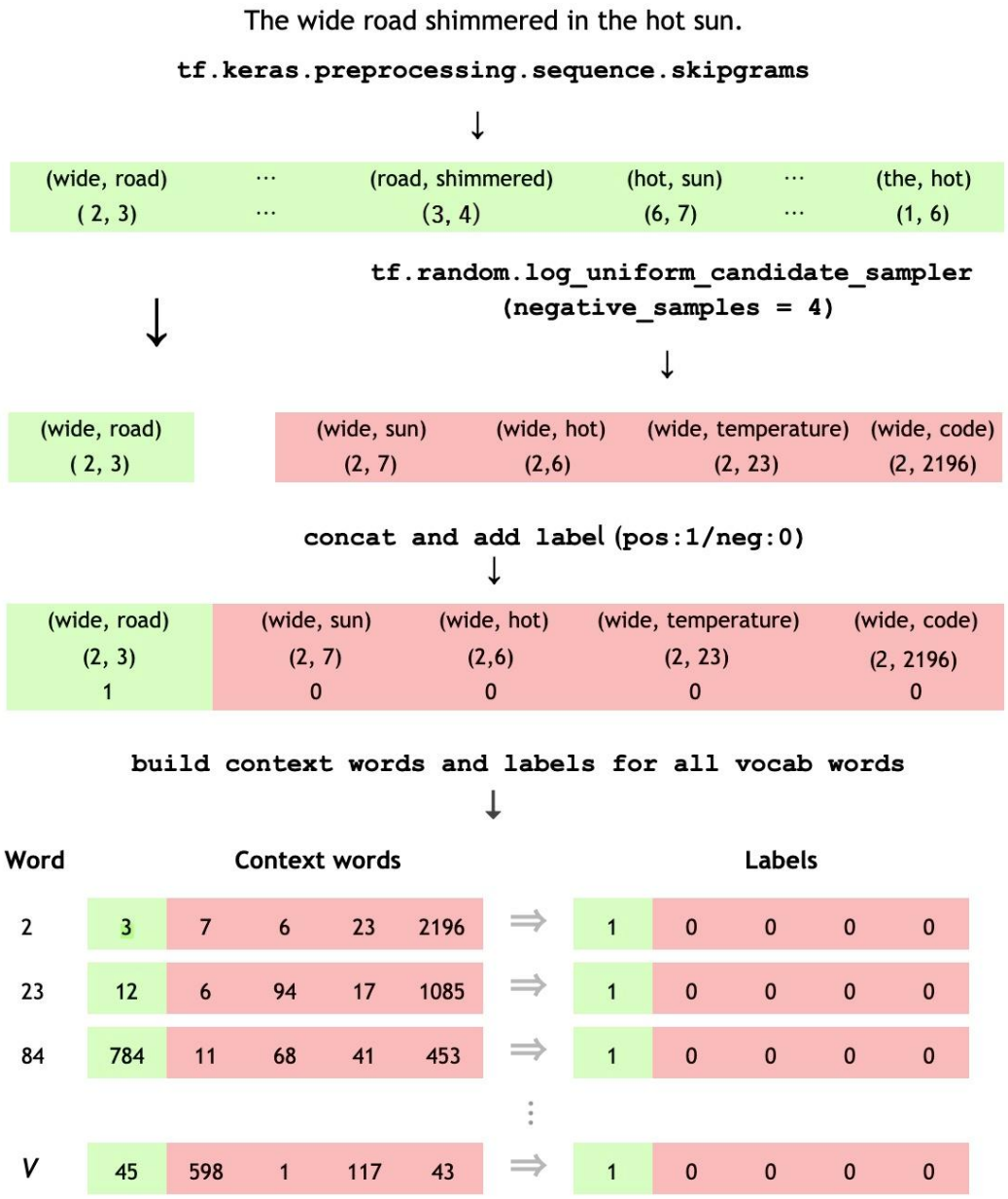
	apricot	pineapple	digital	information
aardvark	0	0	0	0
computer	0	0	2	1
data	0	10	1	6
pinch	1	1	0	0
result	0	0	1	4
sugar	1	1	0	0

Context: those other words within a small "window" of a target word

$$\max_{h,v} \sum_{c,w \text{ pairs}} \text{count}(c,w) \log p(c \mid w)$$

$$p(c \mid w) \propto \exp(h_c^T v_w)$$

Example (Tensorflow)



https://www.tensorflow.org/text/tutorials/word2vec

Word2Vec has Inspired a Lot of Work

Off-the-shelf embeddings

- https://code.google.com/archive/p/word2vec/

Off-the-shelf implementations

- https://radimrehurek.com/gensim/models/word2vec.html

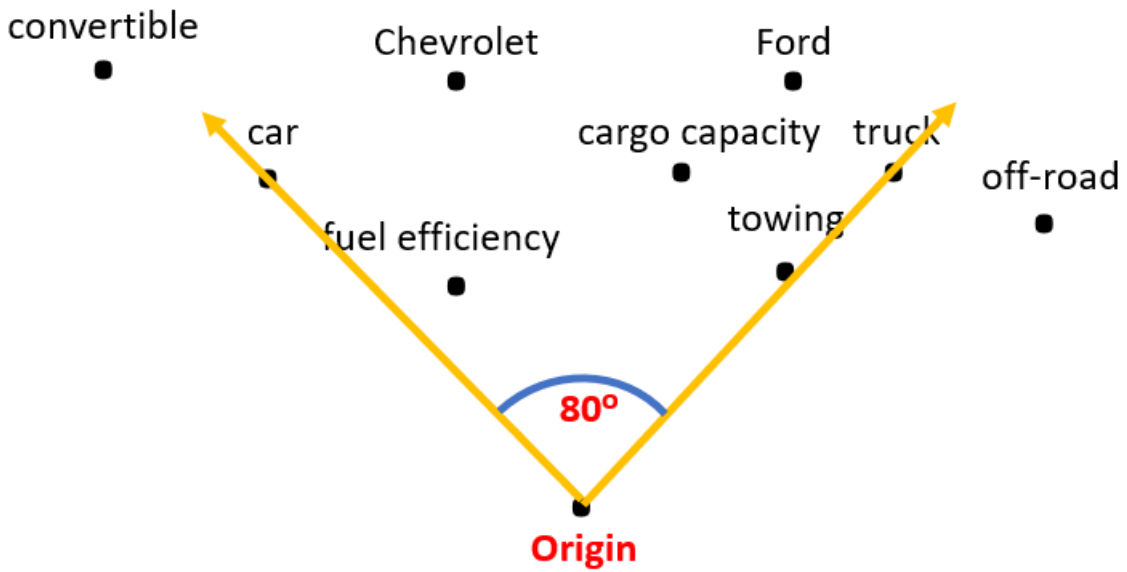
Follow-on work

- J. Pennington, R. Socher, and C. D. Manning, "**GLoVe: Global Vectors for Word Representation**," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014, pp. 1532–1543. doi: 10.3115/v1/D14-1162.
  - https://nlp.stanford.edu/projects/glove/
- Many others
- 15000+ citations

Comparing/Evaluating Word Embeddings

Cosine: Measuring Similarity

Given 2 target words v and w how similar are their vectors?



https://upload.wikimedia.org/wikipedia/commons/2/23/CosineSimilarity.png

Cosine: Measuring Similarity

Given 2 target words v and w how similar are their vectors?

