Text Mining and Transformers

Sampo Pyysalo

Applied deep learning in bioinformatics summer school
August 7th 2023, Copenhagen

TURKUNLP
.ORG

TURUN
YLIOPISTO

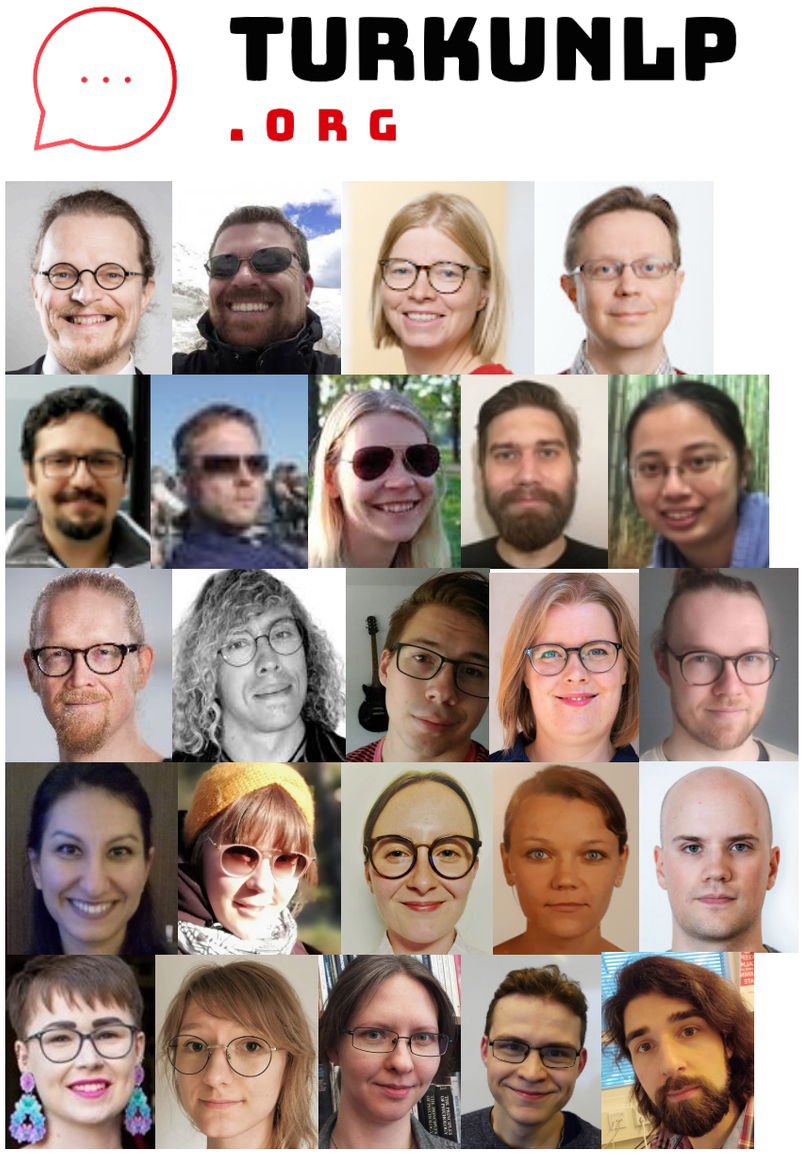
Self / group intro

Sampo Pyysalo

- Research fellow @ University of Turku
- CS / Machine Learning background
- Research on ML applied to NLP

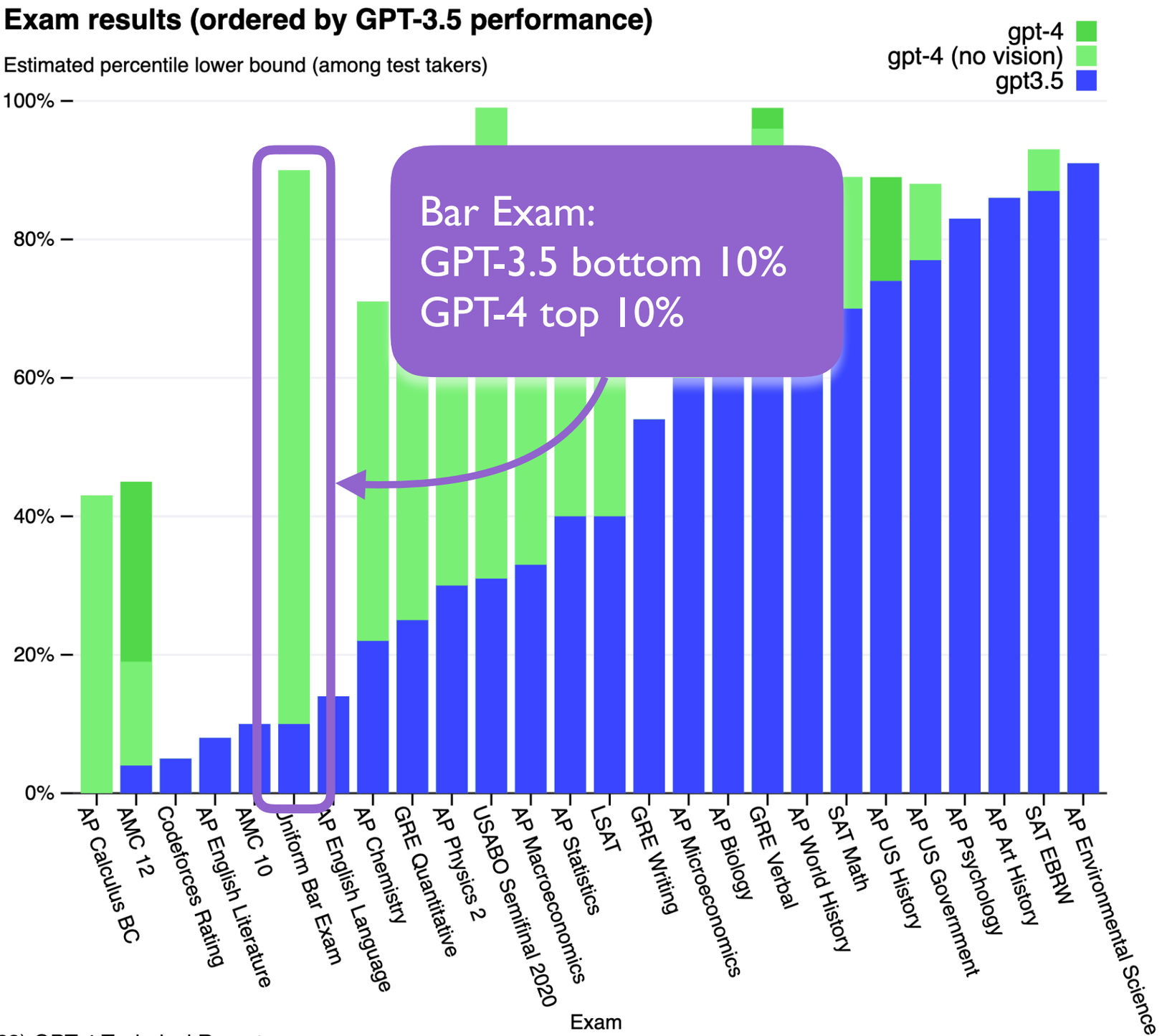
TurkuNLP

- Research group in CS / Linguistics
- Founded 2001 as Turku BioNLP group, now ~25 members
- Focus on ML-based NLP for scientific text mining and Finnish text processing (+recently large language models)



Motivation

Why transformers?

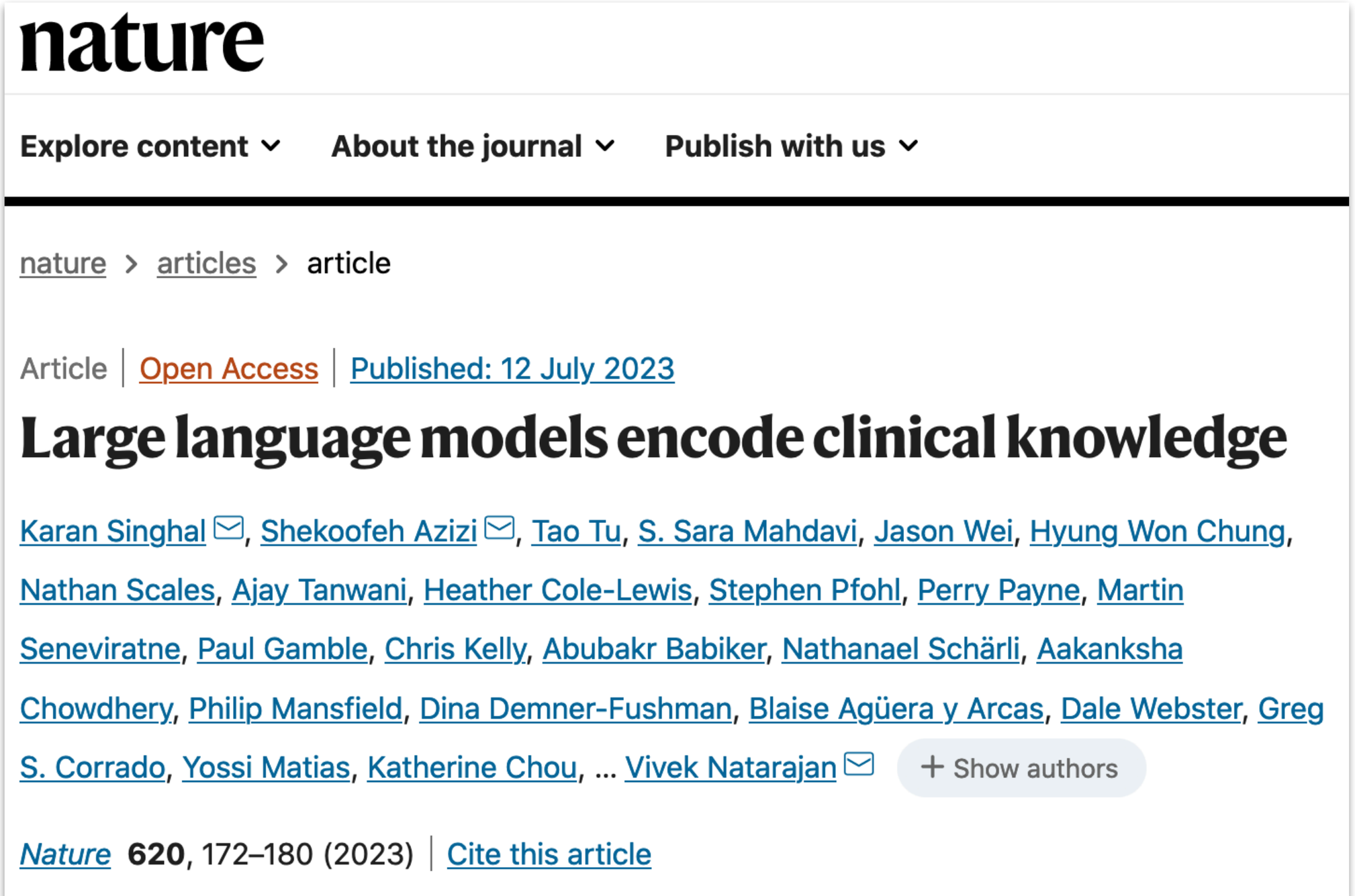


OpenAI (2023) GPT-4 Technical Report

Some headlines:

- OpenAI claims GPT can beat 90% of humans on the SAT (CNBC)
- 300 million jobs could be affected by latest wave of AI (CNN)
- Is GPT-4 already showing signs of artificial general intelligence? (New Scientist)

Why transformers?



https://www.nature.com/articles/s41586-023-06291-2

Why transformers?

GPT-4

Draft a scientific paper surveying the role of p53 in cancer.

The Role of p53 in Cancer: A Comprehensive Survey

Abstract

The p53 protein, commonly dubbed the “guardian of the genome,” plays an indispensable role in the regulation of the cell cycle and apoptosis. This paper provides a comprehensive survey on the function and significance of p53 in the context of cancer, emphasizing its role as a tumor suppressor, the implications of its mutations, and potential therapeutic interventions targeting the p53 pathway.

1. Introduction

p53, a product of the TP53 gene, is a pivotal transcription factor that regulates several cellular processes including DNA repair, cell cycle arrest, senescence, and apoptosis. When

Why transformers?

GPT-4

Explain the role of p53 in cancer using Shakespearean prose.

In the grand theatre of our body's cells,
There plays a sentinel, p53 by name.
With vigilant eye, it ever dwells,
To guard against those that seek ill fame.

When DNA, the script of life, is tainted,
By sun, by smoke, or errors unbidden,
This guardian, by Nature's hand painted,
Decides if the script should be mended or hidden.

"Repair!" it commands, to enzymes near,
Or "halt!" to the cell, in its life's progression.
And if the damage's too severe,
It orders the cell to make its own recession.

Yet, when p53 is itself besmirched,
Its role as sentinel gets fully usurped.
Cancer, that vile fiend, then goes unperched,
And uncontrolled growth leaves health usurped.

Why not transformers?

Why does salty liquorice cause cancer?

Researchers from the University of Michigan in Ann Arbor have isolated a compound from salty liquorice that can stimulate the growth of certain cancers. The substance acts as an inducer of a gene called E2F1, which is linked to the growth of cancer cells.

The E2F1 "driver gene" is activated in many cancers. This gene is a member of a family of transcription factors that controls the cell cycle, and its activation in cancer cells can lead to unlimited growth.

The salty liquorice compound, 2-(2-hydroxy-3-methoxypropyl)-5,4'-dihydroxy-7-methoxy-8-isopentenylflavanone, stimulates the growth of breast cancer cells in vitro.

"This compound is the first in a new class of naturally occurring compounds identified as inducers of E2F1," says Dr. William J. DePinho, Ph.D.

(this is pure hallucination; the unnamed model is quite small)

Why not transformers?

GPT-4

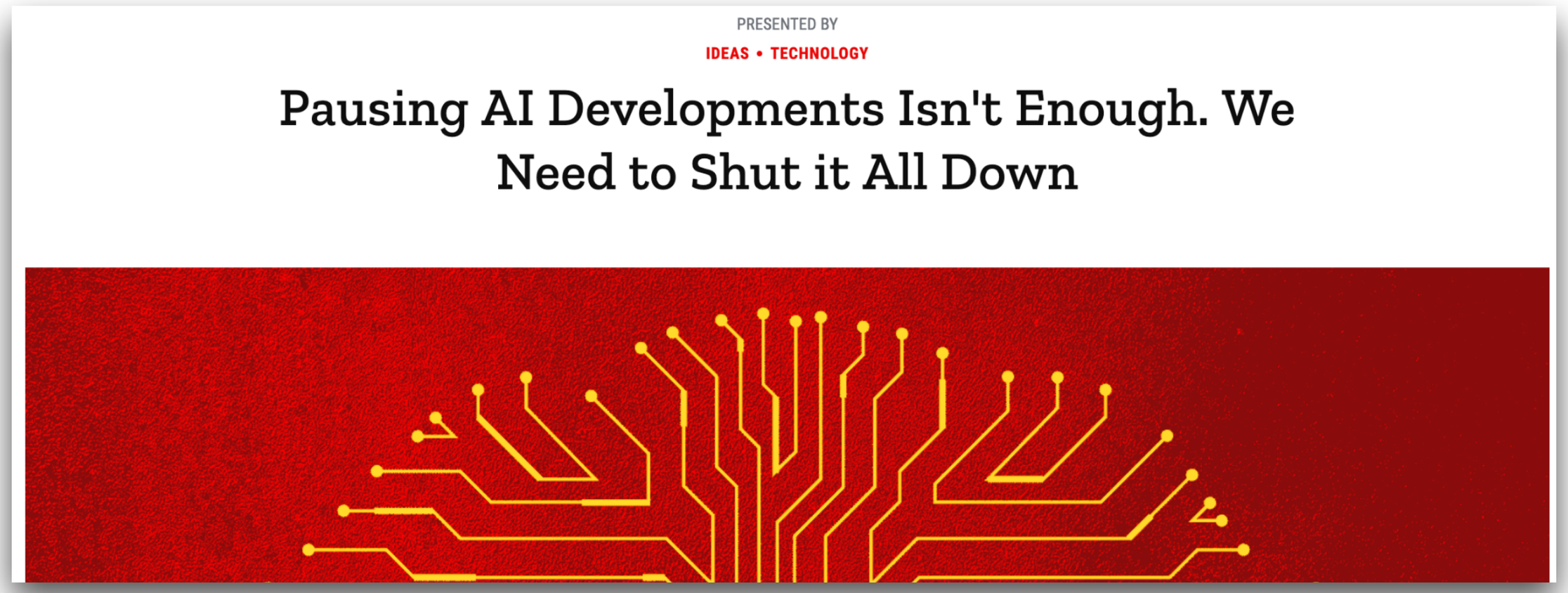
Why does salty liquorice cause cancer?

