AI INSTITUTE #AIISC
UNIVERSITY OF SOUTH CAROLINA

UNIVERSITY OF SOUTH CAROLINA

CSCE 580: Introduction to AI

Week 5 - Lectures 9 and 10: Large Language Models

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

16TH AND 18TH SEP 2025

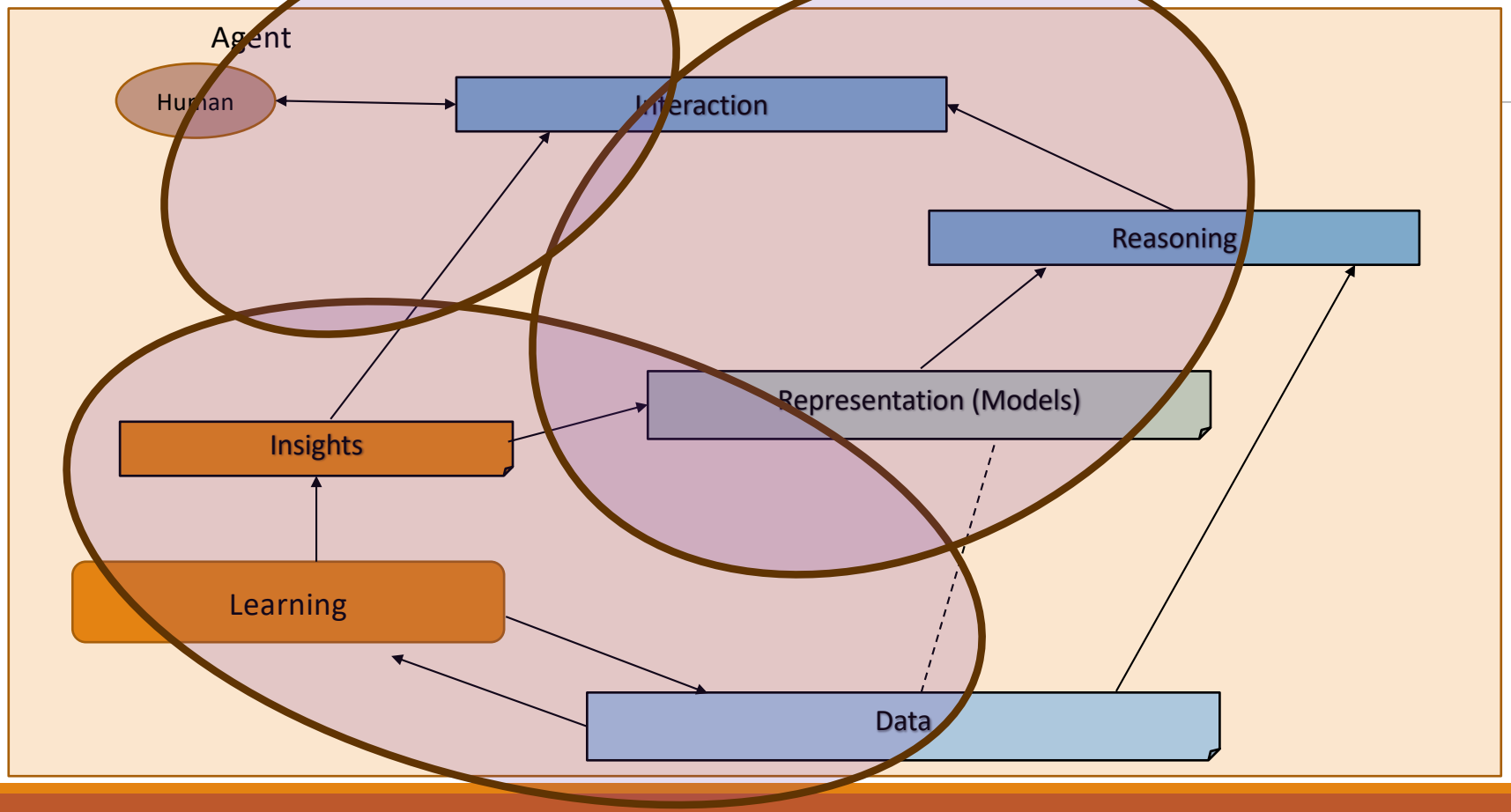
Carolinian Creed: "I will practice personal and academic integrity."

Credits: **Copyrights of all material reused acknowledged**

Organization of Week 5 - Lectures 9, 10

- Introduction Section
  - Recap from Week 4 (Lectures 7 and 8)
  - Due: Quiz 1, Project 1
  - AI news
- Main Section
  - L9: NN Wrapup; Large Language Models (LLMs)
  - L10: Using LLMs, Trust
- Concluding Section
  - About next week – W6: Lectures 11, 12
  - Ask me anything

Relationship Between Main AI Topics (Covered in Course)



Recap of Week 4

- We talked about
  - Unsupervised ML
  - NN and DL
  - Quiz 1

- Week 1: Introduction, Aim: Chatbot / Intelligence Agent
- Weeks 2-3: Data: Formats, Representation and the Trust Problem
- Week 3: Machine Learning – Supervised (Classification)
- Week 4: Machine Learning - Unsupervised (Clustering) –
- Topic 5: Learning neural network, deep learning, Adversarial attacks
- Week 6: Large Language Models – Representation and Usage issues
- Weeks 7-8: Search, Heuristics - Decision Making
- Week 9: Constraints, Optimization – Decision Making
- Topic 10: Markov Decision Processes, Hidden Markov models - Decision making
- Topic 11-12: Planning, Reinforcement Learning – Sequential decision making
- Week 13: Trustworthy Decision Making: Explanation, AI testing
- Week 14: AI for Real World: Tools, Emerging Standards and Laws; Safe AI/ Chatbots