Dot product or inner product from linear algebra

$$\text{dot-product}(\vec{v},\vec{w}) = \vec{v}\cdot\vec{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N$$

- High when two vectors have large values in same dimensions, low for orthogonal vectors with zeros in complementary distribution

Correct for high magnitude vectors $\dfrac{\vec{a}\cdot\vec{b}}{|\vec{a}||\vec{b}|}$

Cosine Similarity

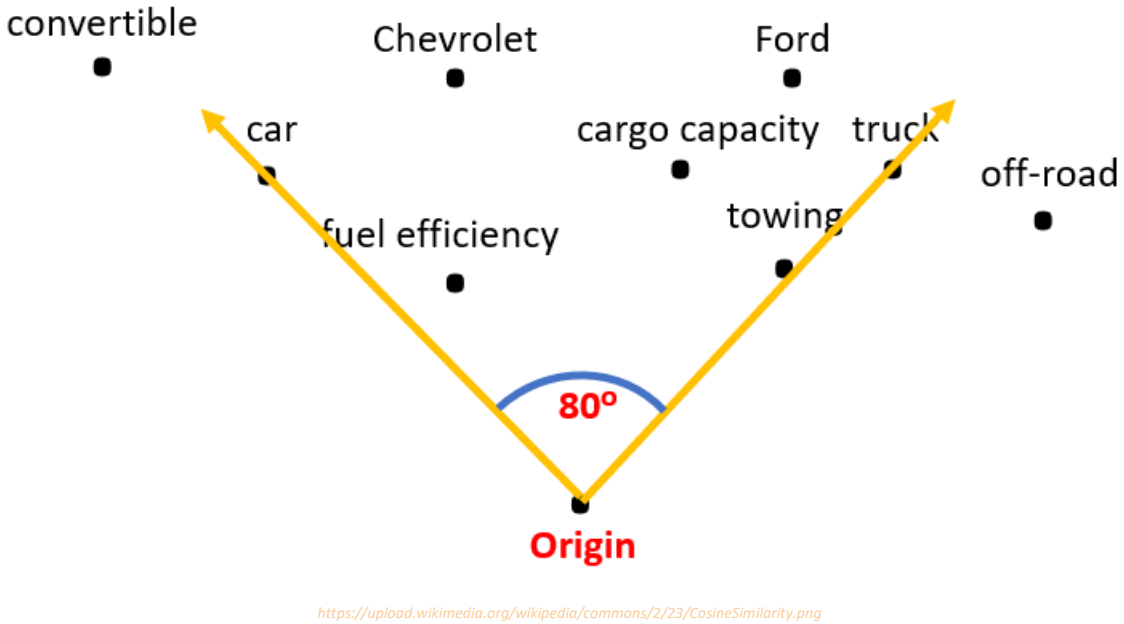
Divide the dot product by the length of the two vectors

$$\frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}$$

This is the cosine of the angle between them

$$\vec{a} \cdot \vec{b} = |\vec{a}||\vec{b}| \cos\theta$$

$$\frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|} = \cos\theta$$



https://upload.wikimedia.org/wikipedia/commons/2/23/CosineSimilarity.png

Example: Word Similarity

$$\cos(x, y) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}$$

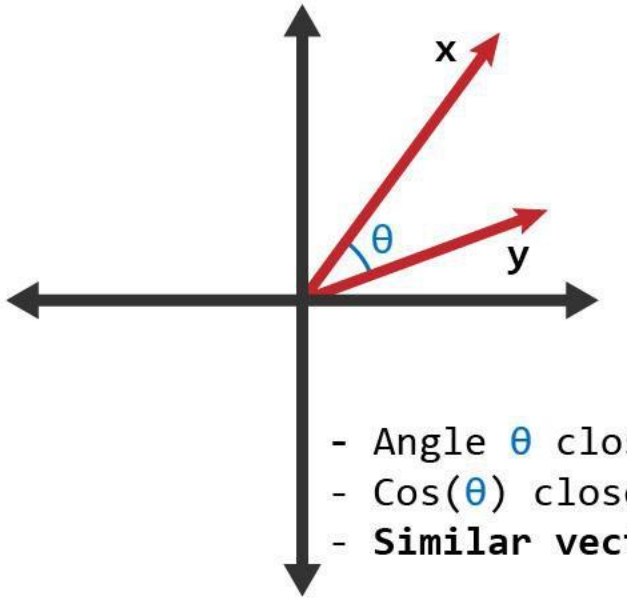
	Dim. 1	Dim. 2	Dim. 3
apricot	2	0	0
digital	0	1	2
information	1	6	1

cosine(apricot,information) = $\frac{2+0+0}{\sqrt{4+0+0}\sqrt{1+36+1}} = 0.1622$

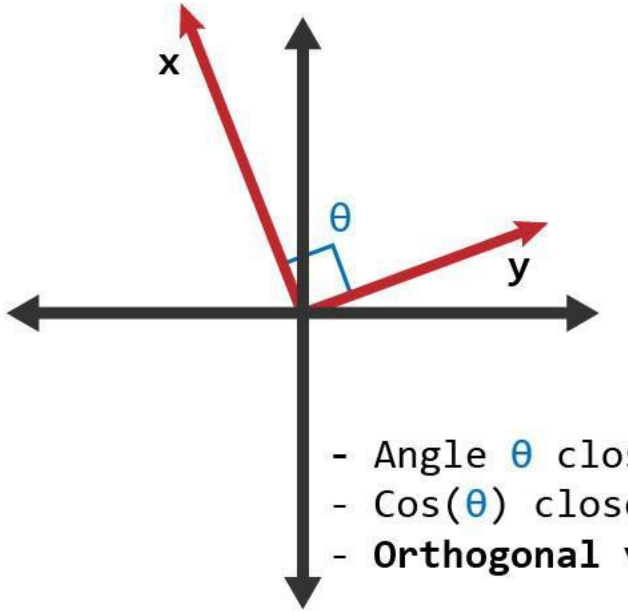
cosine(digital,information) = $\frac{0+6+2}{\sqrt{0+1+4}\sqrt{1+36+1}} = 0.5804$