As of my knowledge cutoff in September 2021, there is no definitive scientific evidence to suggest that salty liquorice, also known as salmiak or ammonium chloride, directly causes cancer.

There may be confusion related to this because liquorice (or licorice) contains a compound called glycyrrhizin, which, when consumed in large quantities, can lead to health issues such as high blood pressure, low potassium levels, and edema (swelling). However, these potential health effects are different from causing cancer.

(hallucinations are less of an issue with state-of-the-art models)

Why not transformers?



TIME magazine (opinion): "... the most likely result of building a superhumanly smart AI, under anything remotely like the current circumstances, is that literally everyone on Earth will die."

https://time.com/6266923/ai-eliezer-yudkowsky-open-letter-not-enough/

Why transformers?

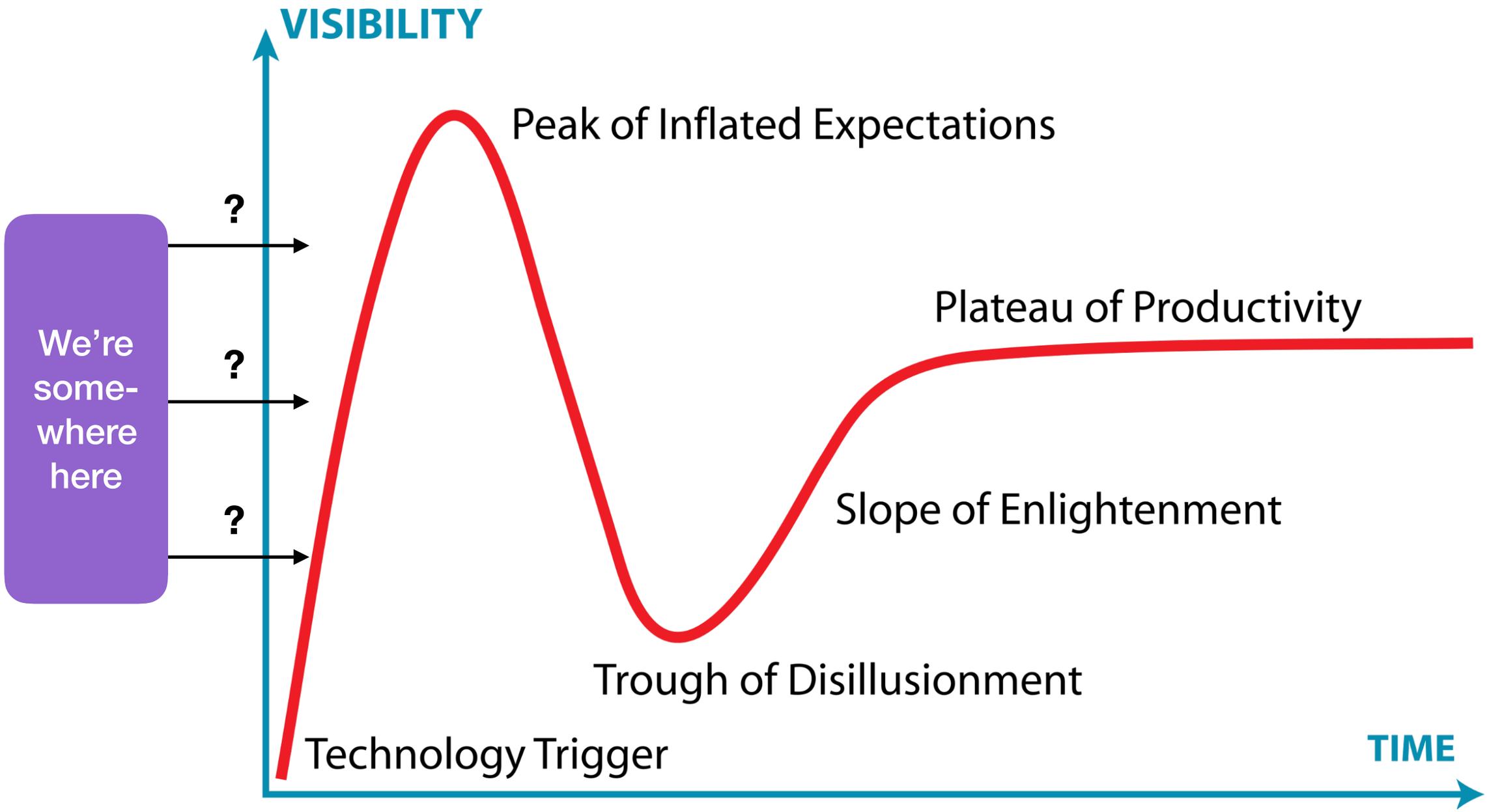


Figure from https://en.wikipedia.org/wiki/Gartner_hype_cycle

Introduction

Key concepts

Unpacking "**Text mining and Transformers**"

Text mining is a task in natural language processing (NLP) that builds on information extraction (another task in NLP), which includes subtasks such as named entity recognition, relation extraction, and normalization (or grounding)

Transformers are a neural network architecture used in deep learning, often combining supervised and unsupervised machine learning approaches to address NLP tasks using very large language models

Let's next briefly define those

Natural language processing (NLP)

NLP is a subfield of **artificial intelligence** (AI) that studies **computational methods** to process **human language**

Subdivisions:

- **Task**: analysis ("understanding") / generation
- **Modality**: text / speech
- **Methodology**: rule-based / statistical (incl. machine learning)

Here, focus on the **analysis of text using machine learning**

Information extraction

Information Extraction (IE) is the NLP task of extracting **structured information** from raw (**unstructured**) text

The goal is to transform text into a form that is easier to process automatically

Subtasks include **named entity recognition** (NER), **relation extraction** and mention **normalization / grounding**

The information to extract is domain- and task-dependent, e.g.

- Business intelligence: organization names, aqcuisition relations
- Biomedical text: protein names, binding relations

Text mining

Text mining refers broadly to the task of extracting high-quality information from (typically) large text resources

The term is sometimes used as a (near) synonym of information extraction; more broadly we can roughly define

text mining = information extraction + data mining

i.e. first convert unstructured text to structured information, then apply statistical methods to analyse it to discover useful patterns

Machine learning

Machine learning (ML) studies systems that improve at performing some task by "learning" from "experience" (data)

Frequently applied to tasks where **no explicit algorithmic solution** exists (or is e.g. computationally infeasible)

Categories:

- **Supervised learning**: learn from labeled data
- **Unsupervised learning**: find patterns in unlabeled data
- **Reinforcement learning**: learn from feedback

Here, focus on supervised and unsupervised ML (+variations)

Supervised learning

Machine learning from data where **inputs** (e.g. texts) are associated with desired **outputs** (e.g. labels)

Training requires (input, output) pairs, which in NLP tasks commonly come from a (manually) **annotated** text collection

→ Creating data has potentially **high cost** due to human effort

Categories:

- **Classification**: outputs represent predefined categories
- **Regression**: outputs are continuous values

Here focus on classification tasks

Unsupervised learning

Learning from raw data **without explicit outputs** (e.g. collection of unannotated text)

No need for manual annotation

→ **Very large** training datasets frequently available at **no cost**

Examples:

- **Clustering**: given a similarity metric for data points, arrange data into groups based on similarity
- **Language modeling** (NLP): given a large collection of text, learn to predict probability of particular word sequences

Neural networks

(Artificial) **neural networks** (NNs) are a class of ML models loosely inspired by biological neural networks (e.g. human brain)

NNs consist of **layers** of **nodes** ("neurons") that perform calculations parameterized by learned **weights** to map inputs into outputs

NNs (+loss functions) are **differentiable** and their weights can be optimized through gradient descent

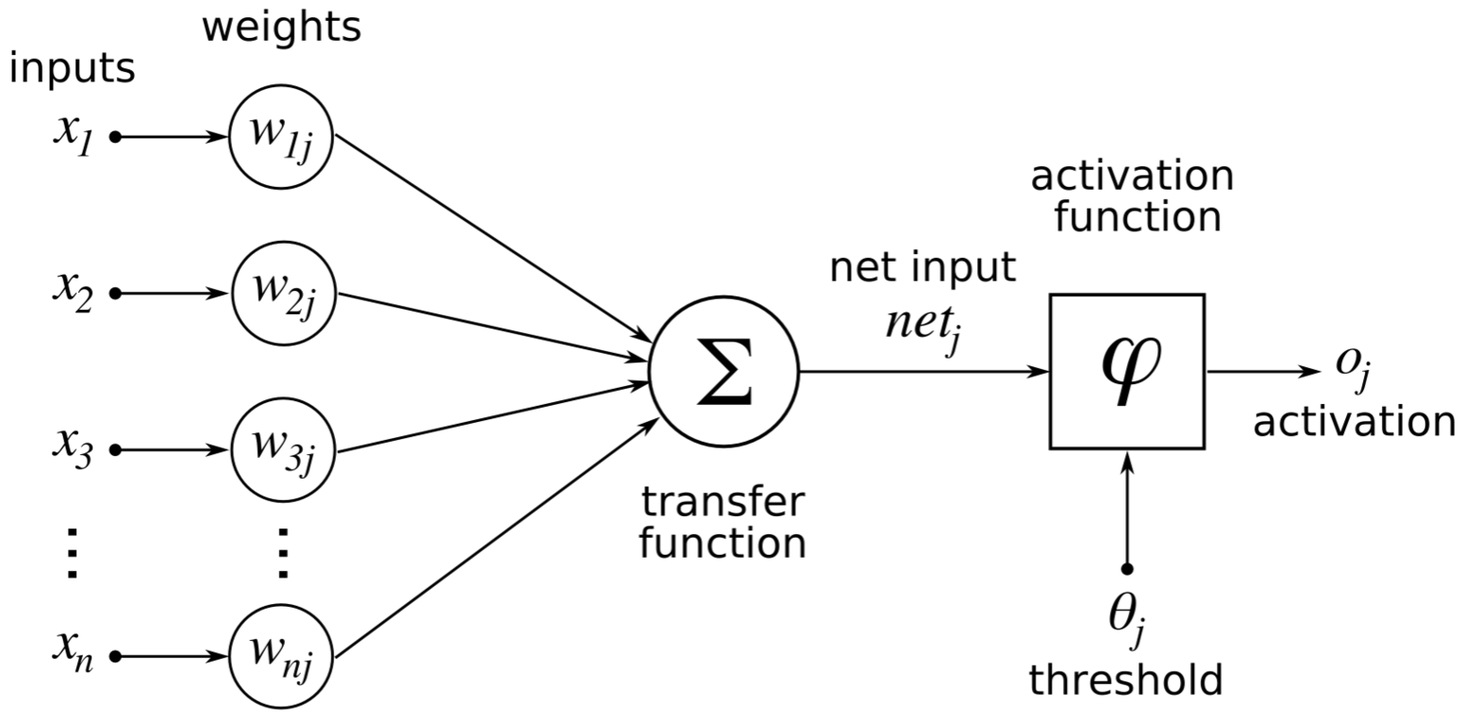


Figure from https://commons.wikimedia.org/wiki/Artificial_neural_network

Neural networks

A common NN architecture consists of an **input layer**, one or more **hidden layers**, and an **output layer**

- **Input layer** neurons represent features of input data
- **Output layer** neurons represent the result of the NN calculation, e.g. a class label like "positive" / "negative"
- **Hidden layers** are intermediate stages of the calculation, progressively transforming input into output

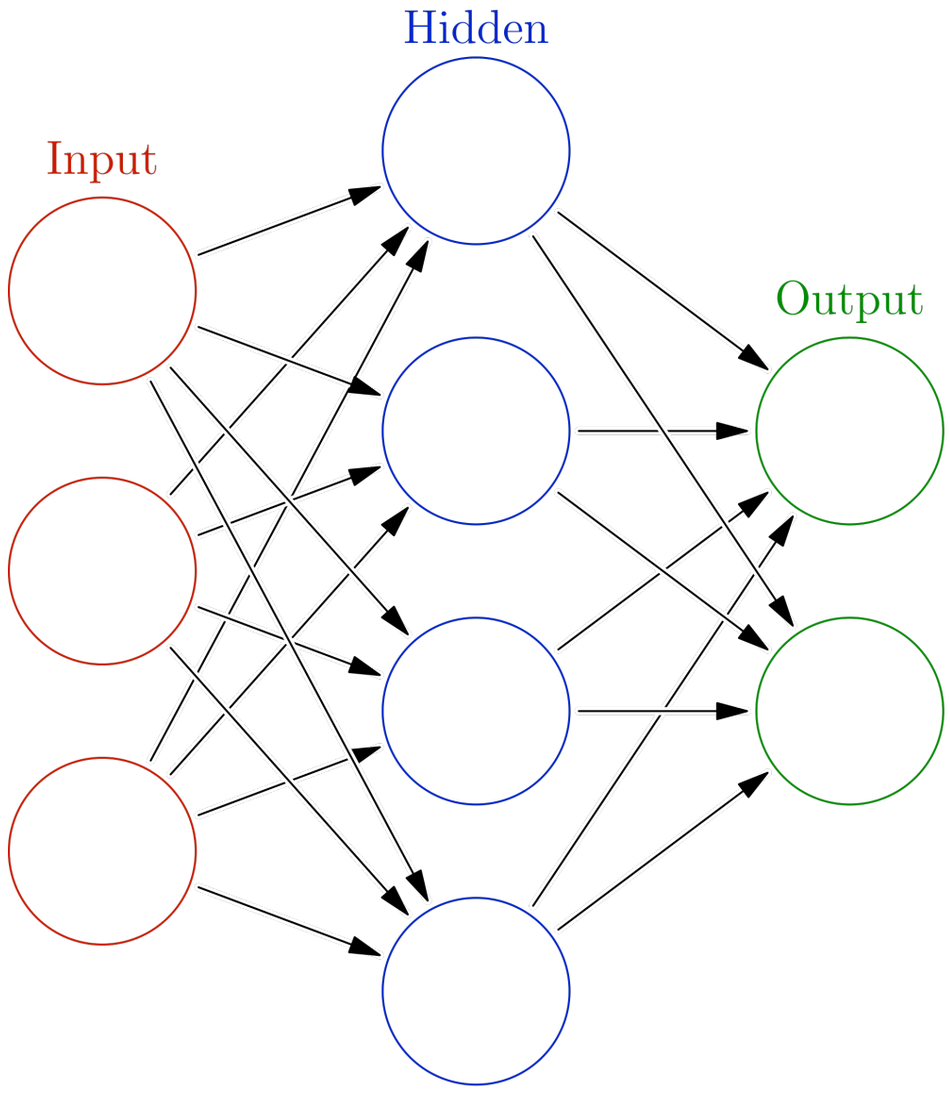


Figure from https://commons.wikimedia.org/wiki/Artificial_neural_network

Deep learning

Deep learning refers to machine learning using neural networks with a large number of layers

(here "deep" just refers to the depth of the NN layer stack)

Frequently associated with **very large models** and **very large datasets**, which are key to many recent successes

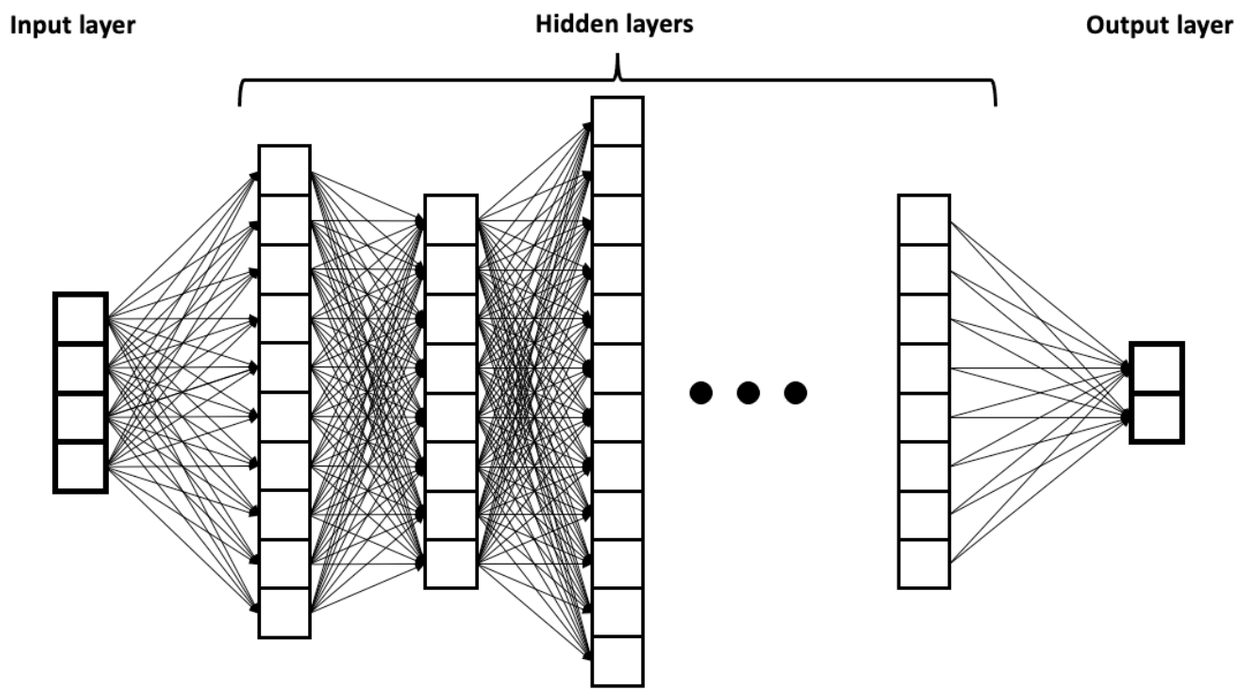


Figure from https://commons.wikimedia.org/wiki/Artificial_neural_network

Transformer

Deep NN architecture introduced in 2007 by Vasvani et al.