Quiz 1: Take Home

- Due date: **Tuesday, Sep 16**

- Remember to update spreadsheet on data/ time when finished **(Column G)**

Projects A: Start (4 weeks; 200 points)

- End date: **Thursday, Sep 18**
  - Remember to update spreadsheet on data/ time when finished **(Column I)**

- See model AI Assignments: http://modelai.gettysburg.edu/
  - Choose a project, preferably within last 5 years (i.e., after 2020).
    - Enter its name in **"Student-InfoShared .." sheet, column G**
  - Follow instructions and do it alone
  - Submit project outcome
    - Create a folder in your Github called **ProjectA**.
    - Create a file called "ProjectInfo.md" with your name, project chosen and URL/ other details.
    - Put deliverables, as per project description, inside the folder, and commit.
    - Timestamp will be used to confirm that Project-A is done on time

AI News

#1 NEWS – AIx for Seniors Event

Day long event:
Friday, Nov 14

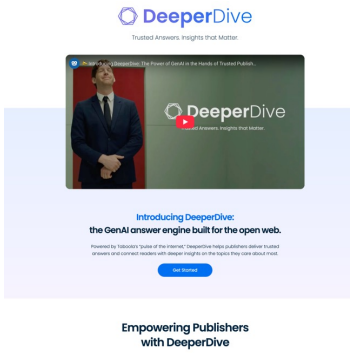
Tentative Schedule

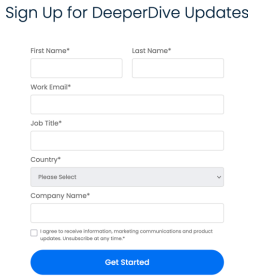
8:30am-9:00 am:	Breakfast
9:00am-9:15am:	Opening comments
9:15am-10:00am:	Talk 1: AI
10:00am-10:45am:	Talk 2: Cybersecurity & AI
10:45am-11:45am:	Open Discussion - How AI can help me?
11:45am-01:00:pm	Demo session
1:00pm-02:00:pm	Lunch
2:00pm-2:45pm:	Talk 3: Law and AI
2:45pm-3:30pm:	Talk 4: Seniors Health & AI
3:30pm-4:30pm:	Open Discussion: AI ready?
4:30pm-5:30pm:	AI Bingo

#2 NEWS – Chatbot by News Organizations

Key points

- USA Today has a chatbot, called DeeperDive, fed with its media content across 100+ papers
- Developed for them my advertising company, Taboola





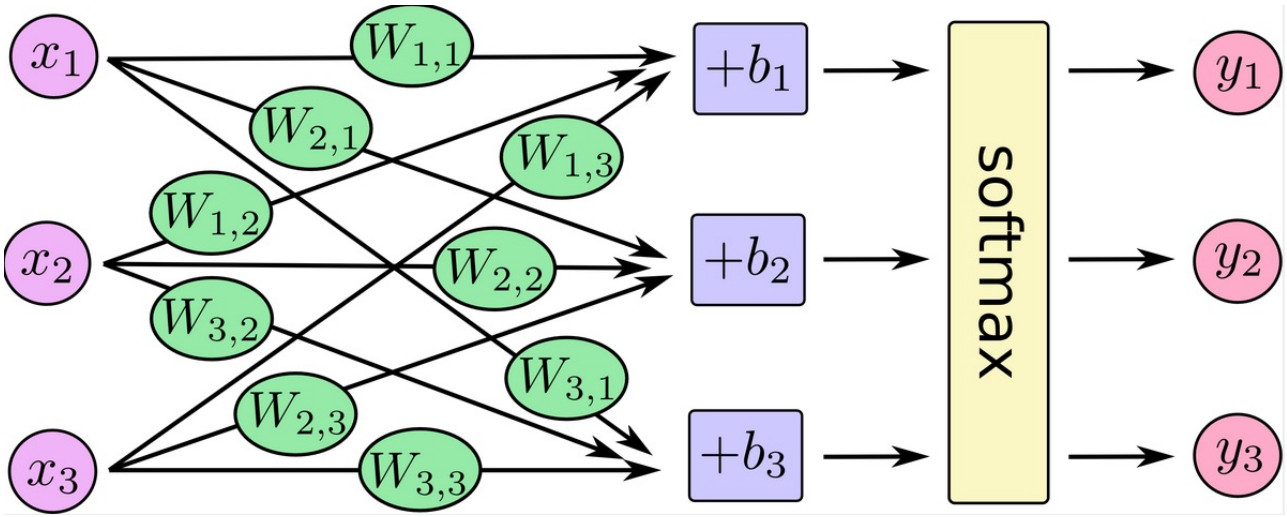
1. Can converse with readers, summarize insights from its journalism, and suggest new content from across its sites
2. Replaces a conventional search box and automatically suggests questions that readers might want to ask
3. **Observation:** More about advertising than better content

Press: https://www.wired.com/story/usa-today-enters-its-gen-ai-era-with-a-chatbot/

Introduction Section

Lecture 9: Wrapping up NN

Using (Feed forward) NN



Softmax

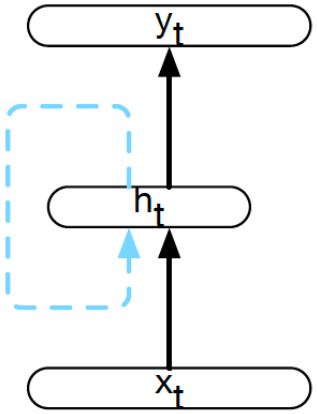
$$f(x)= \frac{1}{1+e^{-x}}$$

Source; see also : https://jalammar.github.io/visual-interactive-guide-basics-neural-networks/, https://jalammar.github.io/feedforward-neural-networks-visual-interactive

RNN - Recurrent Neural Networks

- **Recurrence**: A *recurrence* relation is an equation that defines a sequence based on a rule that gives the next term as a *function* of the previous term(s).
[https://mathinsight.org/definition/recurrence_relation]