cosine(apricot,digital) = $\frac{0+0+0}{\sqrt{4+0+0}\sqrt{0+1+4}} = 0.0$

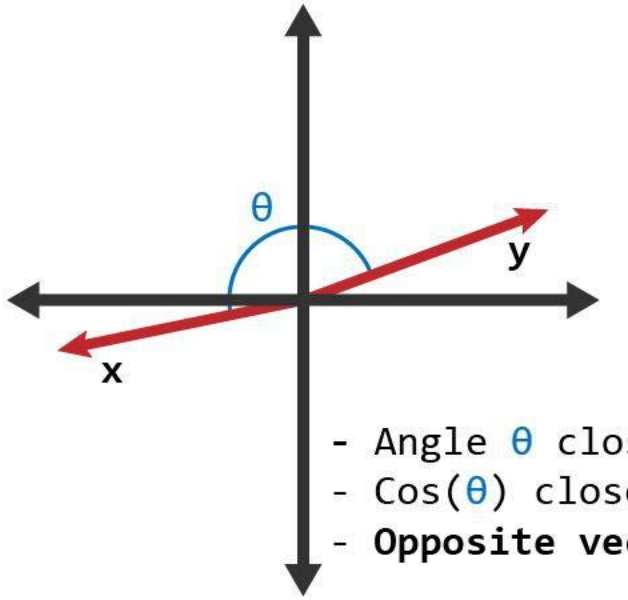
Cosine Similarity Range



- Angle θ close to 0
- Cos(θ) close to 1
- **Similar vectors**



- Angle θ close to 90
- Cos(θ) close to 0
- **Orthogonal vectors**



- Angle θ close to 180
- Cos(θ) close to -1
- **Opposite vectors**

https://www.learndatasci.com/glossary/cosine-similarity/

Other Similarity Measures

$$\text{sim}_{\text{cosine}}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i \times w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

$$\text{sim}_{\text{Jaccard}}(\vec{v},\vec{w}) = \frac{\sum_{i=1}^{N} \min(v_i, w_i)}{\sum_{i=1}^{N} \max(v_i, w_i)}$$

$$\text{sim}_{\text{Dice}}(\vec{v},\vec{w}) = \frac{2 \times \sum_{i=1}^{N} \min(v_i, w_i)}{\sum_{i=1}^{N} (v_i + w_i)}$$

$$\text{sim}_{\text{JS}}(\vec{v}||\vec{w}) = D\left(\vec{v}\,\middle|\,\frac{\vec{v}+\vec{w}}{2}\right) + D\left(\vec{w}\,\middle|\,\frac{\vec{v}+\vec{w}}{2}\right)$$

Other Neural Word Embedding Models

FastText

"Enriching Word Vectors with Subword Information" Bojanowski et al. (2017; TACL)

Main idea: learn **character n-gram embeddings** for the target word (not context) and modify the word2vec model to use these

Pre-trained models in 150+ languages

- https://fasttext.cc

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "**Enriching Word Vectors with Subword Information**," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017, doi: 10.1162/tacl_a_00051.

FastText Details

Main idea: learn **character n-gram embeddings** and for the target word (not the context) modify the word2vec model to use these

Original word2vec:

$$p(c\,|w) \propto \exp(h_c^T v_w)$$

FastText:

$$p(c\,|w) \propto \exp\left(h_c^T \left(\sum_{\text{n-gram } g \text{ in } w} z_g\right)\right)$$

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "**Enriching Word Vectors with Subword Information**," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017, doi: 10.1162/tacl_a_00051.

FastText Details

Main idea: learn **character n-gram embeddings** and for the target word (not the context) modify the word2vec model to use these

$$p(c \mid w) \propto \exp\left(h_c^T \left(\sum_{\text{n-gram } g \text{ in } w} z_g\right)\right)$$

decompose

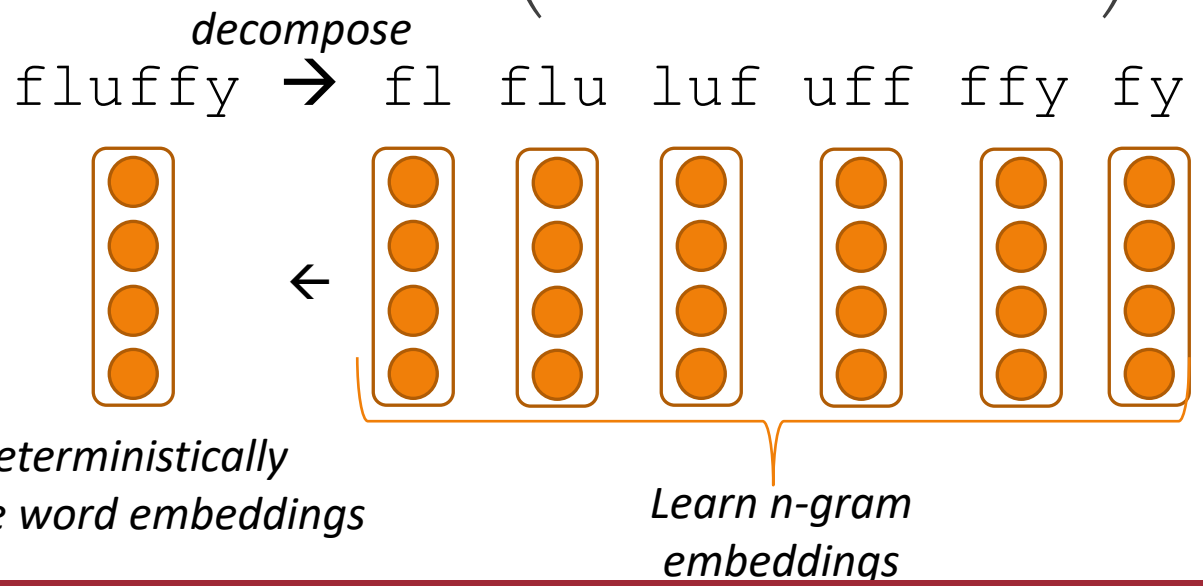
`fluffy` → `fl flu luf uff ffy fy`

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "**Enriching Word Vectors with Subword Information**," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017, doi: 10.1162/tacl_a_00051.

FastText Details

Main idea: learn **character n-gram embeddings** and for the target word (not the context) modify the word2vec model to use these

$$p(c \mid w) \propto \exp\left(h_c^T \left(\sum_{\text{n-gram } g \text{ in } w} z_g\right)\right)$$



P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "**Enriching Word Vectors with Subword Information**," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017, doi: 10.1162/tacl_a_00051.

Contextual Word Embeddings

Word2vec-based models are not context-dependent

- Single word type → single word embedding

If a single word type can have different meanings…

- bank, bass, plant,…

… why should we only have one embedding?