Originally proposed as a model for **machine translation**, applied very broadly to tasks in NLP and other domains since

Used in language models such as **GPT**, **BERT**, and **T5**, as well as in very large number of state-of-the-art systems for various tasks spanning text, speech, images, biological sequences, etc.

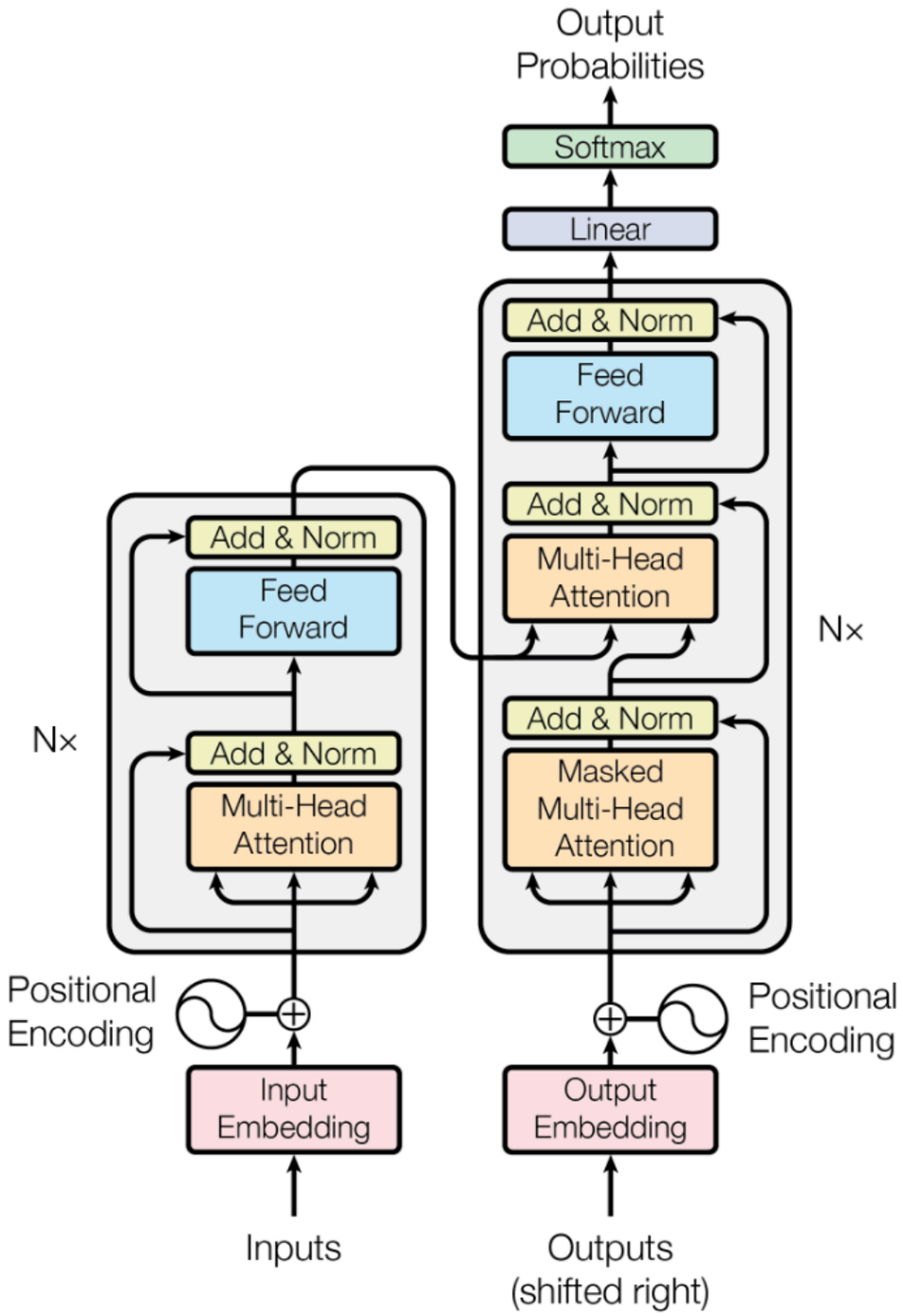
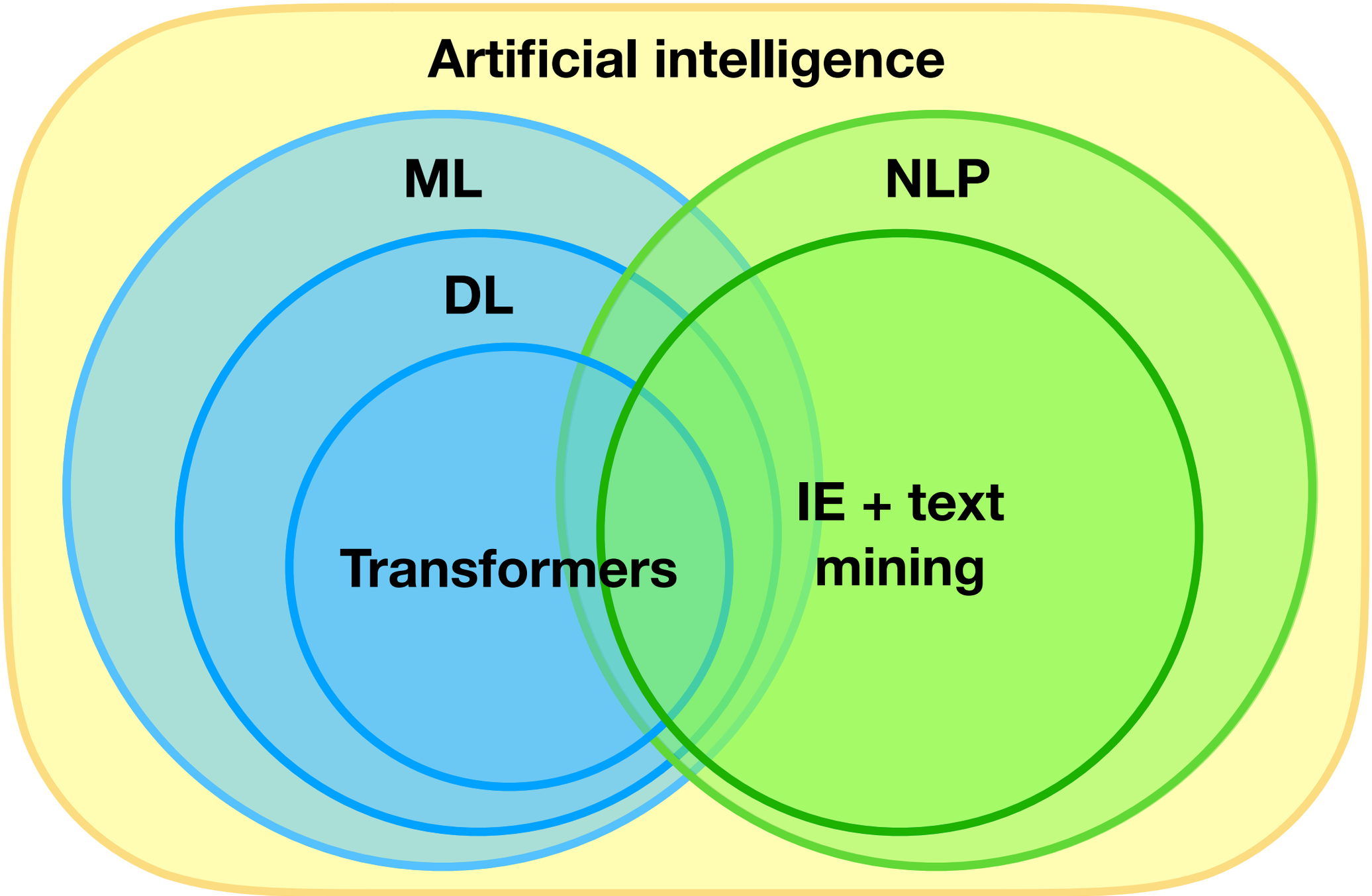
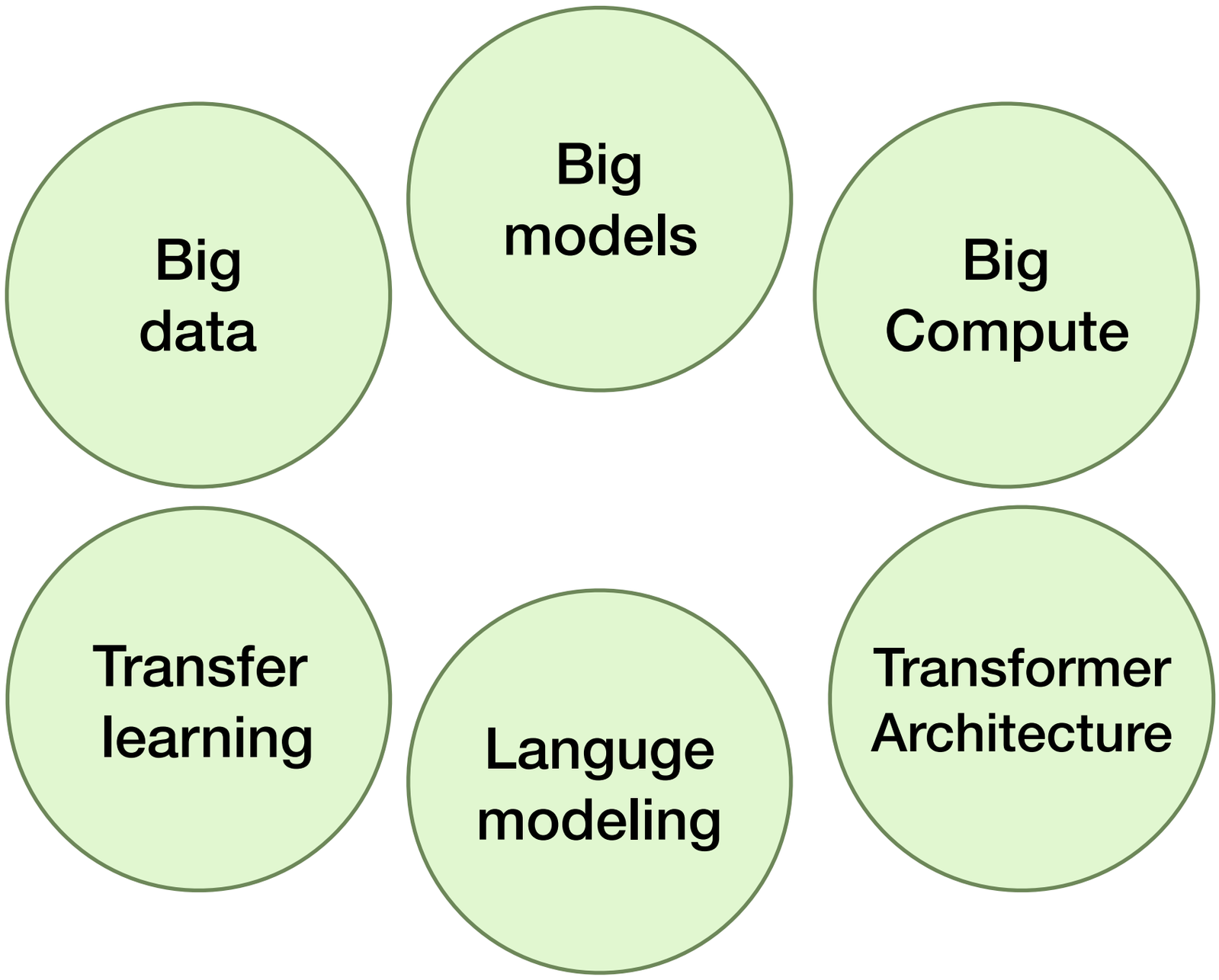


Figure from Vasvani et al. (2017) "Attention Is All You Need"


Artificial intelligence
ML
DL
Transformers
NLP
IE + text
mining

A winning combination

Key elements behind the recent successes of models such as GPT



Transfer learning

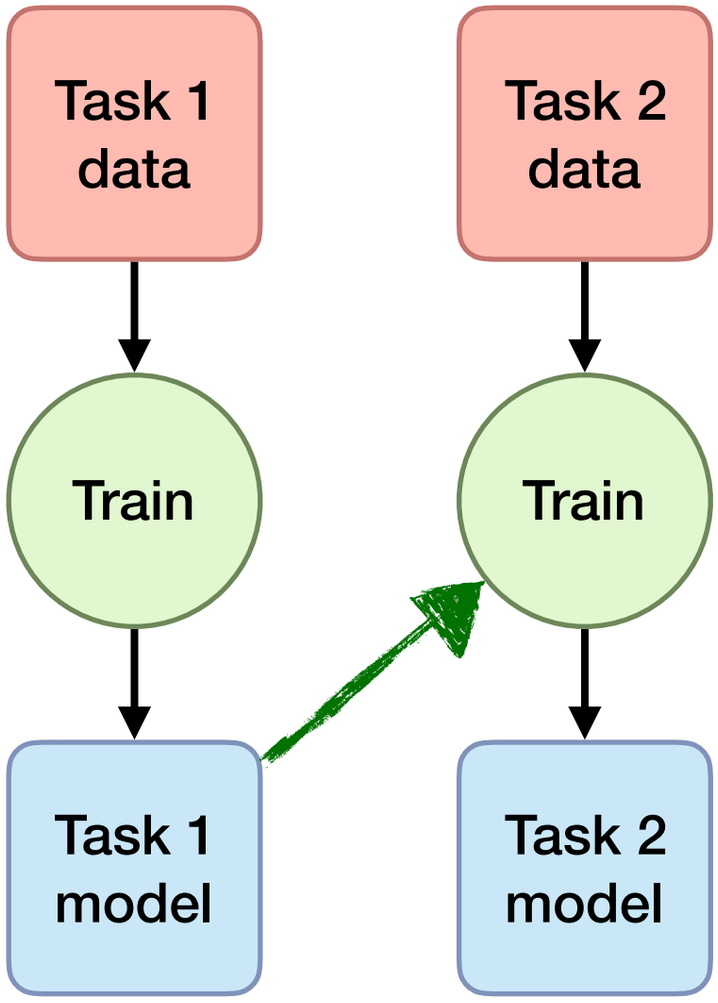
Transfer learning

Models can be trained "from scratch": at the start, the model knows nothing

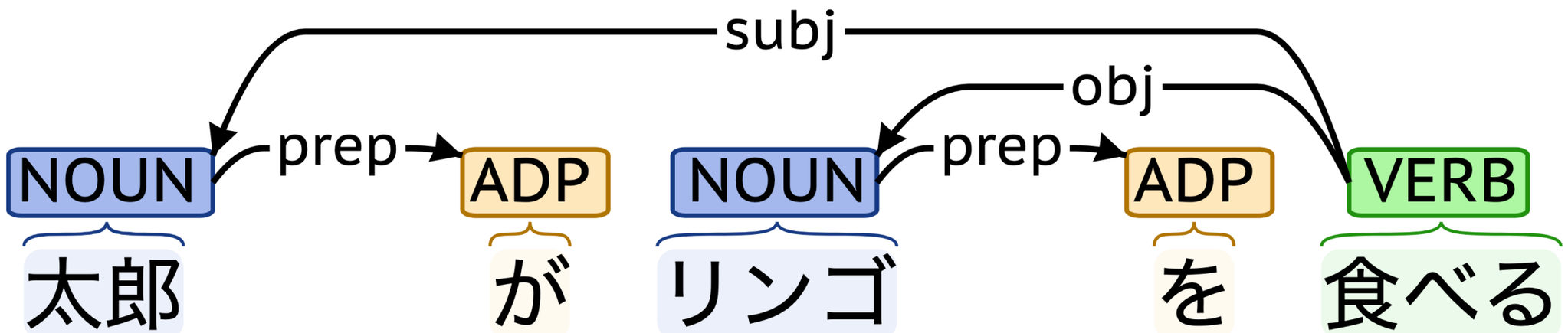
(e.g. NN weights initialized randomly)