- Simple Recurrent NN or Elman Network

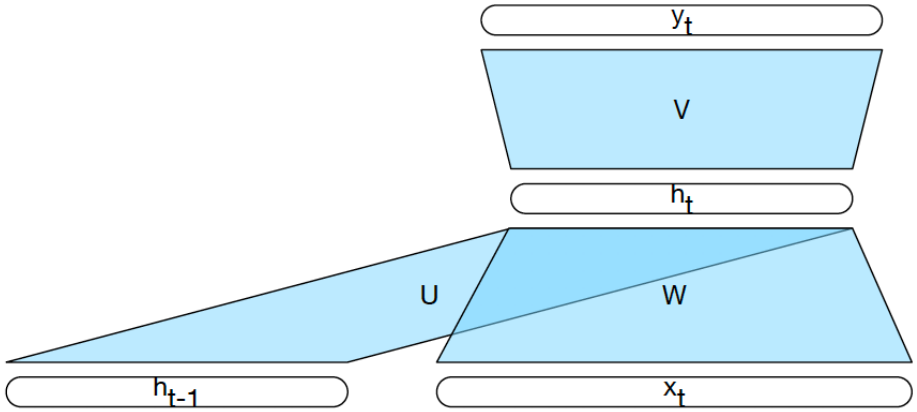


Source: Jurafsky and Martin

RNN

Recurrence unrolled

U, W, V are Weights to be learned



$$h_t = g(Uh_{t-1} + Wx_t)$$
$$y_t = f(Vh_t)$$

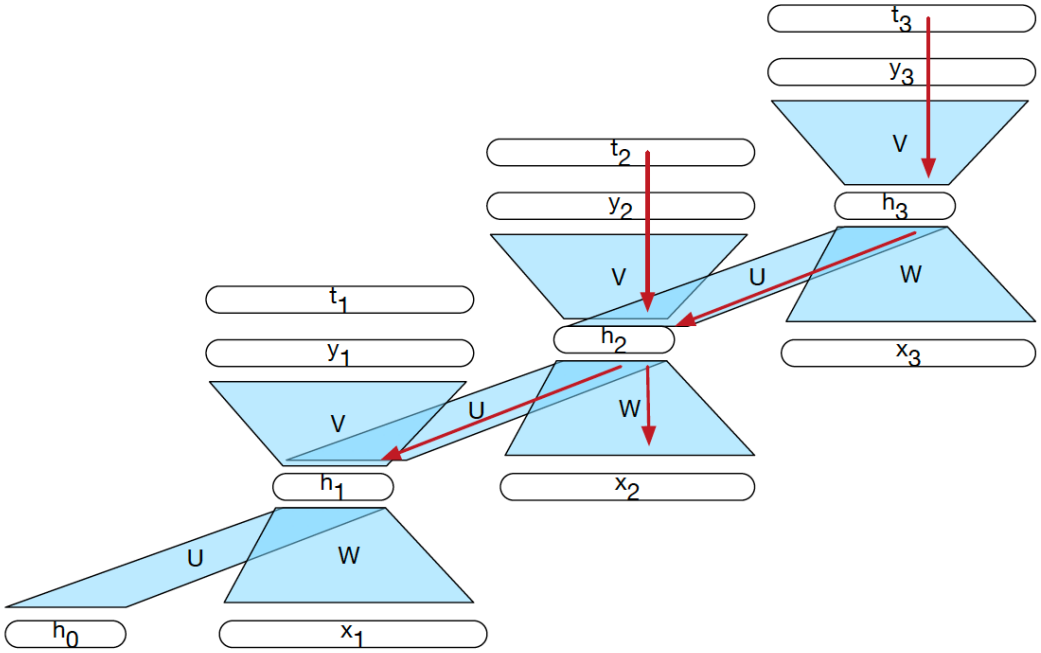
$$y_t = softmax(Vh_t)$$

Source: Jurafsky and Martin

RNN Backpropagation of Errors

Recurrence unrolled

U, W, V are Weights to be learned



Source: Jurafsky and Martin

RNN-based Language Model

- Based on characters or words
- At each step (i.e., character or word)
  - the network retrieves a word embedding for the current word as input
  - combines it with the hidden layer from the previous step to
    - compute a new hidden layer
    - generate an output layer which is passed through a softmax layer to generate a probability distribution over the entire vocabulary.

$$P(w_n|w_1^{n-1}) = y_n$$
$$= softmax(Vh_n)$$

Prob. of a word

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k|w_1^{k-1})$$
$$= \prod_{k=1}^{n} y_k$$