Entire task devoted to classifying these meanings:
Word Sense Disambiguation

Contextual Word Embeddings

Growing interest in this

Off-the-shelf is a bit more difficult

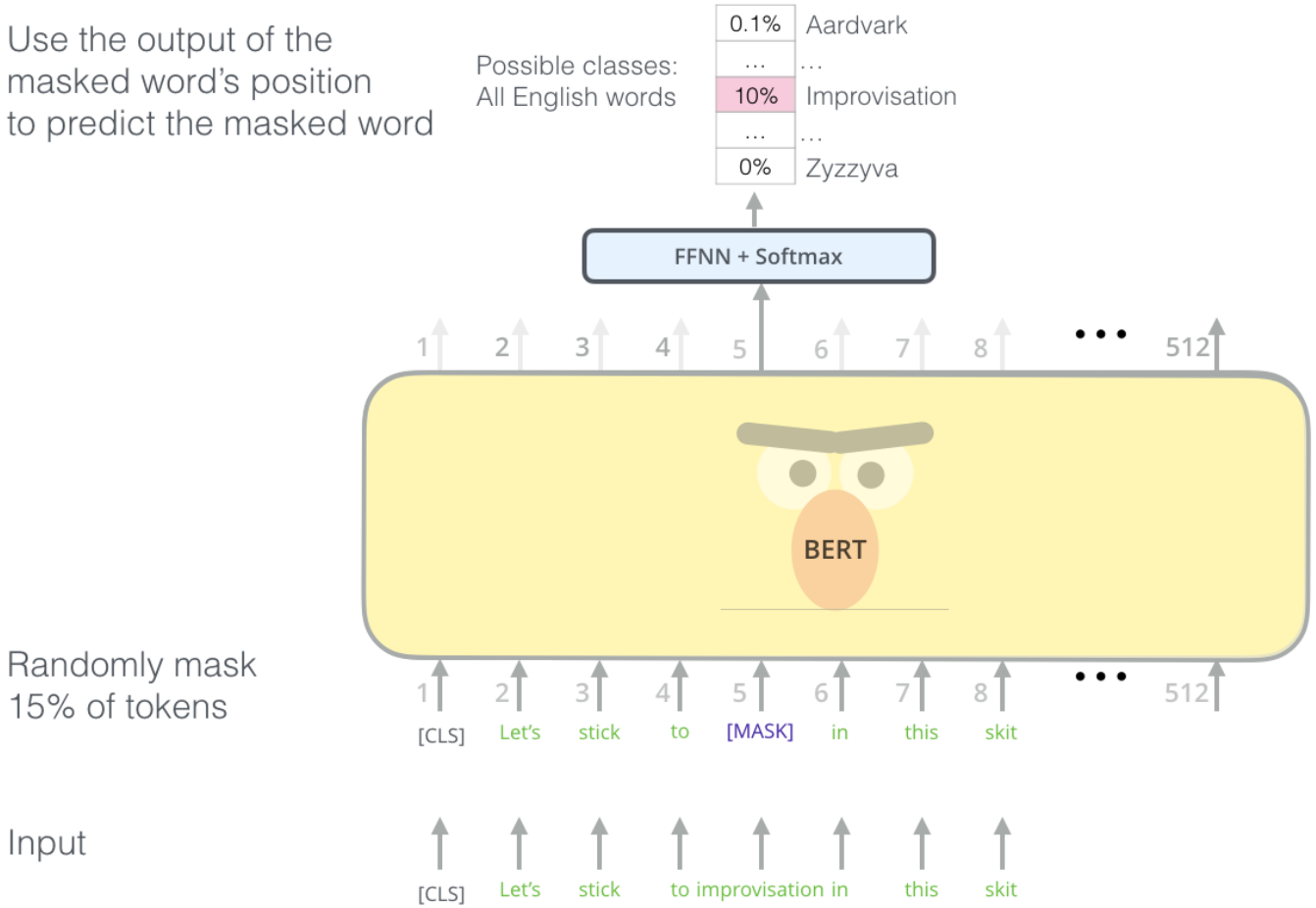
- Download and run a model
- Can't just download a file of embeddings

Two to know about (with code):

- ELMo: "Deep contextualized word representations" Peters et al. (2018; NAACL)
- https://allennlp.org/elmo
- BERT: "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" Devlin et al. (2019; NAACL)
  - https://github.com/google-research/bert

Back to Transformers…

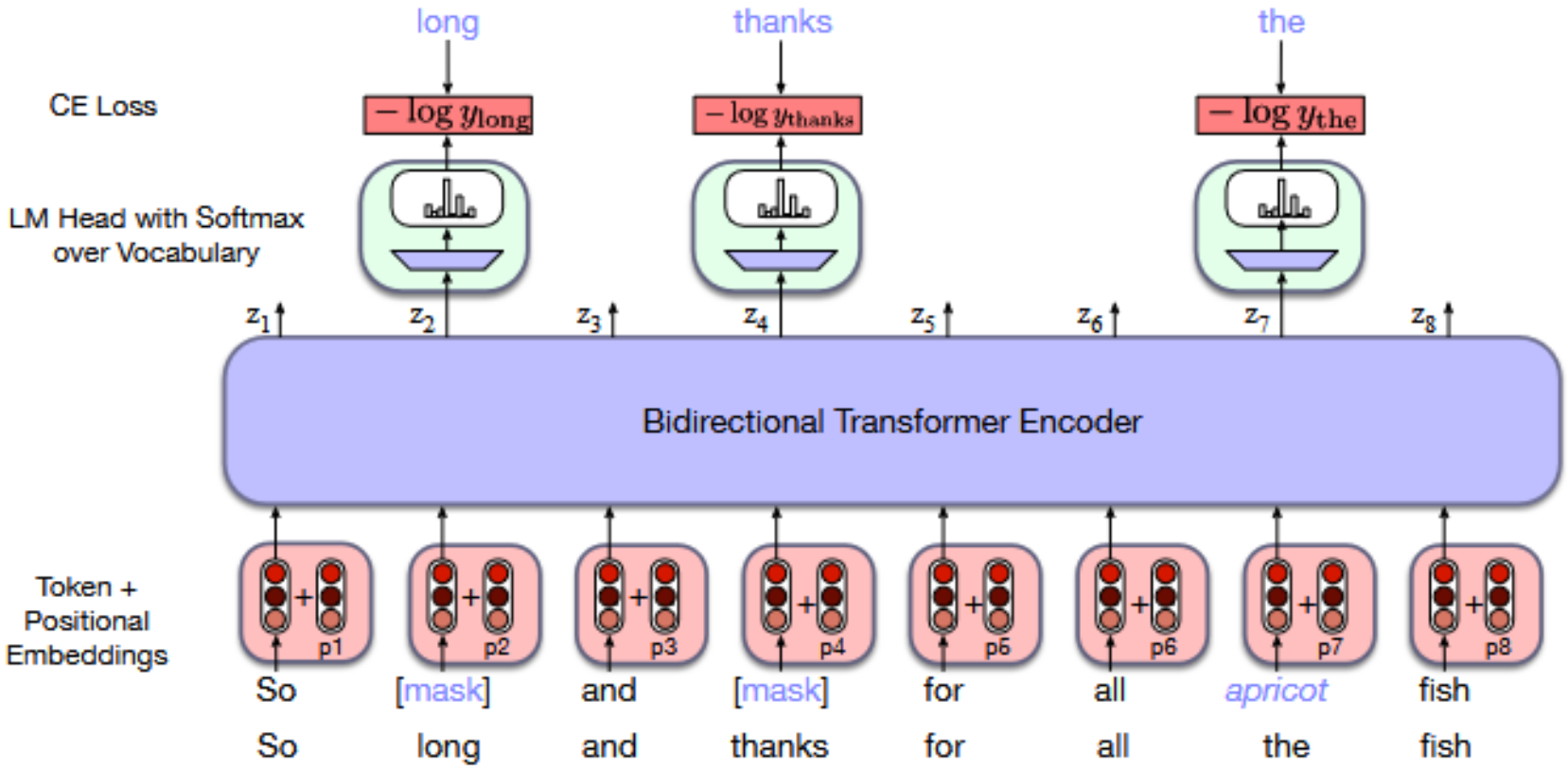
BERT (Devlin et al. 2019)