For related tasks, knowing one can support learning another

In **transfer learning**, knowledge from one task is used when learning another



Transfer learning



To learn NLP tasks, it helps if you know the language first

e.g. easier to learn to annotate syntax for a language you know

“know a language” ~ “have a good language model”

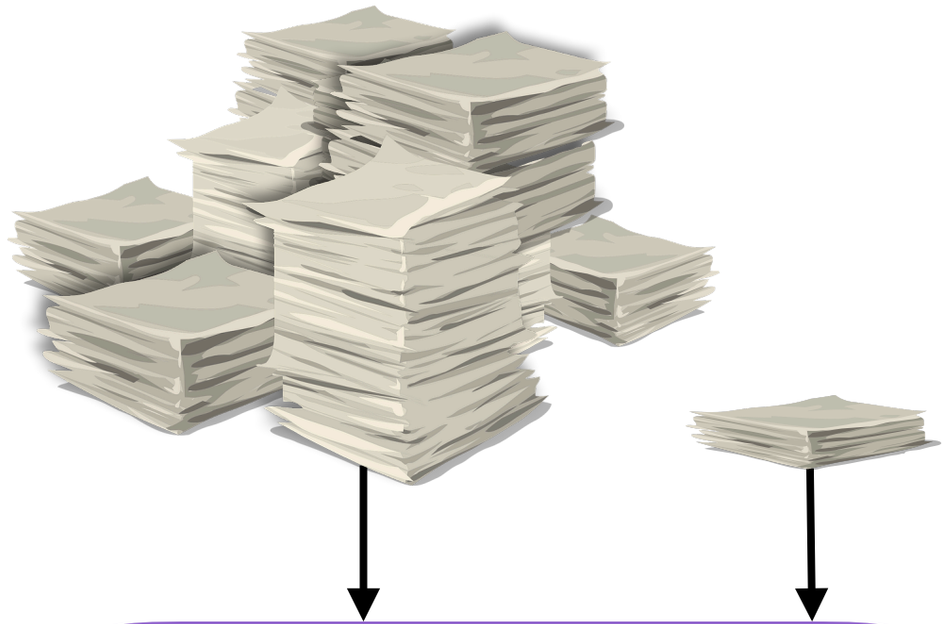
Transfer learning

For NLP, transfer from *unsupervised* tasks is particularly appealing:

Billions of words of **raw text** readily available on the internet and in resources such as PubMed

Annotated corpora are comparatively small and expensive to create

Deep learning methods are data-hungry



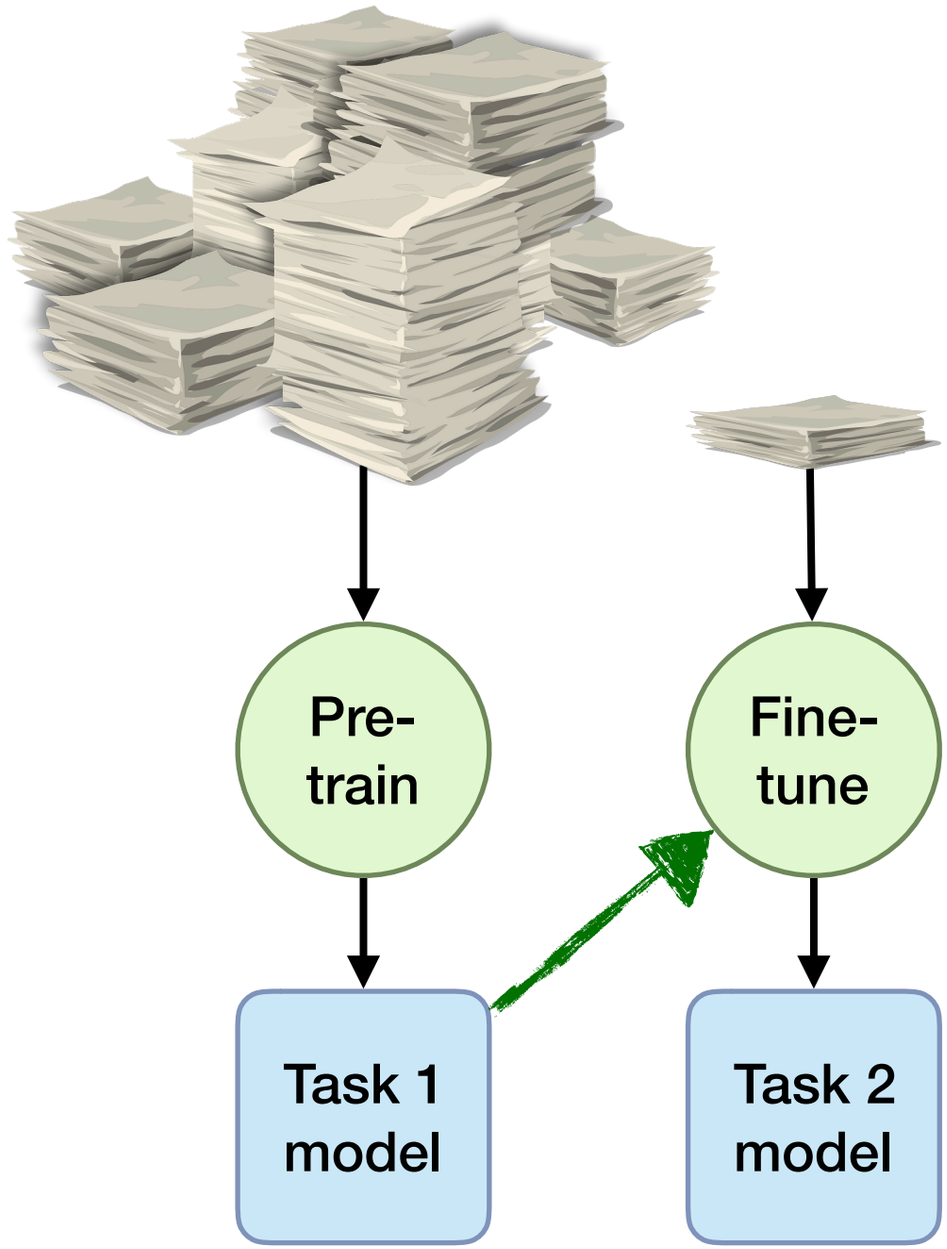
In numbers: for a manually annotated corpus, a **million** words is a respectable size, but LMs are now frequently trained on **trillions** of words (1,000,000 x larger!)

Transfer learning

Weight initialization is a straightforward way to implement transfer learning for NN models

Pre-training: train language model using large collection of raw text

Fine-tuning: initialize NN weights with pre-trained model, continue training on task-specific data



Language modeling

Language modeling

A **language model** (LM) is a machine learning model of (typically) human language

Traditionally used e.g. in spell checking and speech recognition, recently proposed as the solution to all of AI

Language modeling task setting:

- **Input**: large collection of raw (unannotated) text
- **Output**: model that can estimate how likely is a word sequence is in the language $P(w_1, w_2, \ldots, w_n)$

Major approaches by methodology:

- **"Traditional"**: count word sequences to estimate probability
- **Neural**: learn a neural network model to predict probability

Language modeling

The probability of a word sequence $P(w_1, w_2, \ldots, w_n)$ can be estimated via probabilities of the individual words in their context

LMs can be grouped by how that context is defined:

- **Causal LMs**: estimate probability of word given previous words only (one-directional, "left-to-right")
- **Bidirectional LMs**: estimate probability of word given both preceding and following words

Broadly speaking, causal LMs are particularly effective in **generation tasks** and bidirectional LMs in **classification tasks**

Language modeling

Using a causal language modeling approach, we can estimate the probability of a word sequence $w_1, w_2, \ldots, w_n$ as

$$P(w_1, w_2, \ldots, w_n) = P(w_1)P(w_2, \ldots, w_n \mid w_1) \text{ (etc.)}$$

Conversely, we can **generate** text from a causal language model by repeatedly sampling a word from the probability distribution conditioned by preceding words

(This is all that's happening in "discussions" with large language models: the model is repeatedly used to select a likely next word)

Count-based LMs

Count-based modeling is a simple statistical approach to LMs:

- **Given** sequence of words $(w_1, w_2, \ldots)$ representing large text collection
- **Count** occurrences of all subsequences of words in collection $C(w_1, w_2, \ldots, w_n)$
- **Estimate** probabilities $P(w_n \mid w_1, w_2, \ldots, w_{n-1})$ using counts C

(notation: $w_1, w_2, \ldots, w_n = w_{1:n}$ for short)

$$P(w_n \mid w_{1:n-1}) = C(w_{1:n}) / C(w_{1:n-1})$$

Data sparsity

There is an obvious issue with naively applying the probability estimate

$$P(w_n \mid w_{1:n-1}) = C(w_{1:n}) / C(w_{1:n-1})$$

Namely, for almost every longer sequence of words, the **counts will be zero**

Key challenge for count-based LMs is **data sparsity**: effectively all interesting non-trivial texts will be entirely novel

Example: what is the most likely next word in

"*We find that p53 promotion of apoptosis is regulated by …*"

N-gram language models

Problem: counts of almost all possible word sequences in a finite collection of text are zero

Solution: when estimating probabilities, only consider the previous *N* words instead of all previous words

- **Bigram** model: $P(w_n \mid w_{1:n-1}) \approx P(w_n \mid w_{n-1})$
- **Trigram** model: $P(w_n \mid w_{1:n-1}) \approx P(w_n \mid w_{n-2:n-1})$
- **4-gram** model: $P(w_n \mid w_{1:n-1}) \approx P(w_n \mid w_{n-3:n-1})$

As *N* is increased, models become better at estimating the next word, but data sparsity challenges increase

Limitations of N-gram LMs

N-gram models have been a key tool in NLP for decades, but have severe limitations

- **Limited use of context** due to short history (N) and unidirectionality
- **Data sparsity**: difficulty estimating rare N-grams
- **High resource usage** due to storage of large N-gram tables
- **No word similarity**: cat != dog, cat != hat
- **Fixed vocabulary**, out-of-vocabulary (OOV) words

Prediction-based LMs

General approach:

- **Given** sequence of words $(w_1, w_2, \ldots)$ representing large text collection
- **Learn** model that can predict probability of word given previous words $P(w_n \mid w_1, w_2, \ldots, w_{n-1})$

Basically any method capable of learning to predict a probability distribution is applicable, but in practice focus on **neural network models**

Notable pre-transformer methods include **word2vec** and **ELMO**

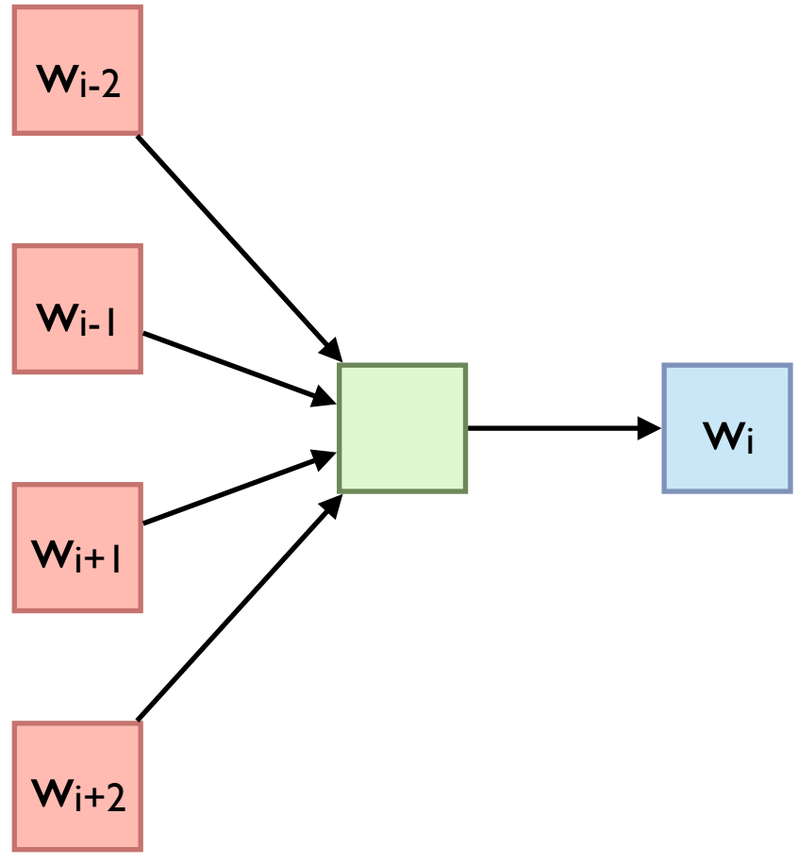
Word2vec

Shallow neural model for efficiently learning dense vector representations (**embeddings**) of words

Continuous bag-of-words: train model to predict word given preceding and following words $P(w_i \mid w_{i-2}, w_{i-1}, \ldots, w_{i+1}, w_{i+2}, \ldots)$

Demonstrated capacity to resolve analogies, e.g. "king" - "man" + "woman" = "queen"

Limited by **lack of context sensitivity**: each word associated with single embedding regardless of context (consider e.g. "pupil")



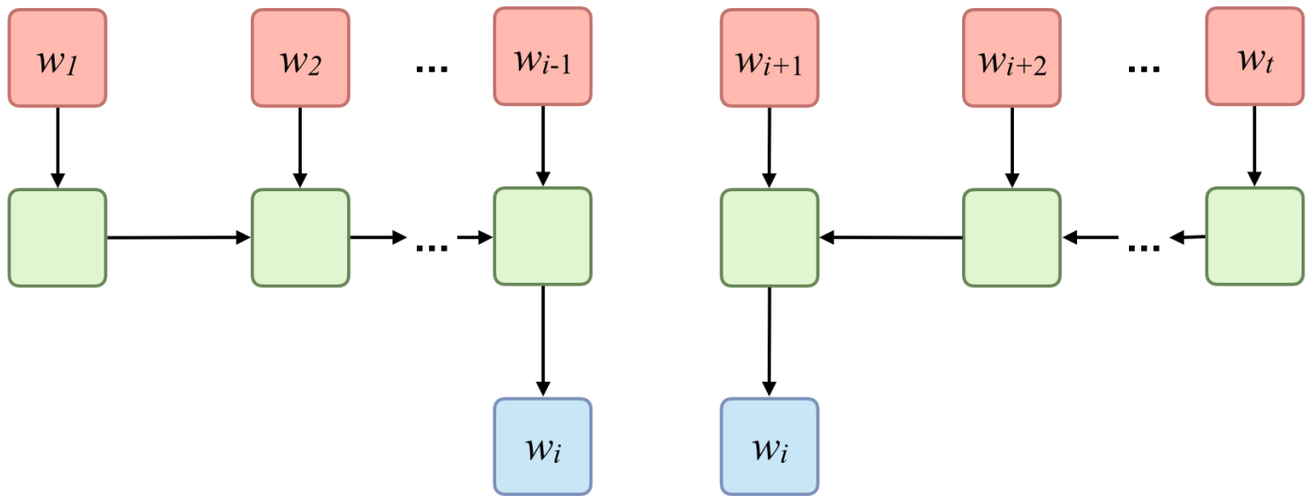
Mikolov *et al*. (2013) *Efficient Estimation of Word Representations in Vector Space*

Embeddings from Language Models (ELMO)

Deep language model using bidirectional **recurrent neural networks** (RNNs) pre-trained on 1B words of English

Contextualized embeddings: the vector representation for a word depends on surrounding words

In combination with supervised NLP tasks, advanced SOTA in tasks including question answering, NER and sentiment analysis



Peters *et al*. (2018) *Deep contextualized word representations*

Embeddings from Language Models (ELMO)

Contextualized representations can differentiate between word senses

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {...}	{...} they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

Peters *et al*. (2018) *Deep contextualized word representations*

Early neural LMs

Early neural LMs resolved a number of issues with N-gram language models, including

- **Improved use of context**: For N-grams, typically N < 10, RNNs have unlimited context in principle and hundreds of words in practice
- **Word similarity**: simple operations (e.g. dot product) on dense learned word embeddings correspond intuitively to word similarity e.g. sim("cat","dog") > sim("cat","hat")

… but challenges remain e.g. in using long contexts (RNNs have limited "memory") and scaling (very large models and data)

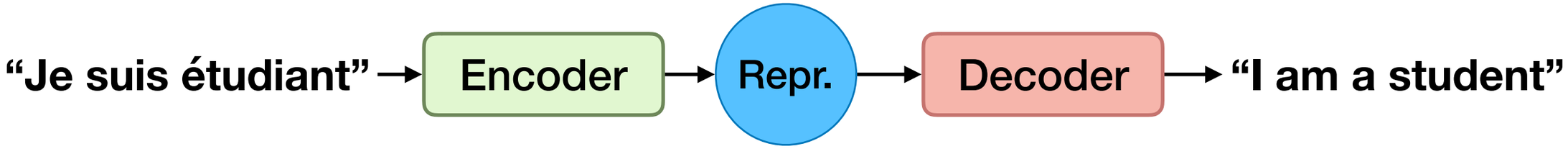
Transformers

Transformer architecture

Transformer model originally proposed for machine translation

Encoder-decoder architecture:

- **Encoder** computes a representation of the meaning of input words in context
- **Decoder** generates output conditioned on encoded representation and previously generated output



Transformer architecture

Top-level view of original transformer architecture

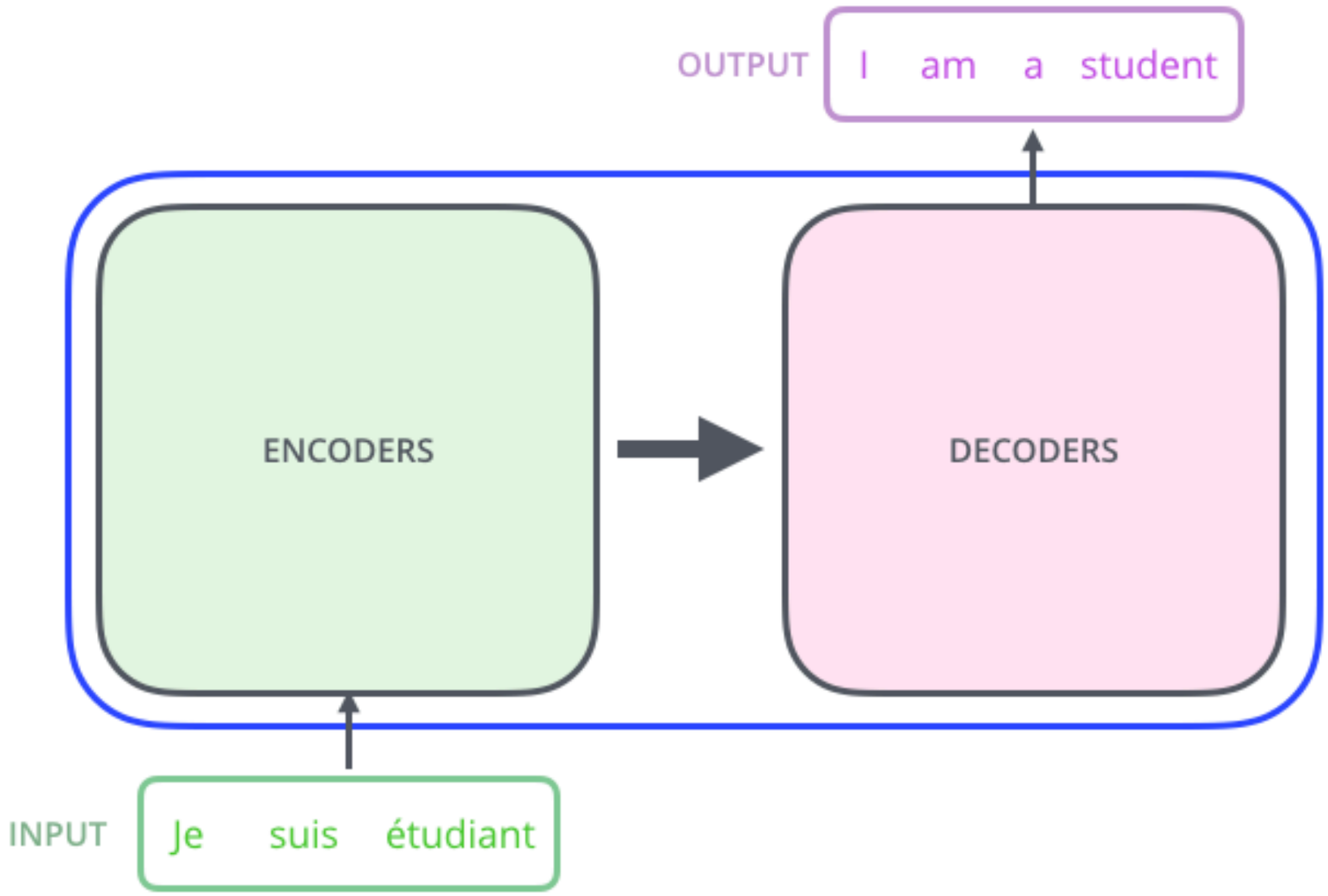


Figure from http://jalammar.github.io/illustrated-transformer/

Transformer architecture

Encoder and decoder consist of layers of identical blocks

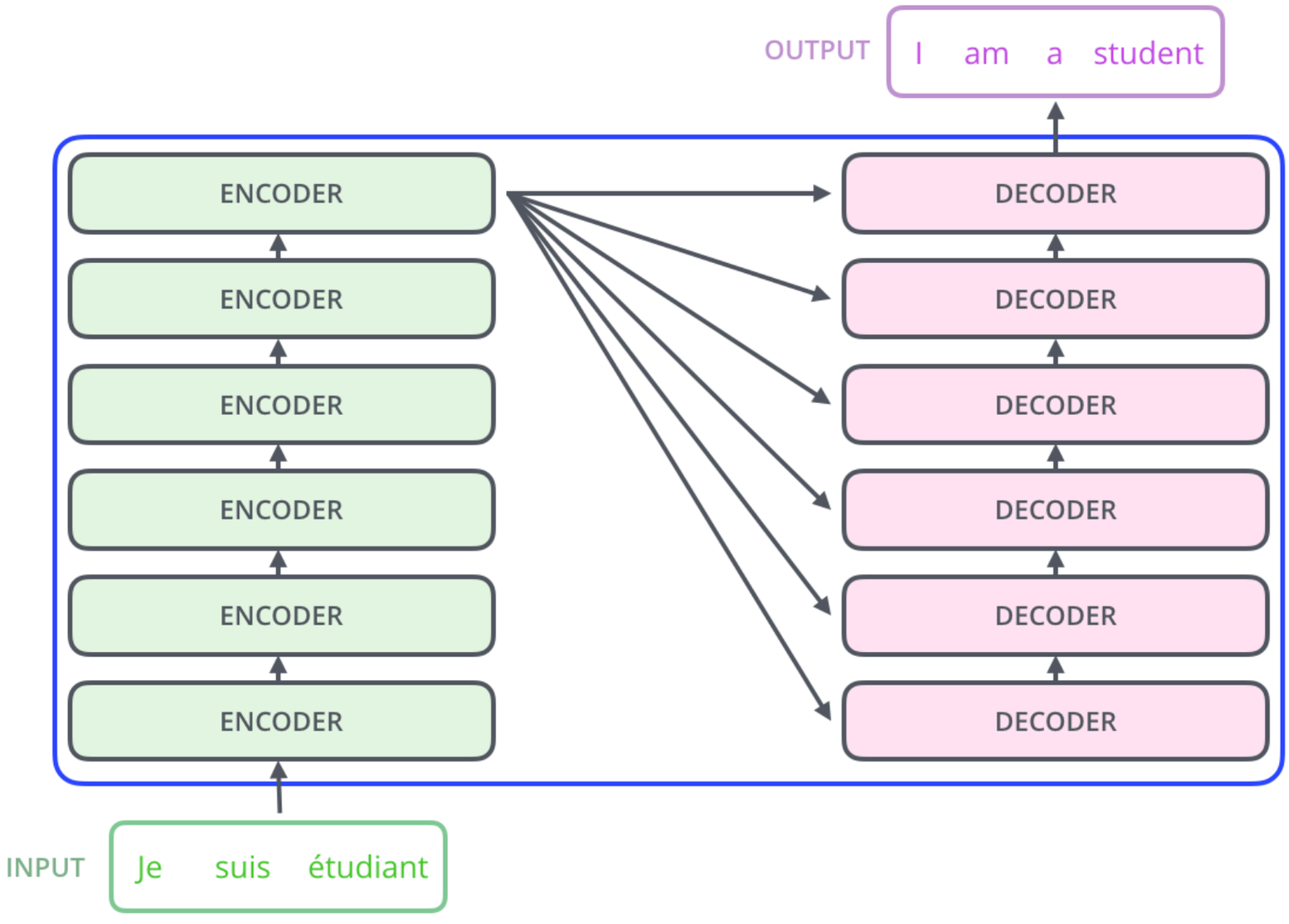


Figure from http://jalammar.github.io/illustrated-transformer/

Transformer architecture

Transformer architecture detail

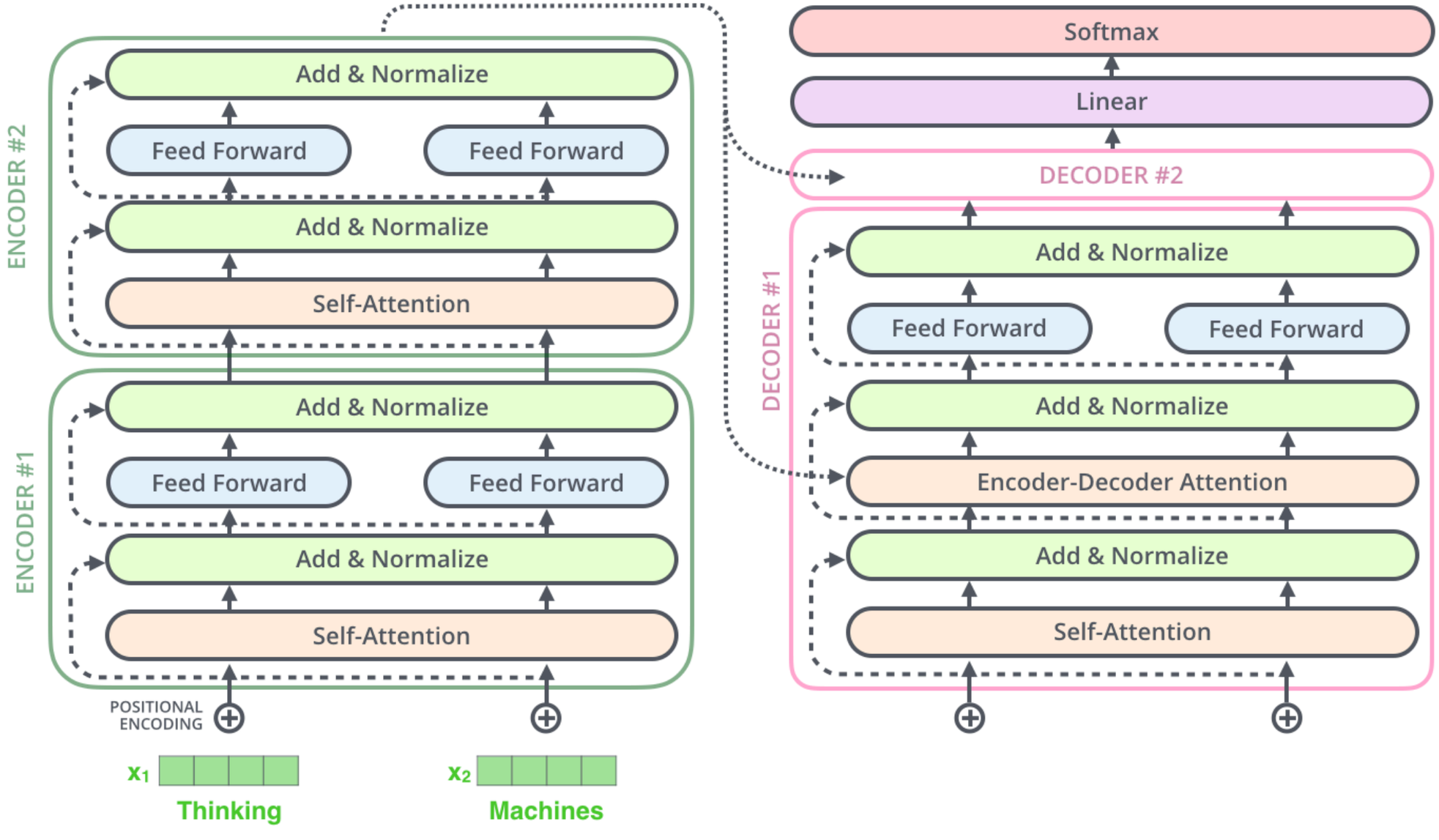


Figure from http://jalammar.github.io/illustrated-transformer/

Transformer architecture

Each encoder and decoder block performs a **self-attention** operation to access information from context and a small standard fully connected NN to compute the next layer representation

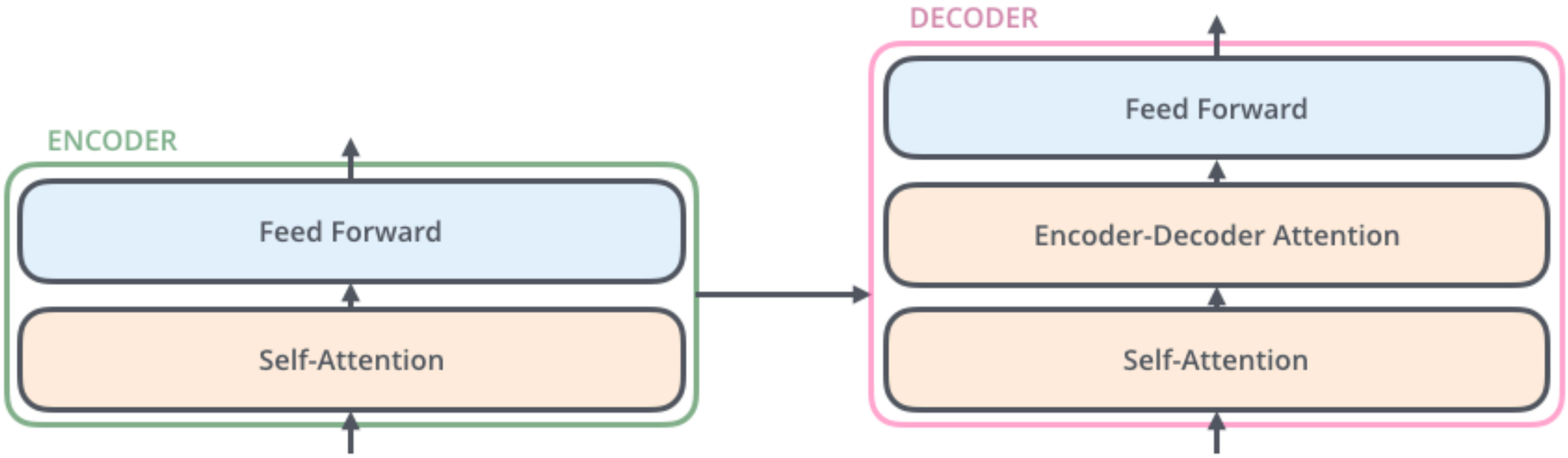


Figure from http://jalammar.github.io/illustrated-transformer/

Self-attention

A key contribution of the transformer model is the use of a **self-attention** mechanism:

- In each encoder/decoder layer, the relevance of each word to each other word in context is computed (*quadratic*!)
- Information from relevant context words is then added to the representation of each word

By contrast to LMs using convolutional (CNN) and recurrent (RNN) architectures, the computation required to pass information from one word representation to another is constant

→ Transformers "remember" everything in the context

Vasvani et al. (2017) "Attention Is All You Need"

Self-attention

The self-attention mechanism is both one of the greatest strengths and a fundamental limitation of the transformer model