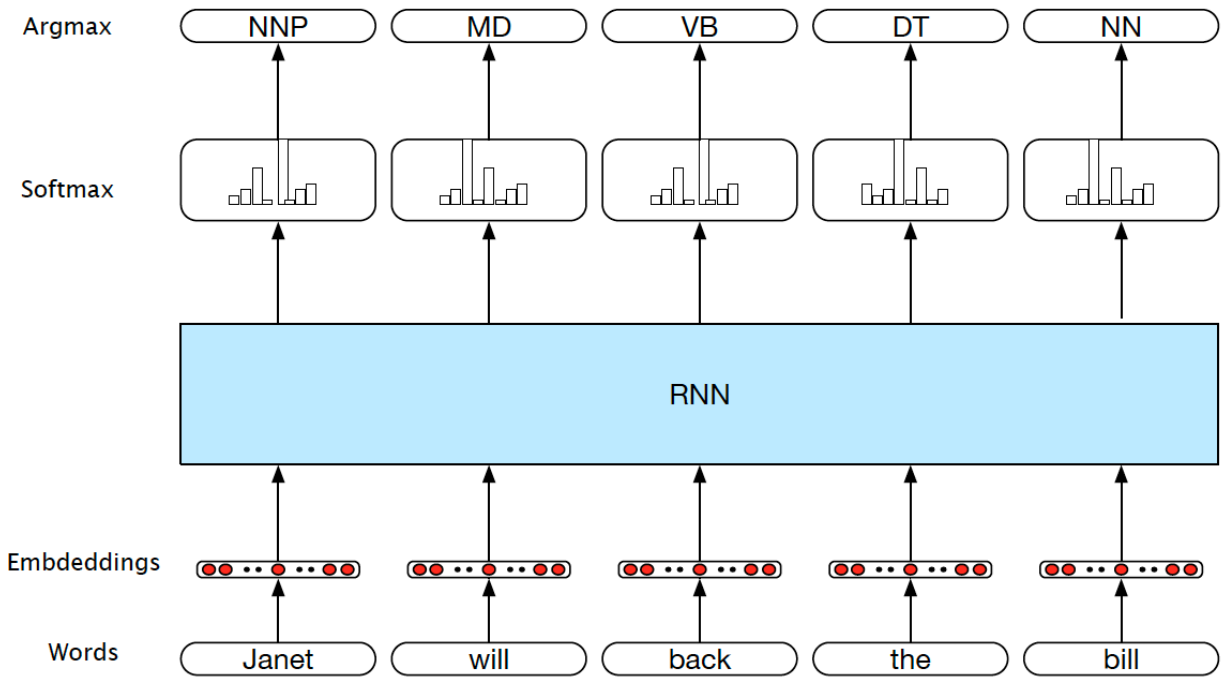
Prob. of a sequence

Source: Jurafsky and Martin

RNN Discussion

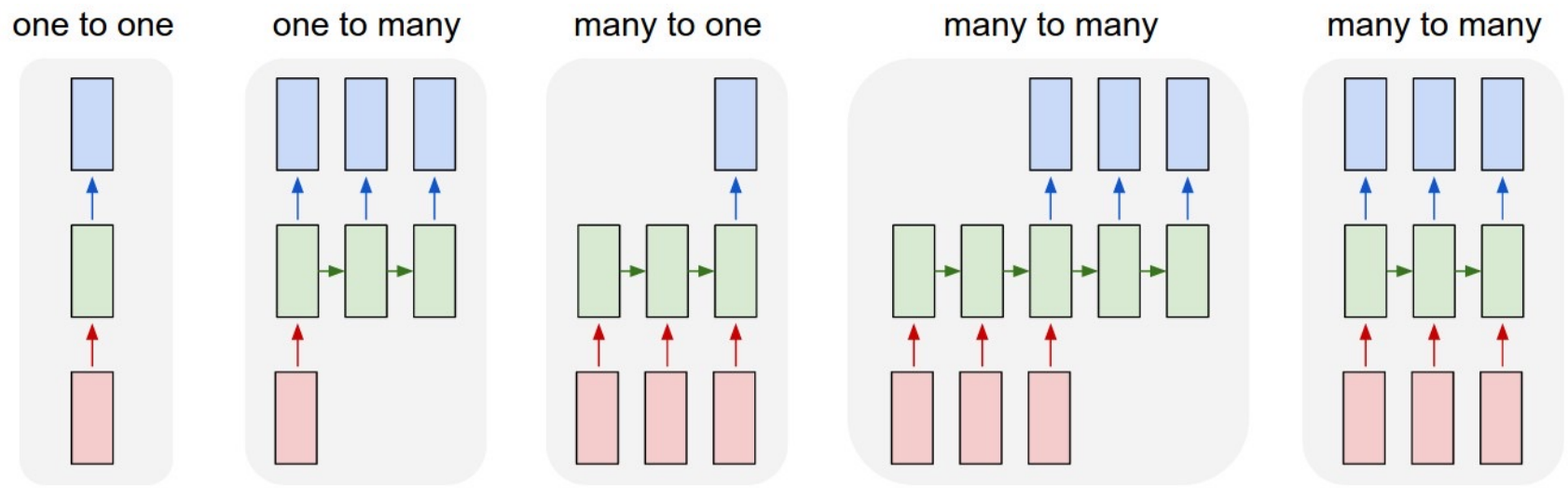
- Language model
  - Not dependent on N-gram boundaries
  - Whole sequence is the context
- Program generation
  - Complexity is Turing-complete
  - In practical terms: On the Practical Computational Power of Finite Precision RNNs for Language Recognition, Gail Weiss, Yoav Goldberg, Eran Yahav, ACL 2018, https://www.aclweb.org/anthology/P18-2117/