http://jalammar.github.io/illustrated-bert/

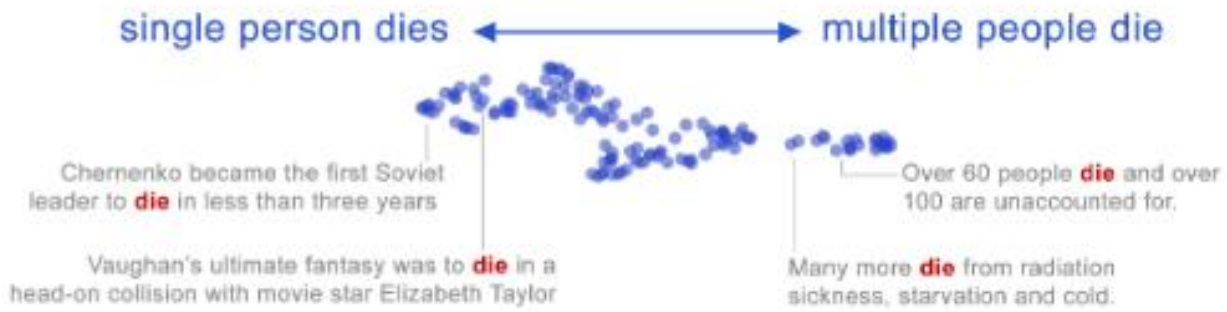
Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL), Volume 1 (Long and Short Papers)*, 4171–4186. https://doi.org/10.18653/v1/N19-1423

Masked Language Models



From Jurafsky & Martin's Speech and Language Processing, 3rd Edition, Chapter 11

Contextual Embeddings



From Jurafsky & Martin's Speech and Language Processing, 3rd Edition, Chapter 11

Uses of Encoder-Only Models

Classification tasks

Sentence embeddings

Context-dependent word embeddings

Any type of fill-in-the-blank tasks

BERT Question

Consider the highlighted words. Which two words would contextual word embeddings from BERT say are closest?

A. I am so excited to use my new **bat** at the baseball game tomorrow.

B. The favorite food of this species of **bat** is mosquitoes.

C. The **cardinal** isn't just a lawn decoration; the species makes themselves useful by eating mosquitoes.

PollEv.com/laramartin527

Word2Vec Question

Remember: word2vec is a dense vector embedding

Consider the highlighted words. Which two words would word2vec say are closest?

A. I am so excited to use my new **bat** at the baseball game tomorrow.

B. The favorite food of this species of **bat** is mosquitoes.

C. The **cardinal** isn't just a lawn decoration; the species makes themselves useful by eating mosquitoes.

PollEv.com/laramartin527

BERT Family of Models

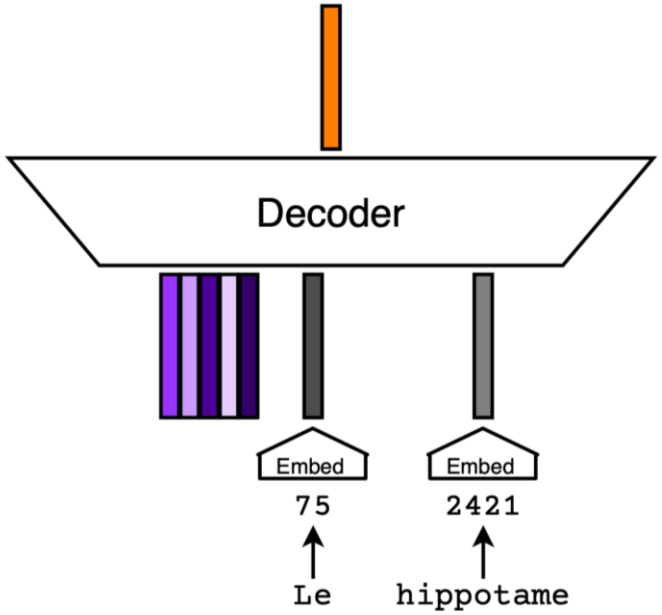
- Encoder-only
  - Input: Corrupted version of text sequence
  - Goal: Produce an uncorrupted version of text sequence
- How to use:
  - Finetune for a classification task
  - Extract word/sentence embeddings