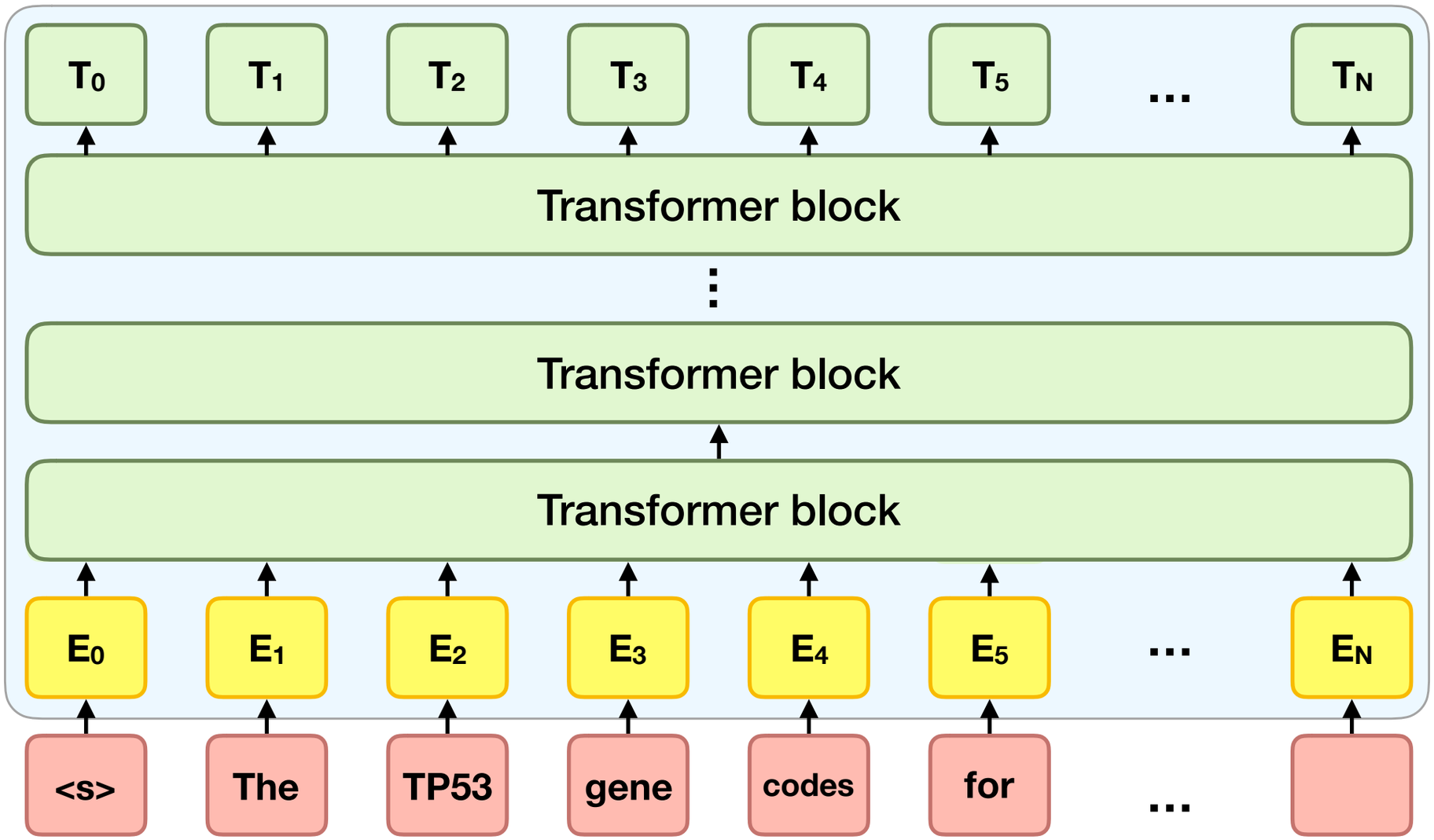
- Allows models to efficiently use consirerably **longer contexts** (input/output text) than previous models, up to **tens of thousands of words** in some recent models
- The basic attention mechanism is **quadratic in space and computational requirements**, limiting the maximum practically usable context length

(Relieving this bottleneck is a major focus of current research)

Inputs and outputs

Input words are mapped to embeddings that are progressively transformed to produce contextual representations

For task-specific fine-tuning, **output** layers typically added on top



Classes of transformer models

The original **encoder-decoder** transformer architecture can be applied to any **text-to-text** task, not just machine translation

The **encoder** and **decoder** components of the original transformer architecture can be used **separately**

Three broad classes of transformer models:

- **Encoder-only** (e.g. BERT): contextualized representations of input words
- **Decoder-only** (e.g. GPT): generates text conditioned on previous words → causal language model
- **Encoder-decoder** (e.g. T5): general text-to-text mapping

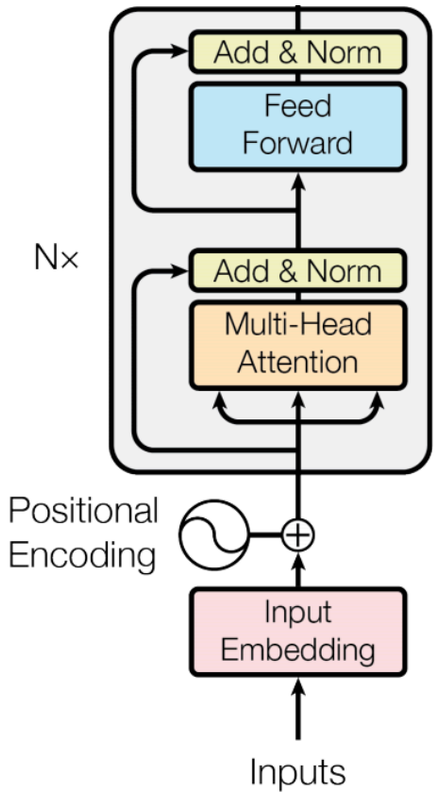
Bidirectional Encoder Representations from Transformers (BERT)

Encoder-only model introduced by Devlin et al. (Google) in 2019

English model pretrained on 3B words (Wikipedia + BooksCorpus)

Model sizes: 110M parameters (base) and 340M parameters (large)

Fine-tuning for various language understanding tasks showed results surpassing human performance



Devlin *et al*. (2019) *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

Generative Pre-trained Transformer (GPT)

Decoder-only models first introduced by Radford et al. (OpenAI) in 2018

GPT-1 pre-trained on ~1B word BooksCorpus (English)

Fine-tuning for supervised NLP tasks, advanced SOTA

Followed up by **GPT-2**, **GPT-3**, **InstructGPT**, **ChatGPT**, and **GPT-4** models

Model sizes: 117M (GPT-1) to 175B (GPT-3); GPT-4 unknown

(ChatGPT and GPT-4 are the media darlings of the family)

Radford *et al*. (2018) *Improving Language Understanding by Generative Pre-Training*

Generative Pre-trained Transformer (GPT)

Types of language model by training process

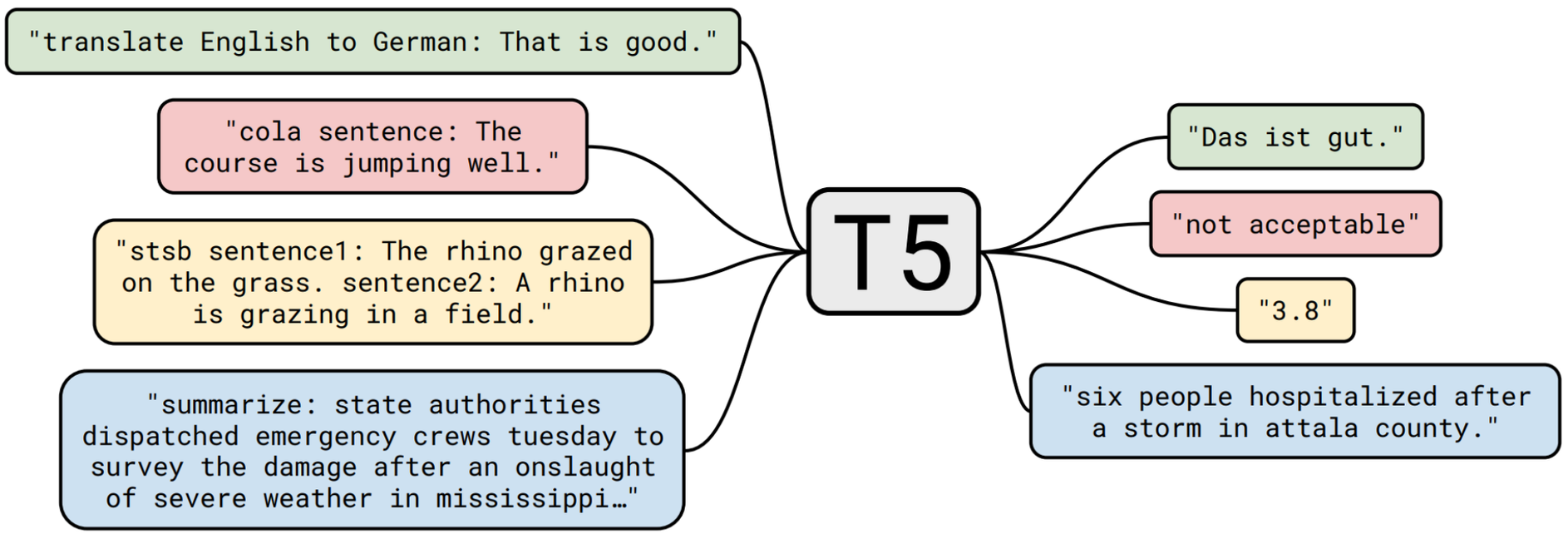
- **"Pure" language models** (e.g. GPT1-3): trained on "naturally occurring" texts such as books and web pages — poor "UI"
- **Instruction-tuned models** (e.g. InstructGPT): trained on texts with an instruction - context - response structure
- **Chat models** (e.g. ChatGPT, GPT-4): trained on texts with explicit dialogue structure

Text-to-Text Transfer Transformer (T5)

Encoder-decoder models introduced by Raffel et al. (2020)

Models ranging from **60M to 11B parameters** trained on 34B words

Key proposal: cast NLP tasks as text-to-text, train single model to perform all tasks



Raffel et al. (2020) “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”

Selecting a class of transformer

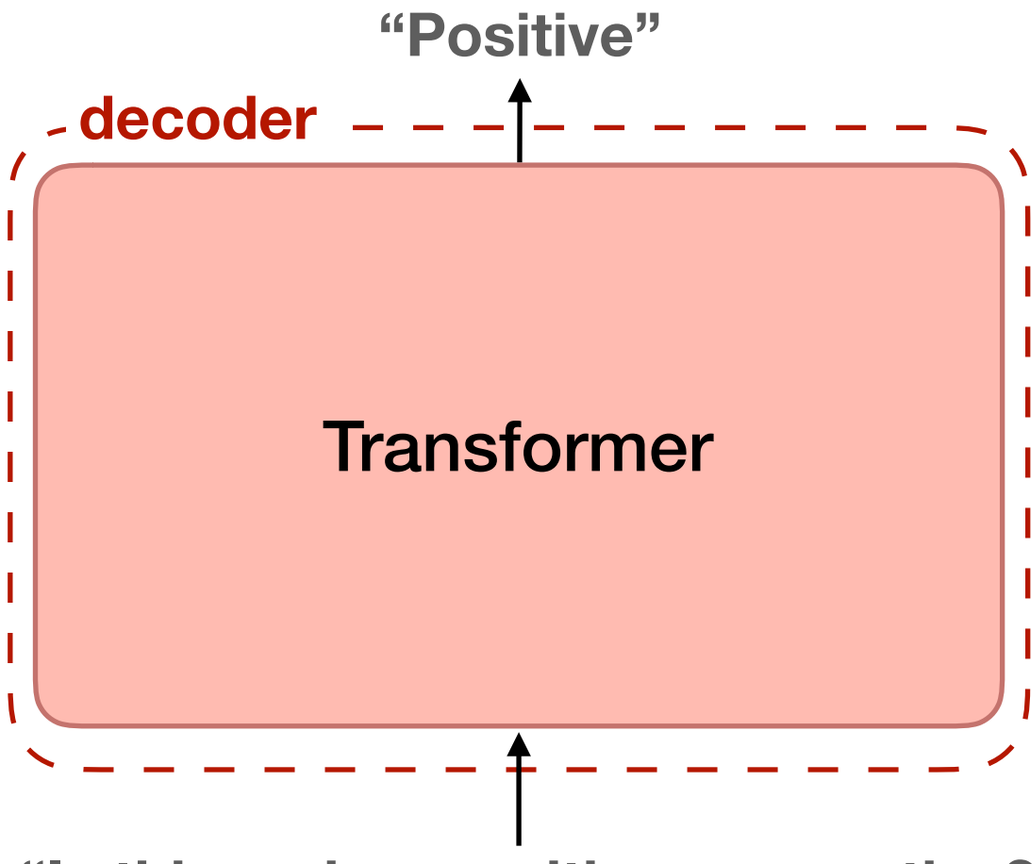
(Simplifying)

- **Encoder-only** ("BERT-like") models excel at **classification** tasks; larger models ~1B parameters (consumer hardware)
- **Decoder-only** ("GPT-like") models excel at **generation**; largest models 100s of billions of parameters (supercomputers)
- **Encoder-decoder** ("T5-like") models excel at **sequence-to-sequence** tasks; largest models 10s of billions of parameters

State-of-the-art encoder-only and encoder-decoder models often fine-tuned, the largest decoder-only models commonly used as-is in **zero- or few-shot settings** via **prompting**

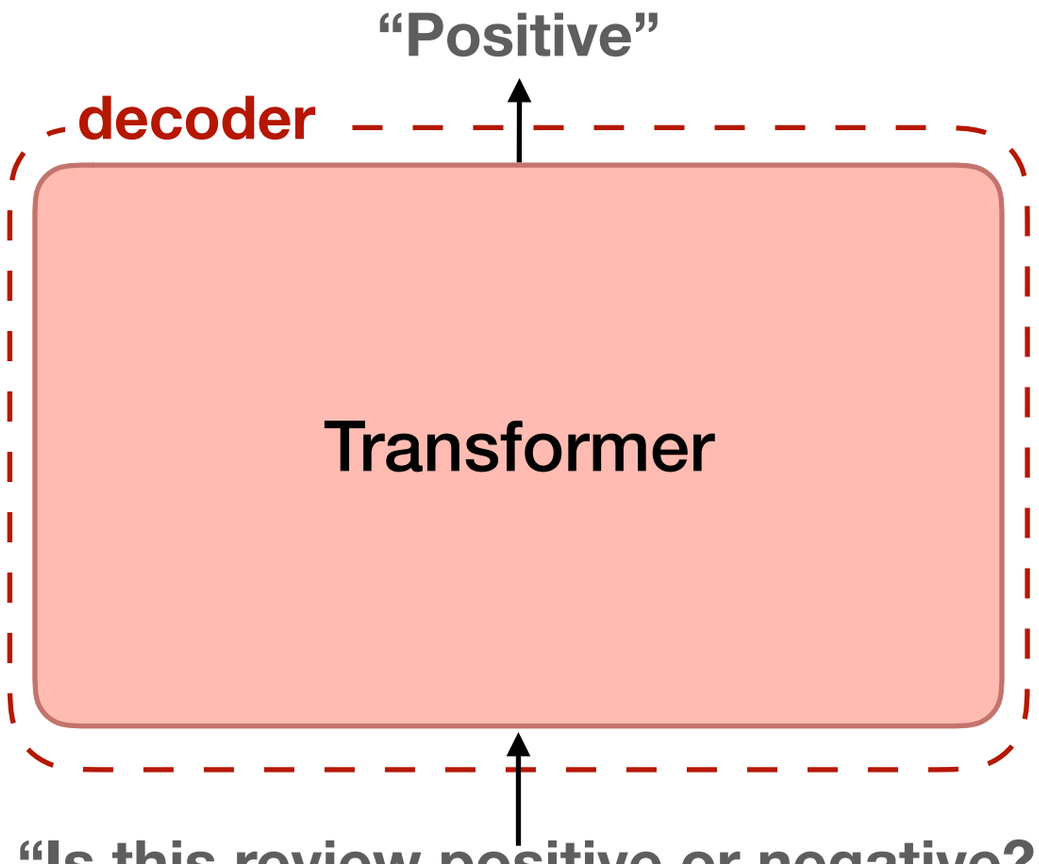
Zero- / few-shot prediction

Zero-shot classification



**“Is this review positive or negative?
Review: This is a very good movie.
Answer: ”**

One-shot classification



**“Is this review positive or negative?
Review: This movie sucks!
Answer: Negative**

**Is this review positive or negative?
Review: This is a very good movie.
Answer: ”**

Text mining with transformers

Approaches

All classes of transformer models can be applied to information extraction and text mining, for example:

- **Encoder-only**: cast tasks as classification, fine-tune separate models for each task
- **Decoder-only**: develop natural language prompts for task, generate using pre-trained instruction / chat model
- **Encoder-decoder**: cast task as text-to-text with document as input and structured data as output, fine-tune single model

We'll here focus on the first in the context of a "traditional" information extraction pipeline

Text mining pipeline

Source texts (e.g. PubMed)

Document selection

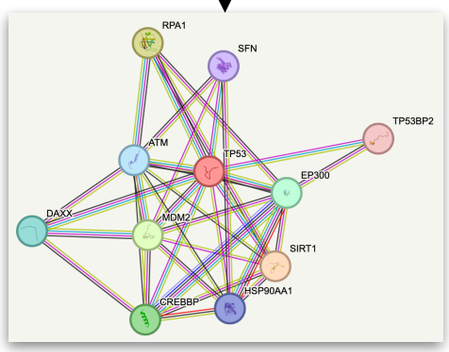
Named entity recognition

Normalization

Relation extraction

Structured data

Analysis



Useful information

Named entity recognition

Example: taxonomic name mention recognition

1	Primary structure of cytochrome c gene from the white root rot Kingdom fungus Species Rosellinia necatrix.
2	The nucleotide sequence of the cytochrome c (CytC) gene of the white root rot Kingdom fungus Species Rosellinia necatrix was analyzed.
3	The structure of this gene, which had three introns in the coding region, was similar to that of Species Aspergillus nidulans.
4	The second intron of the Species R. necatrix CytC gene was not present in Species Neurospora crassa or Species Fusarium oxysporum.
5	However, the amino acid sequence of Species R. necatrix was most similar to that of Species Neurospora crassa.
6	Thus, it seemed that the second intron of the Species R. necatrix CytC gene was inserted into its present position after Species R. necatrix and its closest relatives diverged evolutionarily.