RNN Usage Example: Sentence Labeling



Source: Jurafsky and Martin

RNN - Many Applications



Language model

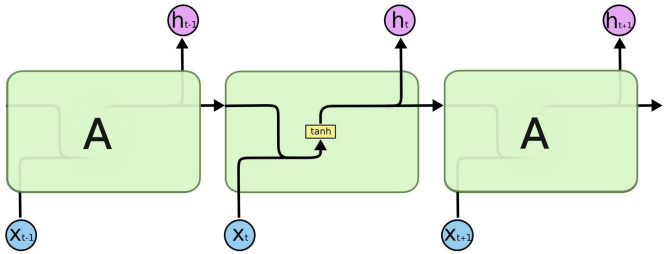
Caption generation

Sentiment detection

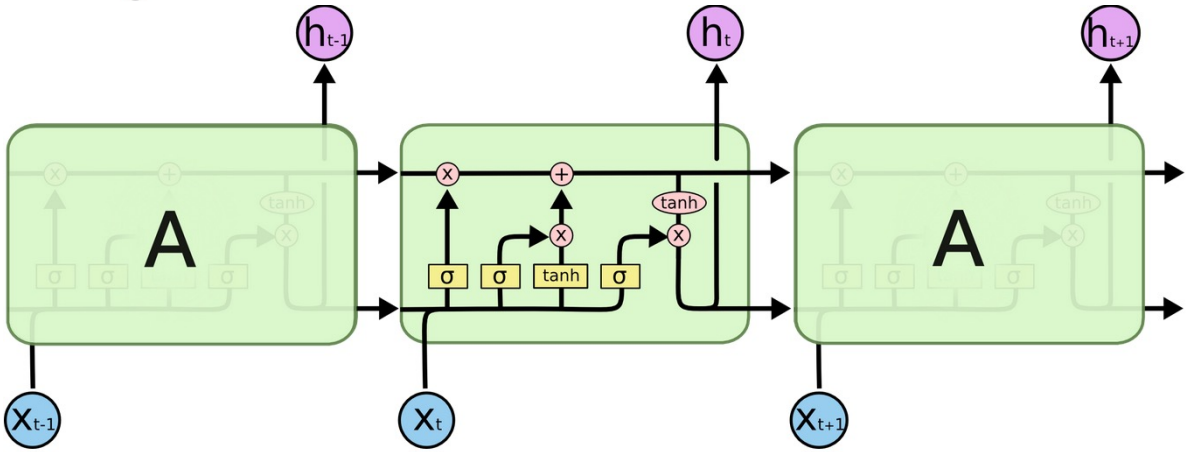
Machine translations

Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNN and LSTM - Long Short Term Memory



RNN



LSTM

To learn long term relationships; has 4 NNs

Source and details: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Trust: Adversarial Attacks

- Cat and mouse on attacks and defenses
  - Example code: https://github.com/Trusted-AI/adversarial-robustness-toolbox/blob/main/notebooks/adversarial_training_mnist.ipynb
- Tools
  - Adversarial Robustness Toolbox (ART) - Python library for Machine Learning Security, https://github.com/Trusted-AI/adversarial-robustness-toolbox

Trust Issues with NN

- Robustness: can the model give the results in the presence of (input) perturbation? Noise?
- Computation/ footprint: why does the learning take so much compute resources?
- Data: is the data representative? How was the data obtained?
- Explainability: why does the model work?
- Fairness: Is the output fair to user groups?

Resources and Books

- Understanding Deep Learning, https://udlbook.github.io/udlbook/
- Deep Learning, Ian Goodfellow, Yoshua Bengio and Aaron Courville, https://www.deeplearningbook.org/
- AI – A Modern Approach, Russell & Norvig, https://aima.cs.berkeley.edu/
- Websites of libraries – Keras.

Lecture 9: LLMs

Language Model

Problem:
Given a sentence fragment, predict what word(s) come next

Language Model:
estimate probability of substrings of a sentence

$$P(w_i|w_1, w_2, \ldots, w_{i-1}) = \frac{P(w_1, w_2, \ldots, w_{i-1}, w_i)}{P(w_1, w_2, \ldots, w_{i-1})}$$

Applications:
- Spelling correction
- speech recognition
- machine translation,
- …

Bigram approximation

$$P(w_i|w_1, w_2, \ldots, w_{i-1}) \approx \frac{P(w_{i-1}, w_i)}{P(w_{i-1})}$$

From Jurafsky & Martin

Language Model

Markovify library
https://github.com/jsvine/markovify

Language Model:
estimate probability of substrings of a sentence

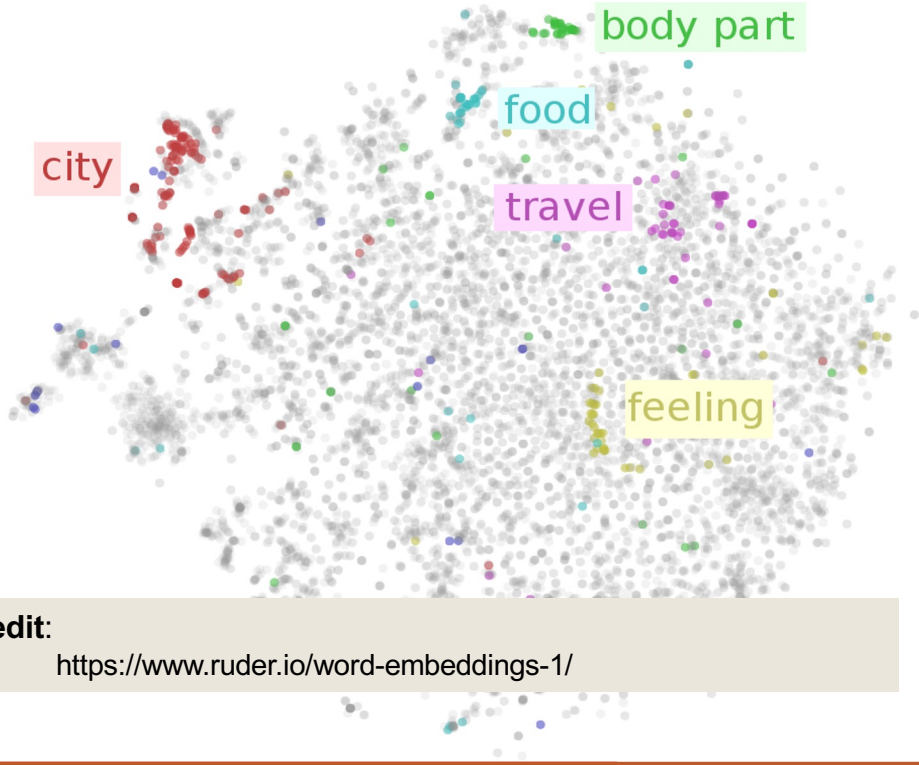
$$P(w_i|w_1, w_2, \ldots, w_{i-1}) = \frac{P(w_1, w_2, \ldots, w_{i-1}, w_i)}{P(w_1, w_2, \ldots, w_{i-1})}$$