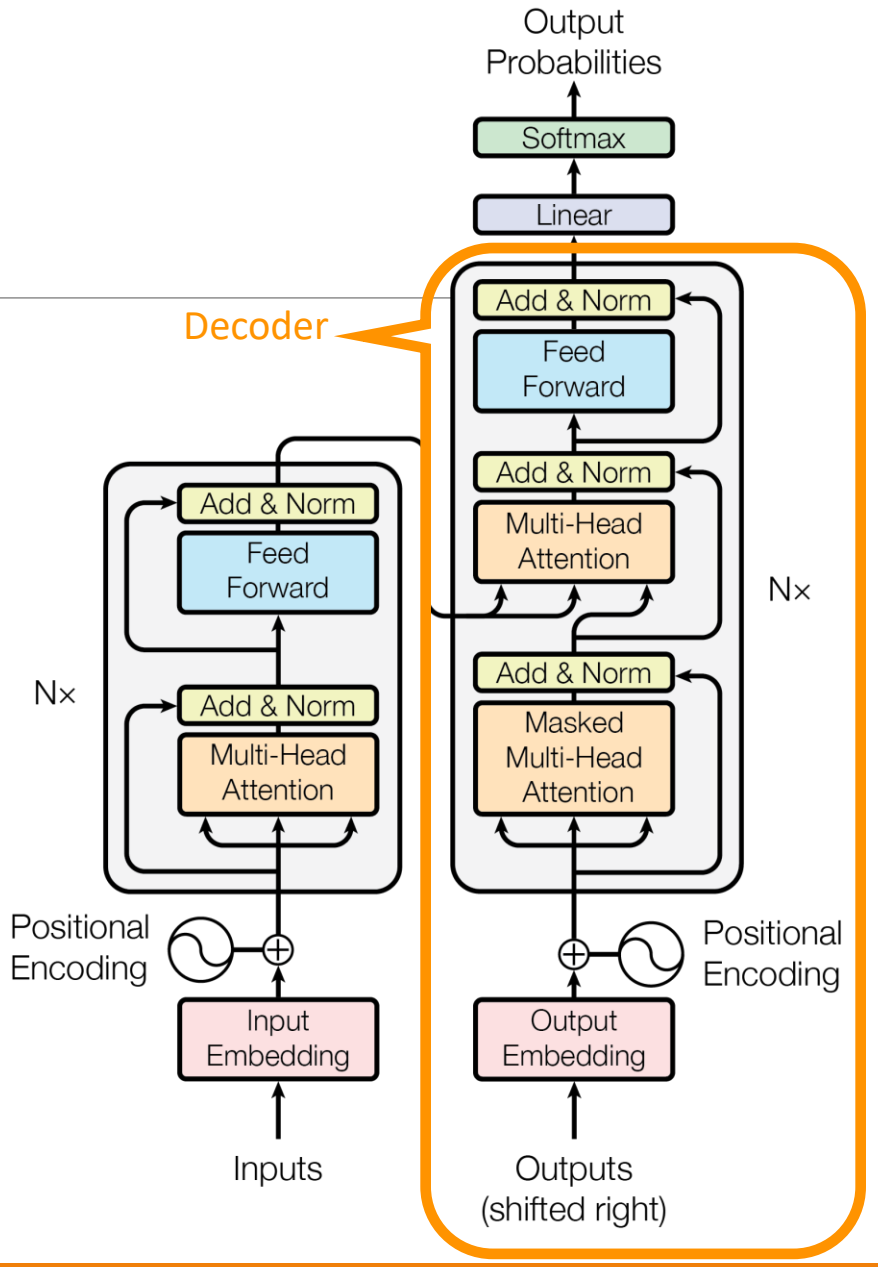
Some important BERT family members

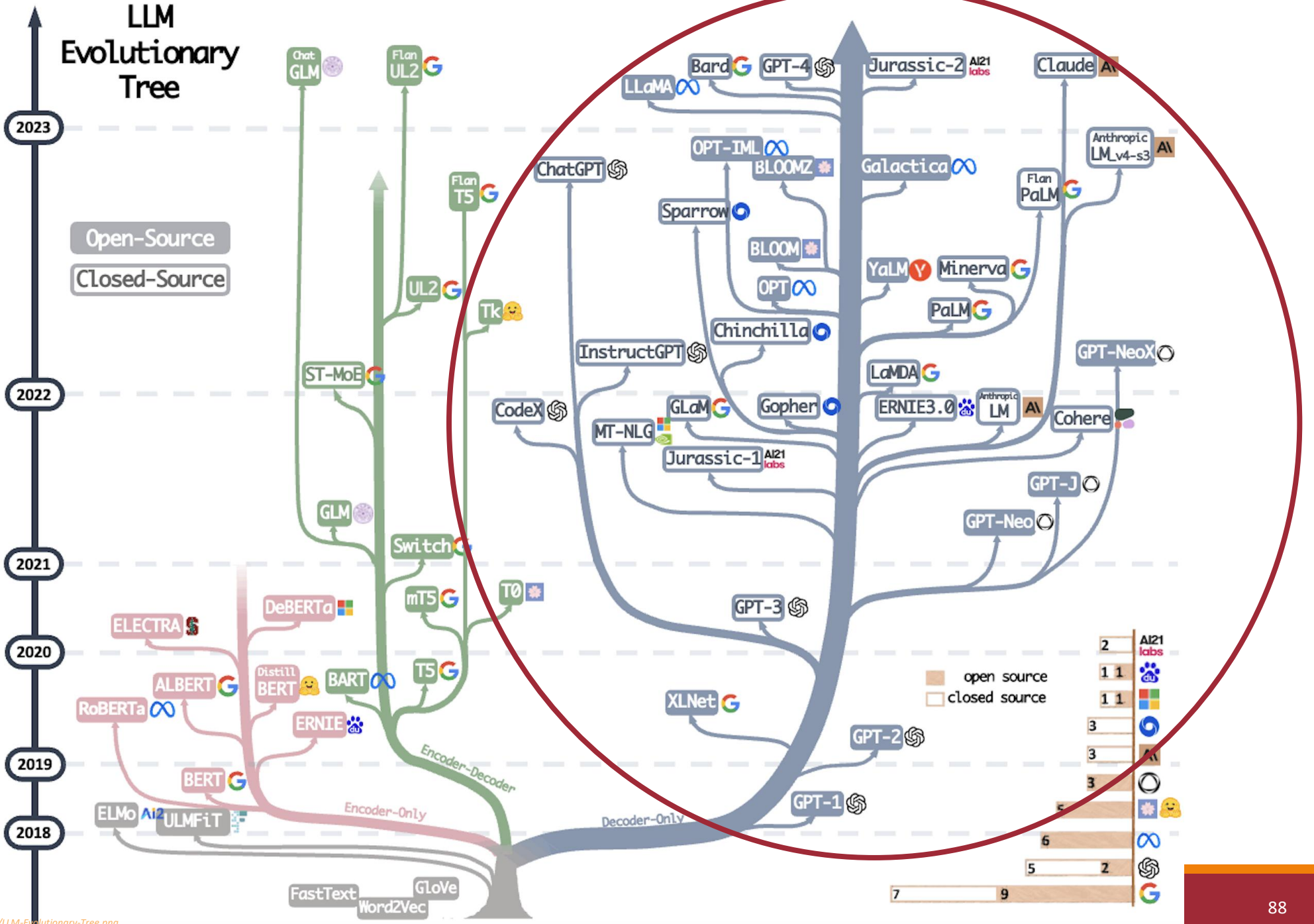
(in my opinion)

- RoBERTa (better version of the original BERT) – Liu et al. 2019 (Facebook)
- Sentence-BERT (BERT fine-tuned to give good sentence embeddings) – Reimers & Gurevych 2019 (Technische Universität Darmstadt)
- DistilBERT (lite BERT) – Sanh et al. 2019
- ALBERT (lite BERT) – Lan et al. 2020
- HuBERT (BERT for speech embeddings) – Hsu et al. 2021

Decoder-Only Models







https://blog.biocomm.ai/wp-content/uploads/2023/05/LLM-Evolutionary-Tree.png

GPT Family

- Decoder-only
  - Input: Text sequence
  - Goal: Predict the next word given the previous ones
- How to use:
  - Ask GPT* to continue from a prompt.
  - Finetune smaller GPTs for more customized generation tasks.
    - ChatGPT cannot be finetuned since it is already finetuned
  - Use OpenAI's API to get them to fine-tune GPT* for you.
- Around GPT-2 was when pre-trained models became popular
- Around GPT-3 was when *just* prompting became a thing

Other Decoder-Only Models

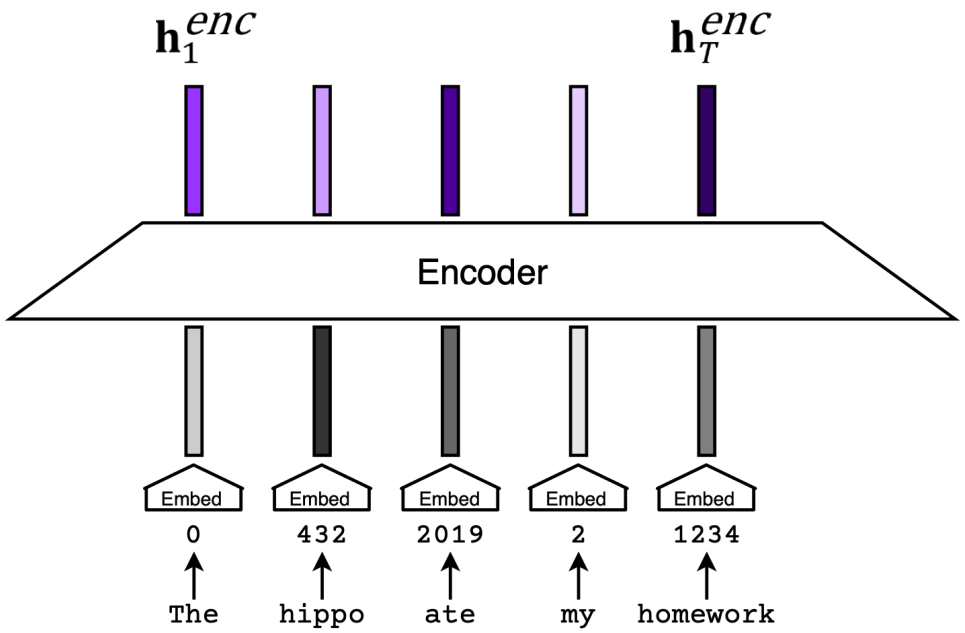
LLaMA 3/4 (Meta)