Luoma et al. (2023) "S1000: A better taxonomic name corpus for biomedical information extraction"

Normalization

Example: taxonomic name normalization to NCBI Taxonomy

1 Primary structure of cytochrome c gene from the white root rot fungus Rosellinia necatrix.

2 The nucleotide sequence of the cytochrome c (CytC) gene of the white root rot fungus Rosellinia necatrix was analyzed.

3 The structure of this gene, which had three introns in the coding region, was similar to that of Aspergillus nidulans.

4 The second intron of the R. necatrix CytC gene was not present in Neurospora crassa or Fusarium oxysporum.

5 However, the amino acid sequence of R. necatrix was most similar to that of Neurospora crassa.

6 Thus, it seemed that the second intron of the R. necatrix CytC gene was inserted into its present position after R. necatrix and its closest relatives diverged evolutionarily.

Species "R. necatrix" ID:T4

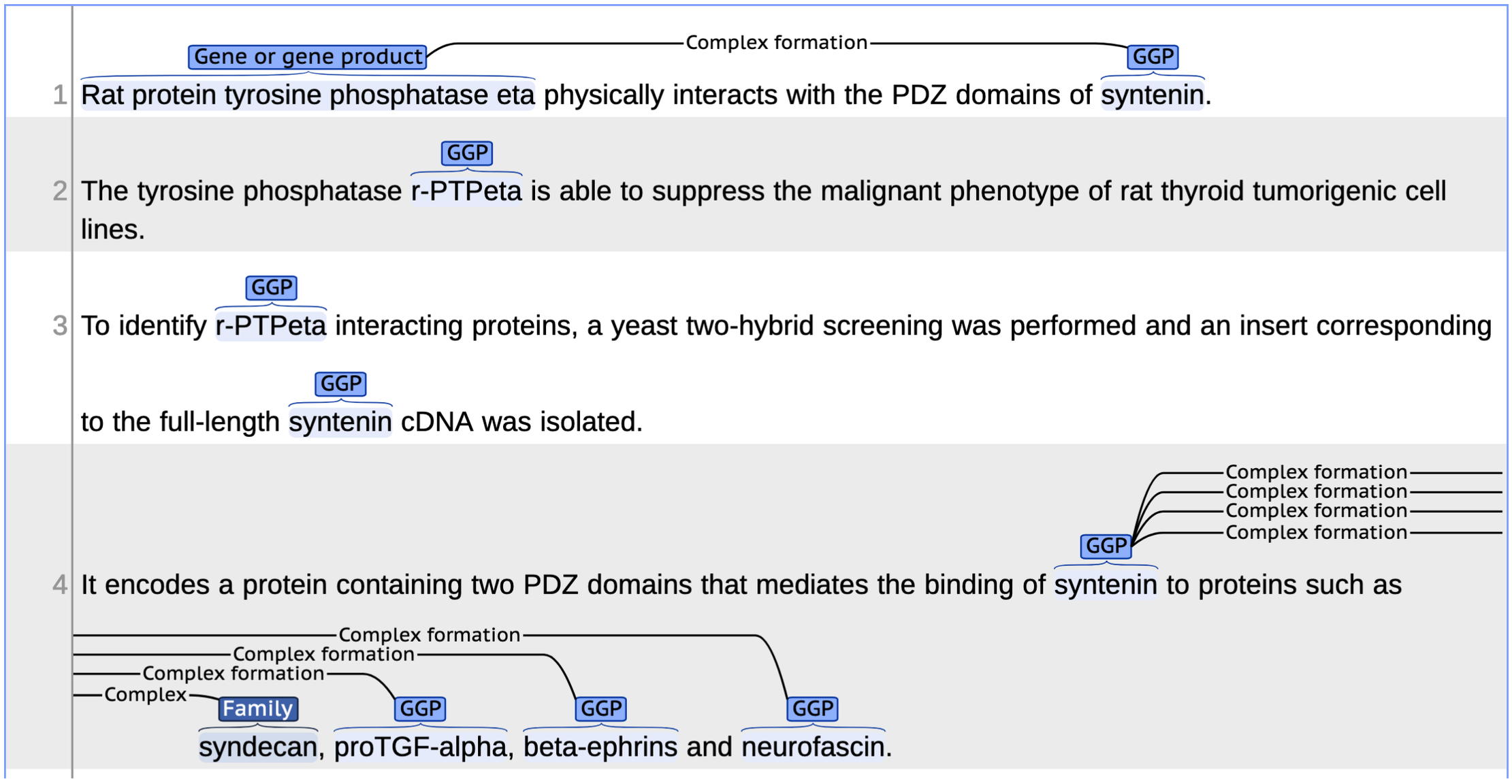
Taxonomy:77044

Scientific name:Rosellinia necatrix

rank:species

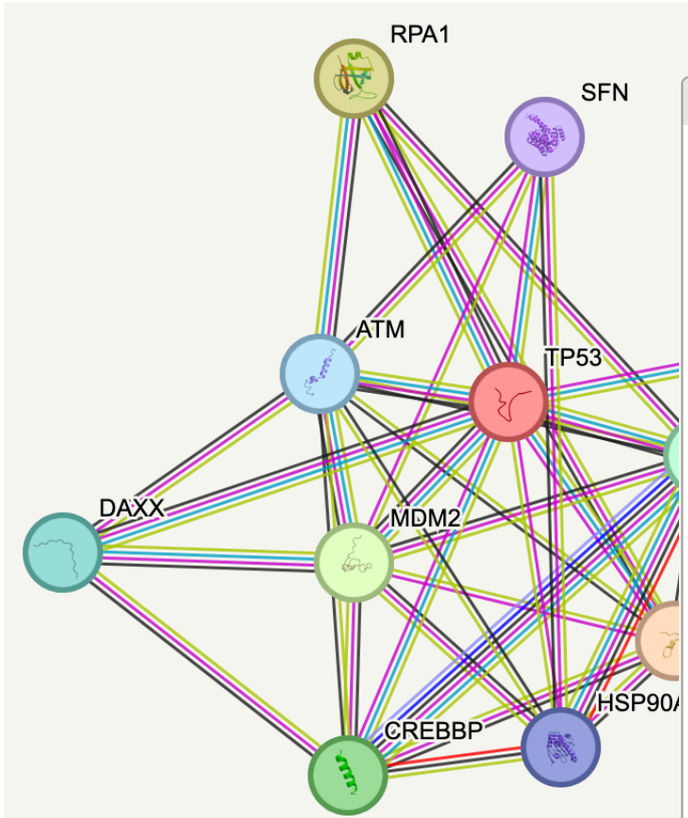
Relation extraction

Example: `Complex formation` relation extraction



Analysis

Example: STRING DB interaction network for human p53



Interaction

TP53 *[ENSP00000269305]*

Cellular tumor antigen p53; Acts as a tumor suppressor in many tumor types; induces growth arrest or apoptosis depending on the physiological circumstances and cell type. Involved in cell cycle regulation as a trans-activator that acts to negatively regulate cell division by controlling a set of genes required for this process. One of the activated genes is an inhibitor of cyclin-dependent kinases. Apoptosis induction seems to be mediated either by stimulation of BAX and FAS antigen expression, or by repression of Bcl-2 expression. Its pro-apoptotic activity is activated via its intera [...]

↔

MDM2 *[ENSP00000258149]*

E3 ubiquitin-protein ligase Mdm2; E3 ubiquitin-protein ligase that mediates ubiquitination of p53/TP53, leading to its degradation by the proteasome. Inhibits p53/TP53- and p73/TP73-mediated cell cycle arrest and apoptosis by binding its transcriptional activation domain. Also acts as a ubiquitin ligase E3 toward itself and ARRB1. Permits the nuclear export of p53/TP53. Promotes proteasome-dependent ubiquitin-independent degradation of retinoblastoma RB1 protein. Inhibits DAXX-mediated apoptosis by inducing its ubiquitination and degradation. Component of the TRIM28/KAP1-MDM2-p53/TP53 [...]

Evidence suggesting a functional link:

Neighborhood in the Genome:	none / insignificant.	
Gene Fusions:	none / insignificant	
Cooccurence Across Genomes:	none / insignificant	
Co-Expression:	yes (score 0.049). In addition, putative homologs are coexpressed in other organisms (score 0.109).	show
Experimental/Biochemical Data:	yes (score 0.999). In addition, putative homologs were found interacting in other organisms (score 0.185).	show
Association in Curated Databases:	yes (score 0.900).	show
Co-Mentioned in Pubmed Abstracts:	yes (score 0.999). In addition, putative homologs are mentioned together in other organisms (score 0.100).	show
Combined Score:	0.999	

Text vs. structured data

Input: "Tyrosine phosphatase eta physically interacts with the PDZ domains of syntenin."

Output (names + normalization):

ID	Type	Start	End	Text	DBRef
T0	Protein	0	24	Tyrosine phosphatase eta	Gene:24326
T1	Protein	70	78	Syntenin	Gene:83841

Output (relations):

ID	Type	Arg1	Arg2
T0	Complex formation	T0	T1

Approach

Many information extraction tasks can be straightforwardly cast as supervised classification, e.g.:

- **Document selection** → text (word sequence) classification
- **Named entity recognition** → token (word) classification
- **Relation extraction** → text classification

We'll briefly look at these three next

(Normalization can be cast e.g. as ranking pairs of text, one representing mention in context and the other a candidate entity definition)

Document selection

Document selection is a simple text classification task

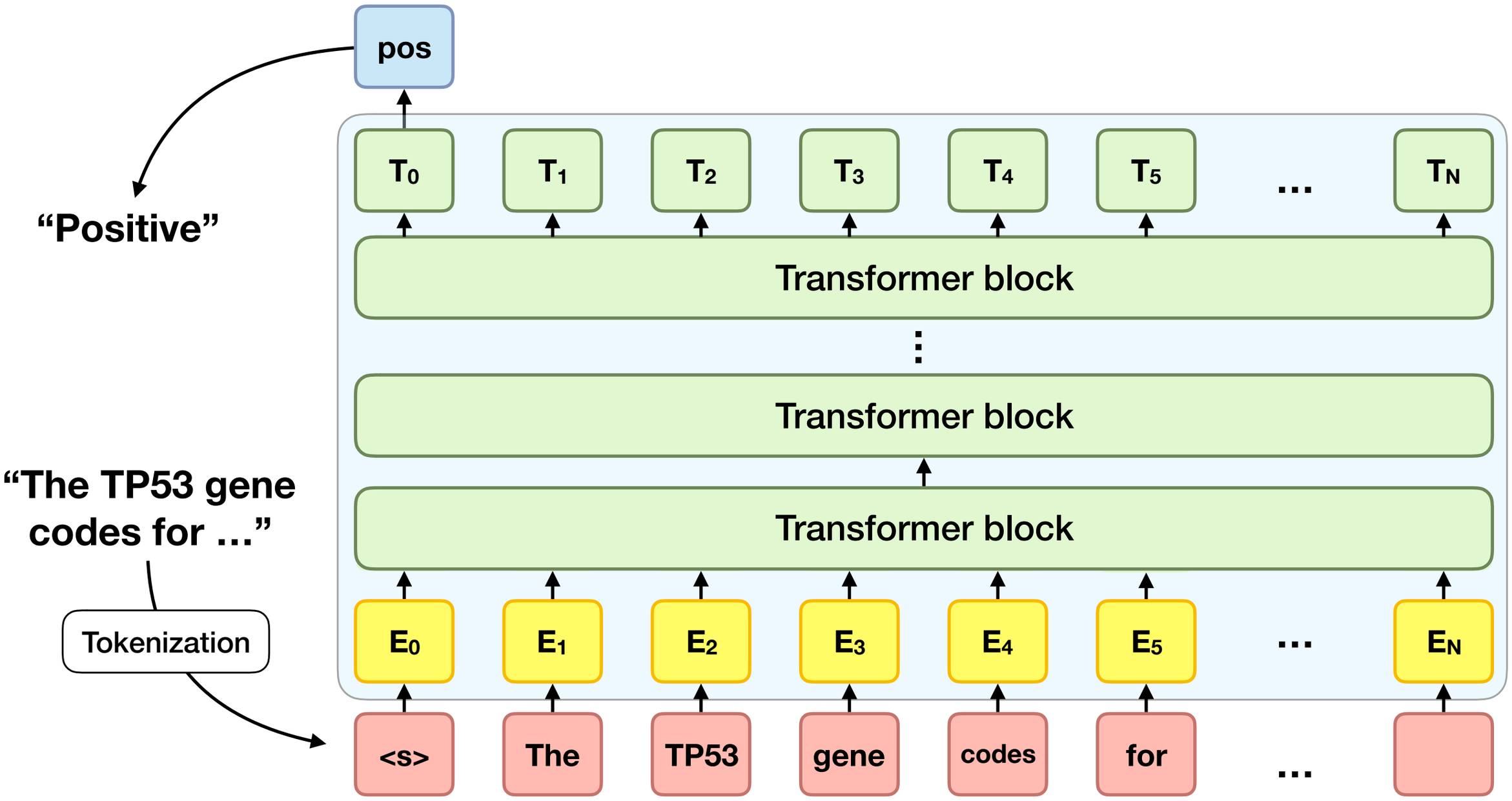
Input: text, represented as sequence of words

Output: label (or set of labels) from predefined categories (e.g. `Positive`/`Negative`, `Relevant`/`Irrelevant`)

Approach: attach output layer e.g. to start-of-text token or a pooling layer (e.g. max-pool)

(multiclass vs. multilabel classification is mostly a matter of choosing appropriate activation and loss functions)

Text classification



Named entity recognition

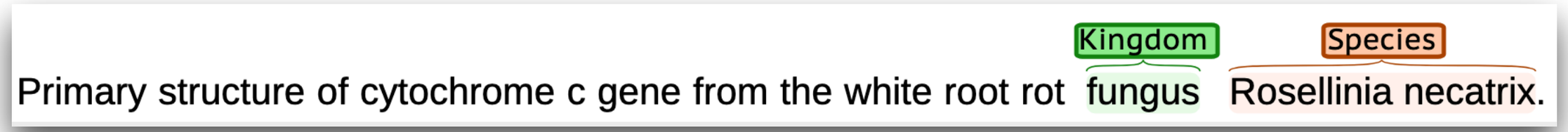
The recognition of (non-overlapping) mentions of names of entities of interest can be cast as token classification:

Input: text, represented as sequence of words

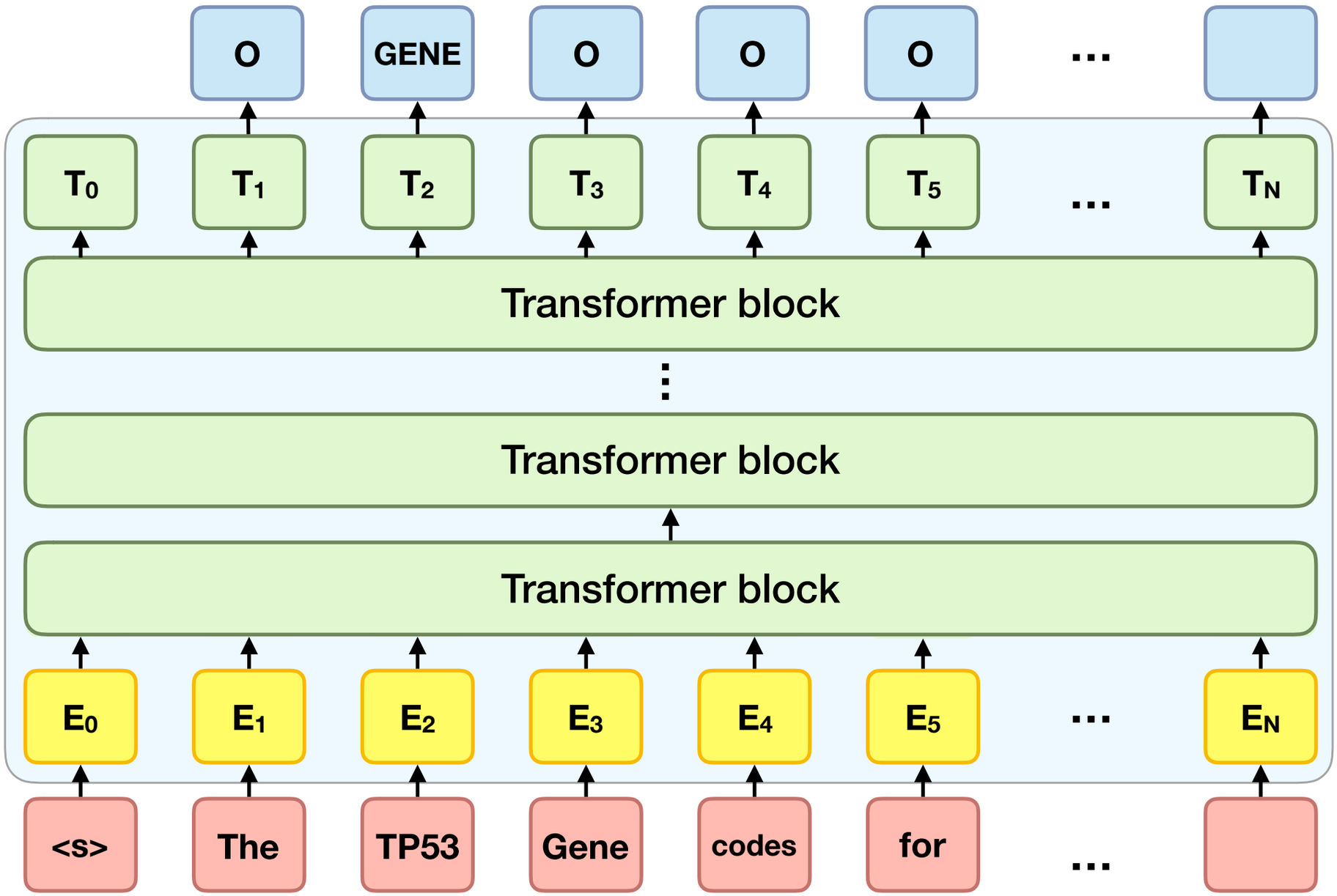
Output: sequence of labels (one per word) from predefined categories (e.g. `Gene`, `Disease`, `Species`, etc.)