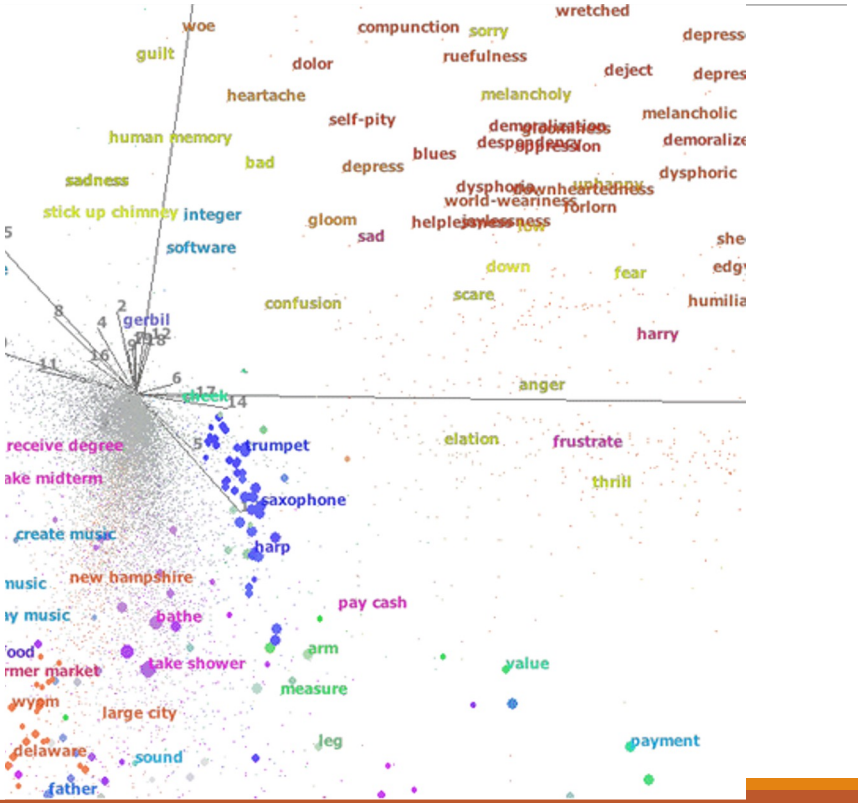
See code samples with Markovify library on Github

- *Prepare data – two datasets shown*
- *Try generator:*
  - *https://github.com/biplav-s/course-nl/blob/master/l7-language/code/TryMarkovifyLangModel.ipynb*

Evaluating LMs

Evaluating Language Models





Credit:

- https://www.ruder.io/word-embeddings-1/

Evaluation – Language Model

- **Intrinsic evaluation**: measure the quality of a model independent of any application
- **Extrinsic evaluation**: situate model in an application and evaluate the whole application for improvement. Also called in-vivo evaluation

Perplexity

$$\text{Average NLL} = -\frac{1}{N}\sum_{i=1}^{N}\log P(w_i|w_1, w_2, ..., w_{i-1})$$

where N is the total number of words in the test dataset, and $P(w_i|w_1, w_2, ..., w_{i-1})$ is the probability of word w_i given the previous words $w_1, w_2, ..., w_{i-1}$.

$$\text{Perplexity} = e^{\text{Average NLL}}$$

Value range: best: 1, **worst**: positive infinite;
practical upper bound: number of words in vocabulary

1. P(John) = 0.1
2. P(bought | John) = 0.4
3. P(apples | bought) = 0.3
4. P(from | apples) = 0.5
5. P(the | from) = 0.6
6. P(market | the) = 0.7

Now, let's compute the probability of the generated sequence:

$P(\text{"John bought apples from the market"}) = P(John) \times P(bought|John) \times P(apples|bought) \times P(from|apples) \times P(the|from) \times P(market|the)$

$P(\text{"John bought apples from the market"}) = 0.1 \times 0.4 \times 0.3 \times 0.5 \times 0.6 \times 0.7$

Hence, P("John bought apples from the market") = 0.00252

Average NLL = – log(0.00252) / 6. [N = 6 as the model generated six words]

Hence, Average NLL = 0.99725

Perplexity = Exp(Average NLL) = 2.71

Credit:

- https://blog.uptrain.ai/decoding-perplexity-and-its-significance-in-llms/
- https://medium.com/@priyankads/perplexity-of-language-models-41160427ed72

Perplexity Comments

1. A model with a vocabulary of 10,000 words and a perplexity of 2.71 is much better than a model with a vocabulary of 100 words and the same perplexity score of 2.71.
2. Lower perplexity results in higher consistency. As we know, LLMs are non-deterministic, i.e., the same inputs can result in two different outputs; a lower perplexity means that the model is more likely to produce the same output over multiple runs.
3. Perplexity is the inverse of the geometric mean of the probability of each word. Hence, the inverse of average probability (2.3077 in the previous case) can be considered a good proxy for quick calculations.
4. This calculation happens in the tokens space (compared to the words space), but the core principle remains the same.