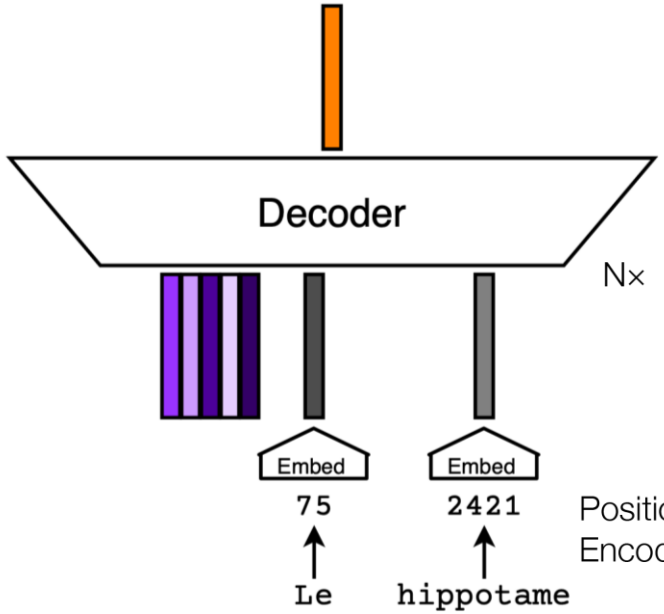
Claude 3 (Anthropic)

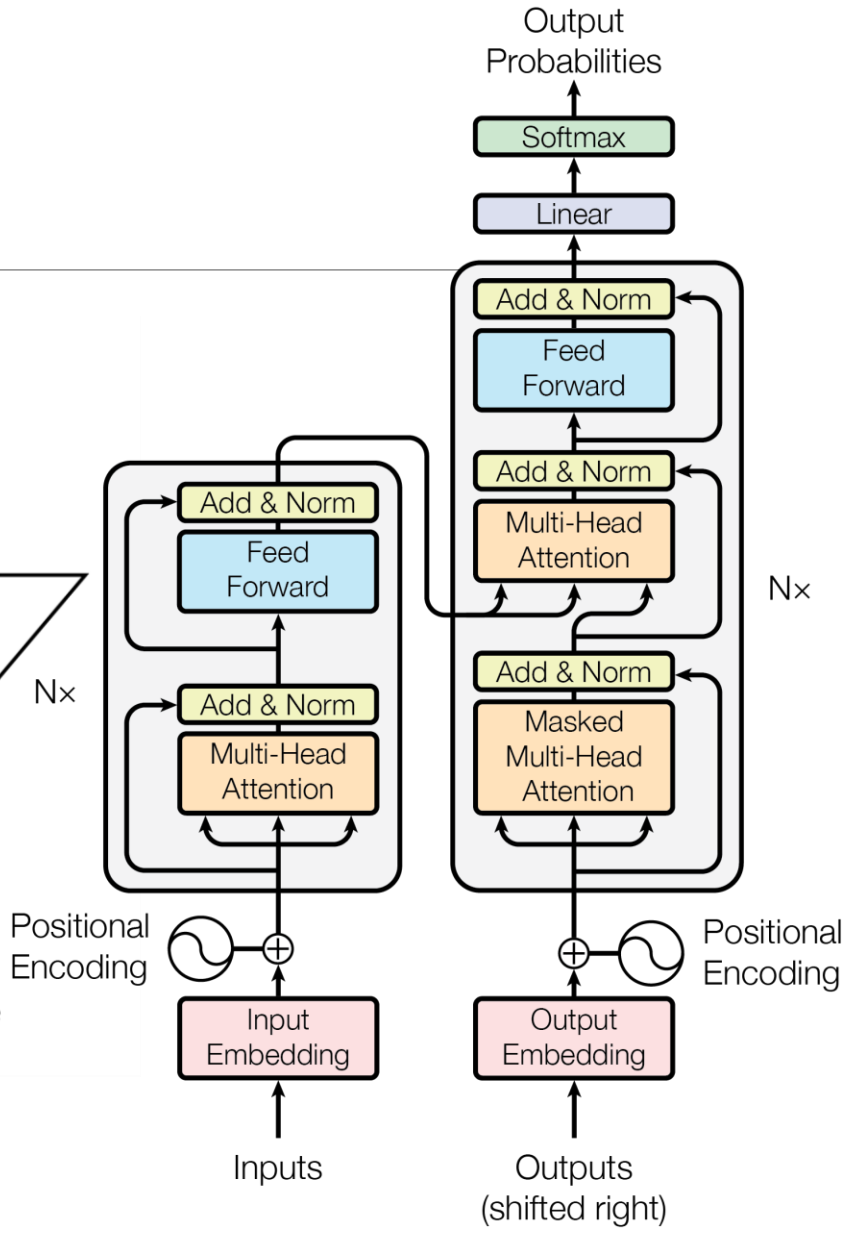
Gemma (Google)