(glossing over boundary issues and IOB encoding)

Approach: attach output layer to each token, fine-tune with gold standard data



Token (word) classification



Relation extraction

Identification and classification of all (typically binary) relations of interest between mentions of entities of interest (from NER)

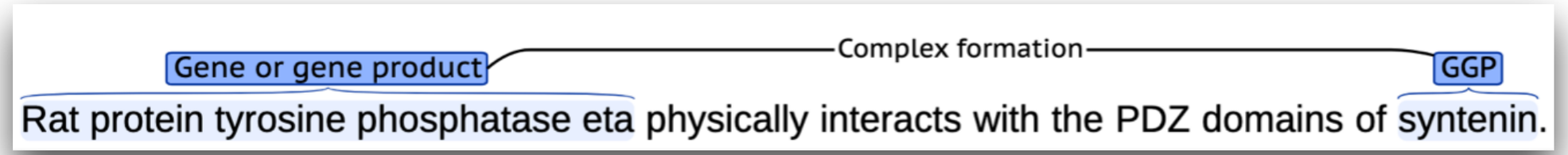
In a pipeline, for example:

Input: text (sequence of words) and entity mentions (types and offsets)

Output: triples of (start-entity, end-entity, relation-type)

Approach: consider all pairs of entity mentions, mark each one in turn in text, cast as text classification with relation type as label

e.g. "<E1>tyrosine phosphatase eta</E1> physically interacts with the PDZ domains of <E2>syntenin</E2>." → `Complex formation`

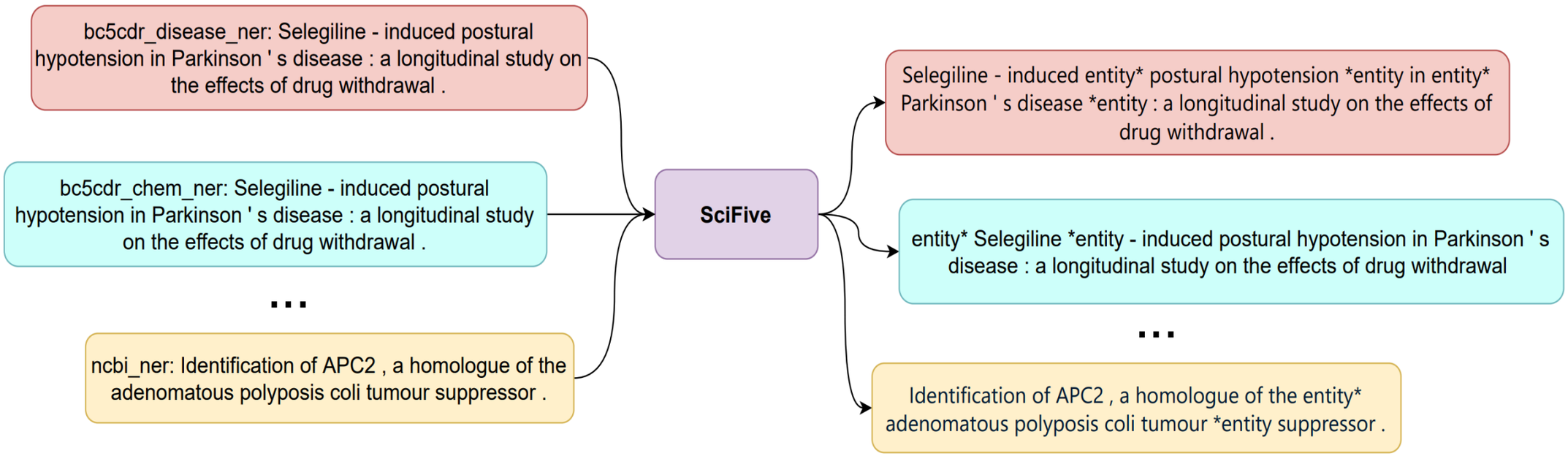


End-to-end approach

Instead of breaking down the mapping from input text to structured representation into subtasks, IE could be performed using and **end-to-end** approach using e.g. a text-to-text model:

Input: text (e.g. PubMed abstract)

Output: structured representation



Phan et al. (2021) "SciFive: a text-to-text transformer model for biomedical literature"

Just ask ChatGPT?

Large causal language models are capable of *attempting* effectively any text mining task out of the box

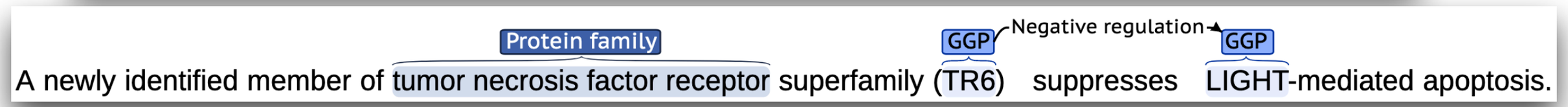
→ Simply fomulate an appropriate prompt

Issues: lack of fine-grained **control**, **costs** of running massive models at very large scale

List the protein names appearing in the following text:

A newly identified member of tumor necrosis factor receptor superfamily (TR6) suppresses LIGHT-mediated apoptosis.

1. Tumor necrosis factor receptor superfamily (TR6)
2. LIGHT



In practice

Practical exercises

Transformers have exellent support for all major deep learning libraries

We'll here focus on **PyTorch**, the library of choise for most work in the area

We'll be using the **Transformers** library and **Hugging Face Hub** resources

Code will be run on **Colab**

https://tinyurl.com/danish-summer

A few words on resources first

Hugging Face 🤗

In recent years, Hugging Face (https://huggingface.co/) has emerged as a leading platform for Transformer models and tools

The Transformers Python library has extensive support for loading, training, and using transformer-based models

The Hugging Face Hub is a large repository including

- Models: over 250,000 models including both major pre-trained models as well as fine-tuned models for a broad range of tasks
- Datasets: over 50,000 datasets representing various tasks in NLP and other areas (e.g. sound and image processing)

Hugging Face 🤗

Models (https://huggingface.co/models)

Models 279,783 | Filter by name | new Full-text search | ↑↓ Sort: Trending

- stabilityai/stable-diffusion-xl-base-1.0
  Text-to-Image • Updated about 23 hours ago • 334k • ♡ 1.4k
- meta-llama/Llama-2-7b
  Text Generation • Updated 17 days ago • ♡ 1.51k
- Qwen/Qwen-7B-Chat
  Text Generation • Updated 36 minutes ago • 7.82k • ♡ 210
- togethercomputer/LLaMA-2-7B-32K
  Text Generation • Updated about 5 hours ago • 5.46k • ♡ 259
- stabilityai/stable-diffusion-xl-refiner-1.0
  Text-to-Image • Updated about 23 hours ago • 115k • ♡ 542
- stabilityai/StableBeluga2
  Text Generation • Updated 5 days ago • 166k • ♡ 717
- THUDM/chatglm2-6b-32k
  Updated 2 days ago • 25.4k • ♡ 127
- amazon/FalconLite
  Text Generation • Updated 2 days ago • 26.6k • ♡ 118
- meta-llama/Llama-2-70b-chat-hf
  Text Generation • Updated 4 days ago • 124k • ♡ 875
- Qwen/Qwen-7B
  Text Generation • Updated 37 minutes ago • 2.33k • ♡ 113
- meta-llama/Llama-2-7b-chat-hf
  Text Generation • Updated 4 days ago • 303k • ♡ 610
- TheBloke/Llama-2-7B-Chat-GGML
  Text Generation • Updated 11 days ago • 5.23k • ♡ 248

Selected models

Hundreds of pre-trained "foundation" models for various languages and domains and hundreds of thousands of fine-tuned models are readily available

Selecting one that fits your needs can be challenging

We've already discussed **GPT**, **BERT**, and **T5**

We'll next look at a few with particular relevance to **biomedical text mining**

BioBERT

Encoder-only models trained by Lee et al. (2020) on **PubMed abstracts** and **PMC open access full text** publications

BERT base (**110M parameters**) and large (**340M parameters**) size models

Highly competitive but far from the only choice of a biomedical domain encoder-only model, also e.g. SciBERT, BlueBERT, PubMedBERT, BioMegatron, …

https://huggingface.co/dmis-lab/biobert-base-cased-v1.2

https://huggingface.co/dmis-lab/biobert-large-cased-v1.1

Lee et al. (2020) “BioBERT: a pre-trained biomedical language representation model for biomedical text mining”

GALACTICA

Decoder-only models pre-trained on **106 billion words** from scientific sources by Meta

Model sizes range from **125M** (“mini”) to **120B** (“huge”) parameters

When published, the models established a new state-of-the-art in various tasks, including medical question answering

https://huggingface.co/facebook/galactica-1.3b

https://galactica.org/

Taylor et al. (2022) "Galactica: A Large Language Model for Science"

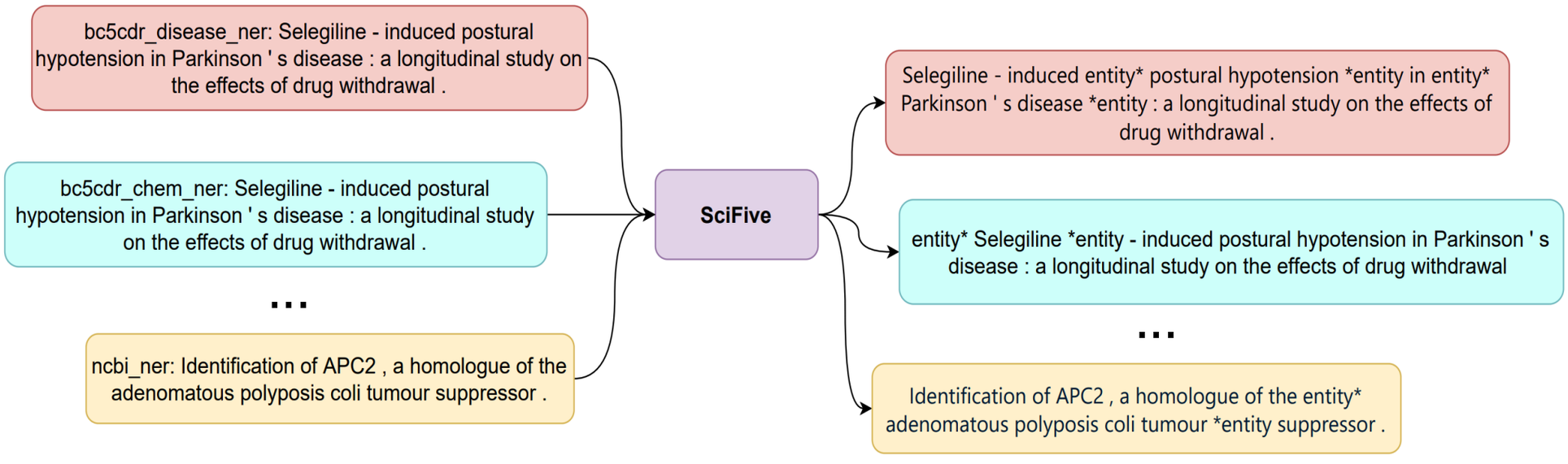
SciFive

Encoder-decoder models trained by Phan et al. (2021)

Model sizes T5-base (**220M parameters**) and T5-large (**770M parameters**)

https://huggingface.co/razent/SciFive-base-Pubmed_PMC

https://huggingface.co/razent/SciFive-large-Pubmed_PMC



Phan et al. (2021) "SciFive: a text-to-text transformer model for biomedical literature"

Biomedical datasets

There is **no central repository** of datasets for biomedical text mining

The **BigScience Biomedical Datasets** collaborative effort has compiled a set of over 100 biomedical domain datasets for e.g.

- Gene/protein, chemical, disease and organism name NER
- Normalization for various biomedical entities
- Protein-protein binding and chemical-protein relations
- Event extraction for various biomedical processes

https://huggingface.co/bigbio

Text generation

GALACTICA text generation notebook

Text generation using a causal LM and LMs as knowledge base / article writing support

Try: prompt the model to generate some claim(s) relevant to your research. Are these correct?

Text generation with GALACTICA

This notebook demonstrates text generation with a small GALACTICA model (https://galactica.org/) on Colab.

First, we'll install the required Python packages. The transformers package is used to load the model and run generation, and the accelerate package supports running large models efficiently on multiple devices.

```
!pip install --quiet transformers accelerate
```

Next, we'll import the `AutoTokenizer` and `AutoModelForCausalLM` classes. These support loading tokenizers and models from the Hugging Face Hub.

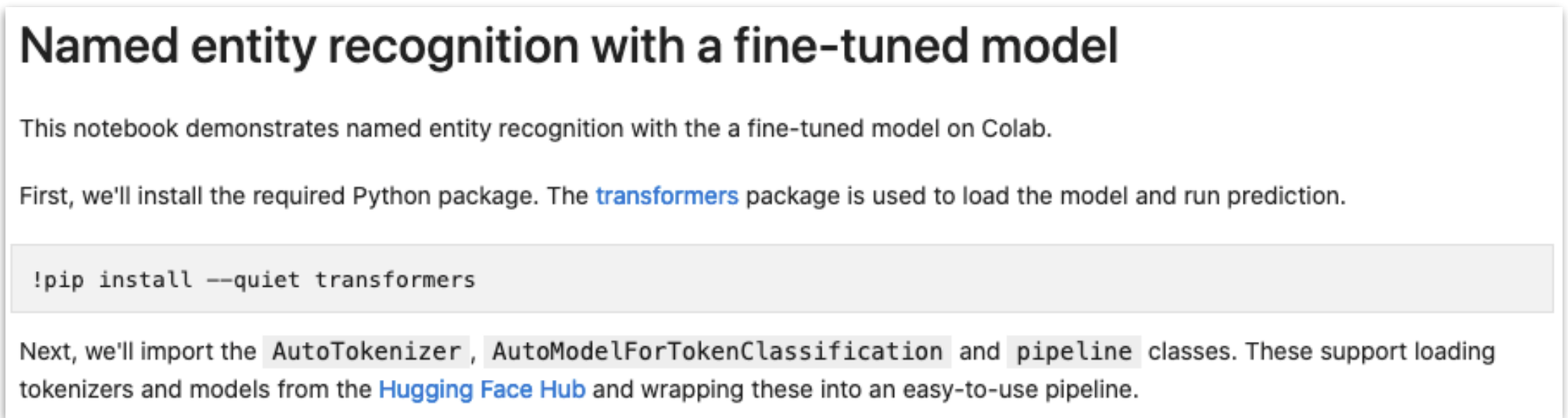
Taylor et al. (2022) "Galactica: A Large Language Model for Science"

Named entity recognition

BioBERT chemical NER notebook

NER using a encoder-only model fine-tuned on a chemical NER dataset

Try: provide the model with sentences that include mentions of chemicals and other entities (e.g. proteins). Does the model tag the correct mentions?



Named entity recognition with a fine-tuned model

This notebook demonstrates named entity recognition with the a fine-tuned model on Colab.

First, we'll install the required Python package. The transformers package is used to load the model and run prediction.

```
!pip install --quiet transformers
```

Next, we'll import the `AutoTokenizer`, `AutoModelForTokenClassification` and `pipeline` classes. These support loading tokenizers and models from the Hugging Face Hub and wrapping these into an easy-to-use pipeline.

Fine-tuning for NER

NER fine-tuning notebook

NER using the original BERT model and a "general domain" dataset

Try: replace the model and dataset with biomedical domain ones, (e.g. BioBERT and `bc2gm_corpus`) to fine-tune a model for biomedical domain NER

Fine-tuning a model for NER

Let's train a transformer model on a Named Entity Recognition (NER) dataset.

Setup

Install the required Python packages:

```
!pip install --quiet transformers datasets evaluate seqeval accelerate
```