Credit:

- https://blog.uptrain.ai/decoding-perplexity-and-its-significance-in-llms/

Perplexity

- Suppose
  - P('X') = 0.25
  - P('Y') = 0.5
  - P('Z') = 0.25
- Perplexity
  - ('XXX') = - exp (log (0.25 x 0.25 x 0.25) * (1/3)) = 3.94
  - Perplexity ('XYX') = - exp(log(0.25 x 0.5 x 0.25) * (1/3)) =
- Lower the number, the better is the model

Evaluation – Extrinsic

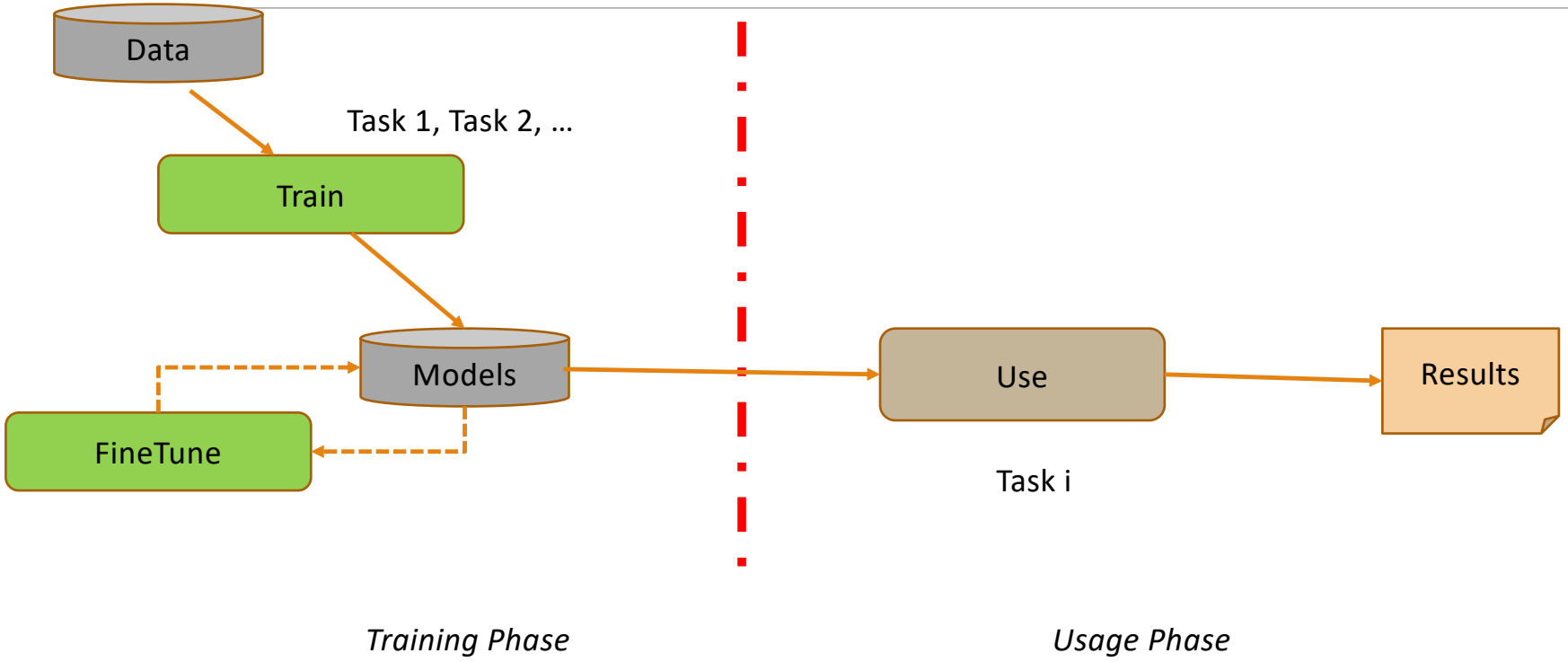
- **Extrinsic evaluation**: situate model in an application and evaluate the whole application for improvement. Also called in-vivo evaluation

- **Examples**:
  - BLEU and Rogue scores of generated text

Credit:
- https://medium.com/@sthanikamsanthosh1994/understanding-bleu-and-rouge-score-for-nlp-evaluation-1ab334ecadcb

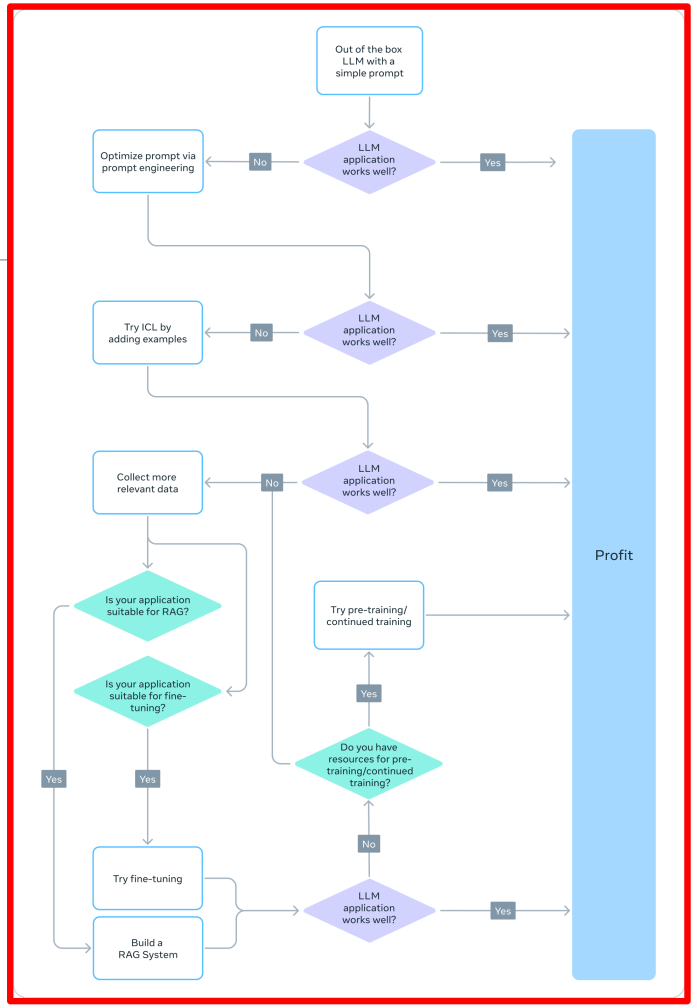
Large LMs (LLMs)