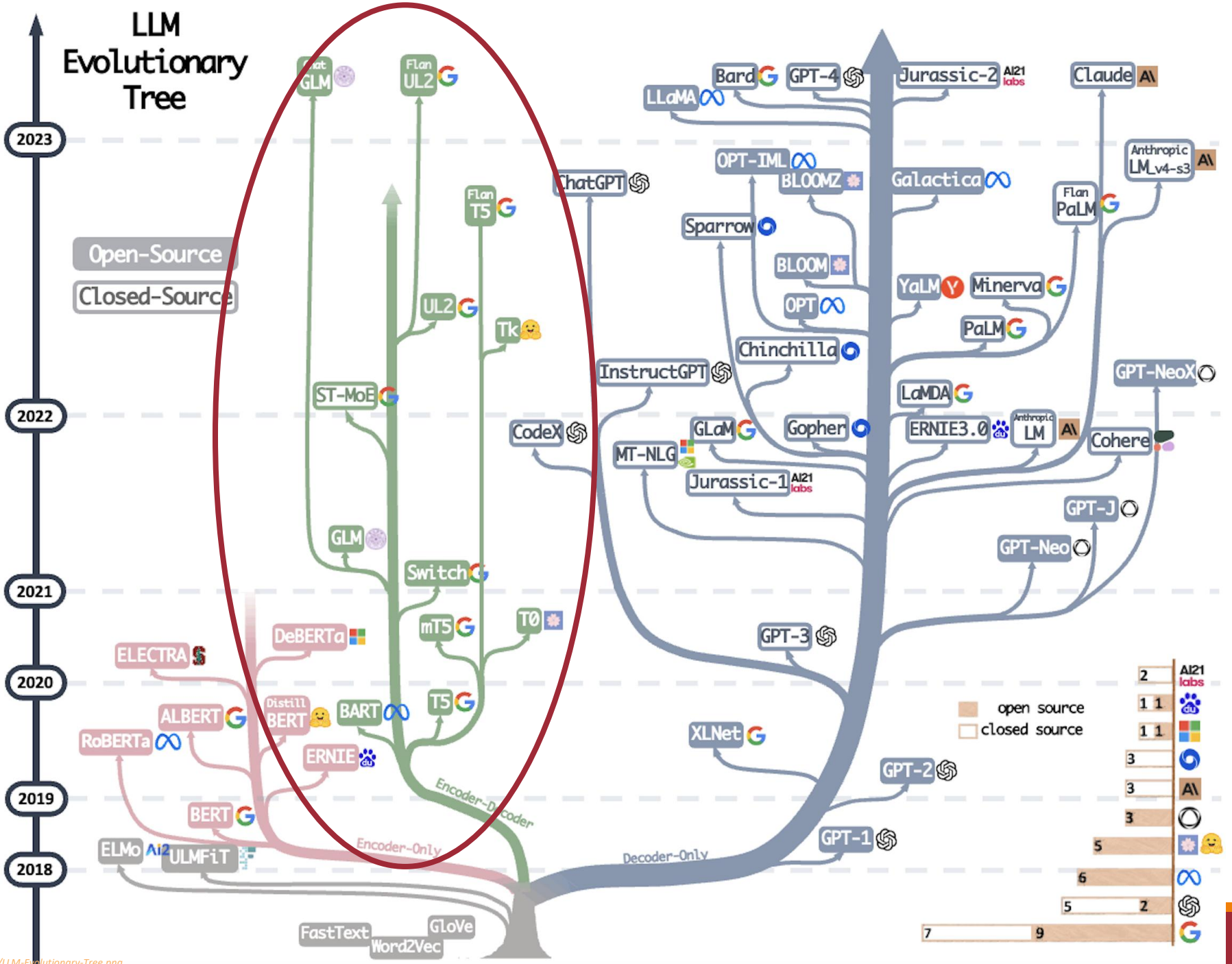
OLMo 2 (AI2)

Encoder-Decoder Models









https://blog.biocomm.ai/wp-content/uploads/2023/05/LLM-Evolutionary-Tree.png

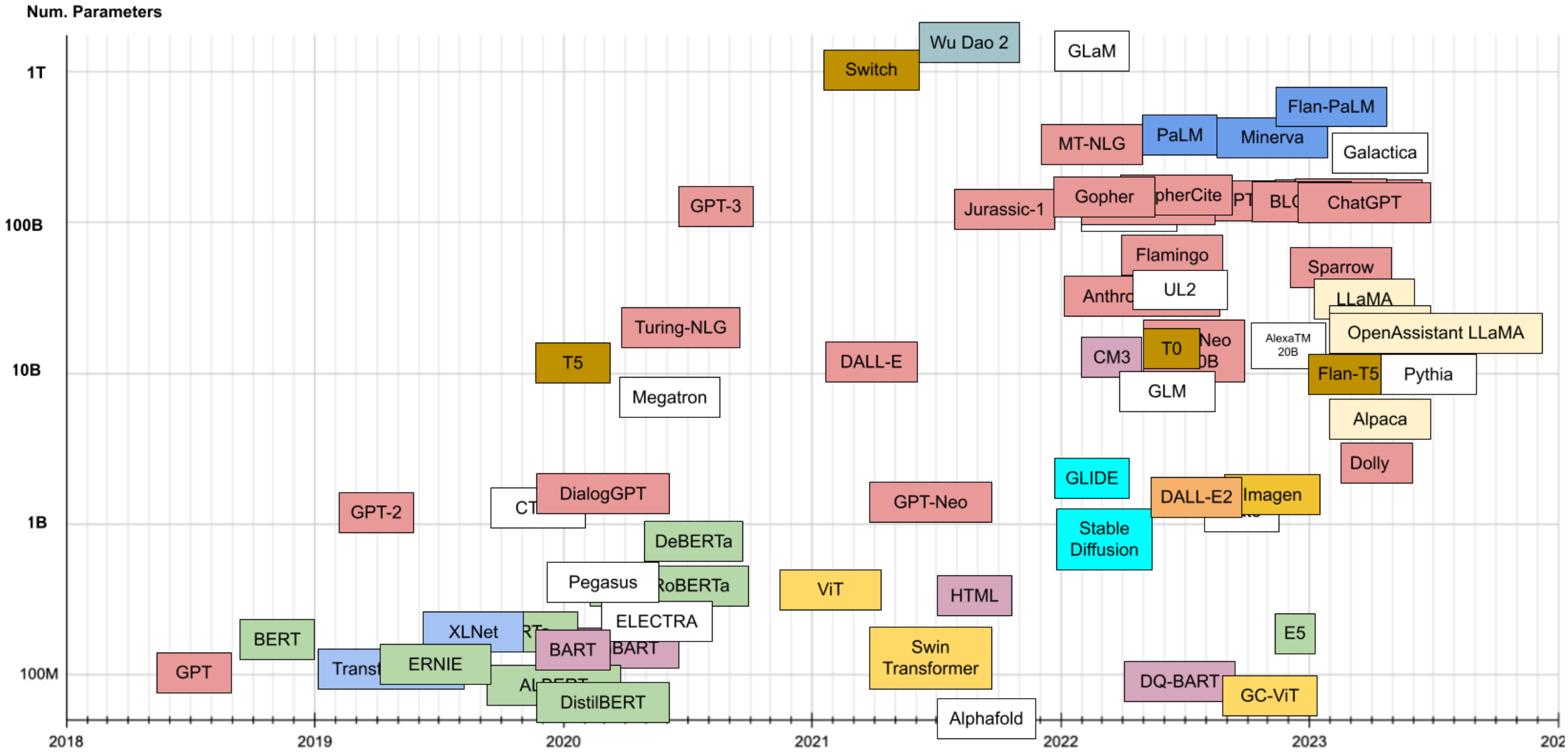
Enc-Dec Family of Models

- Encoder-decoder
  - Input: Text sequence with random word spans deleted
  - Goal: Generate the deleted word spans
- How to use:
  - Finetune smaller ones for either generation or classification tasks.
  - Prompt tuning (train a sequence of embedding which get prefixed to the input)

Some Enc-Dec family members

- T5 – Raffel et al. 2020 (Google)
- BART (combo of GPT and BERT) – (Facebook)

- DALL-E 2 (for caption prediction)



https://amatriain.net/blog/transformer-models-an-introduction-and-catalog-2d1e9039f376/