Large Language Models (LLMs) Basics



Improving Generative Output

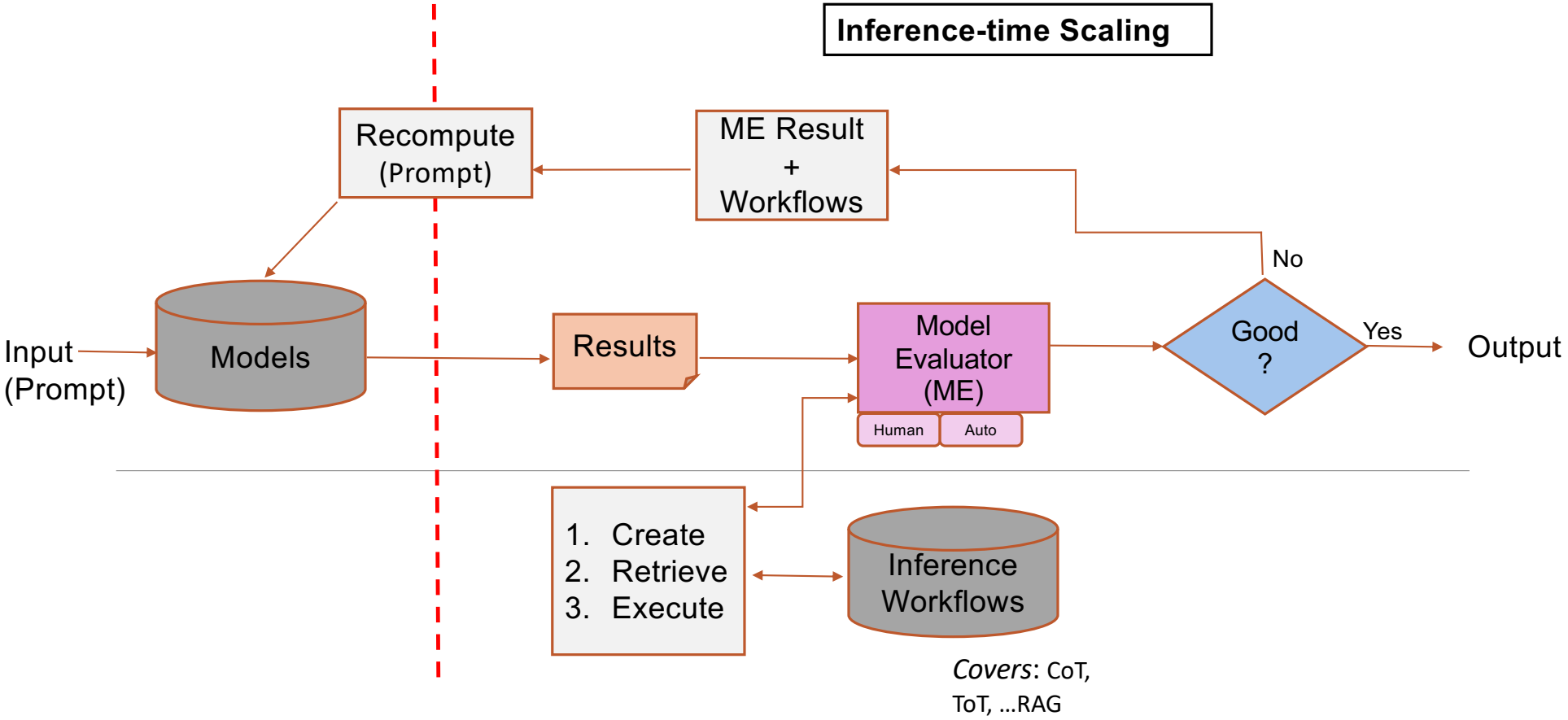
- Improving prompt
  - Prompt variants
  - In-context learning (ICL)
  - Retrieval augmented generation (RAG)
- Improving data and training
  - Finetuning
  - Continued pre-training



More reading: https://github.com/biplav-s/course-ai-f25/blob/main/reading-list/Readme-LLMs.md

Image Credit: https://ai.meta.com/blog/adapting-large-language-models-llms/

Inference Time with LLMs



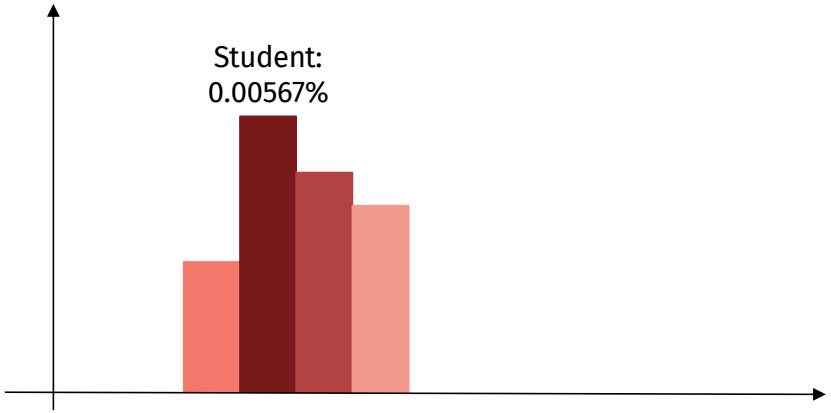
(Large) Language Modeling

Input Kaushik Roy is a PhD Student

Sequence	Kaushik	Roy	is	a	PhD	Student
	1	2	3	4	5	6

Student

Did you mean: Student
Did you mean: Student Square
Did you mean: Student Success



Credit: Kaushik Roy, CSCE 771 Guest Lectures

BERT - **B**idirectional **E**ncoder **R**epresentations from **T**ransformers

Learns with two tasks

- Predicting missing words in sentences
  - mask out 15% of the words in the input, predict the masked words.
- Given two sentences A and B, is B the actual next sentence that comes after A, or just a random sentence from the corpus?

(12-layer to 24-layer Transformer) on (Wikipedia + BookCorpus)

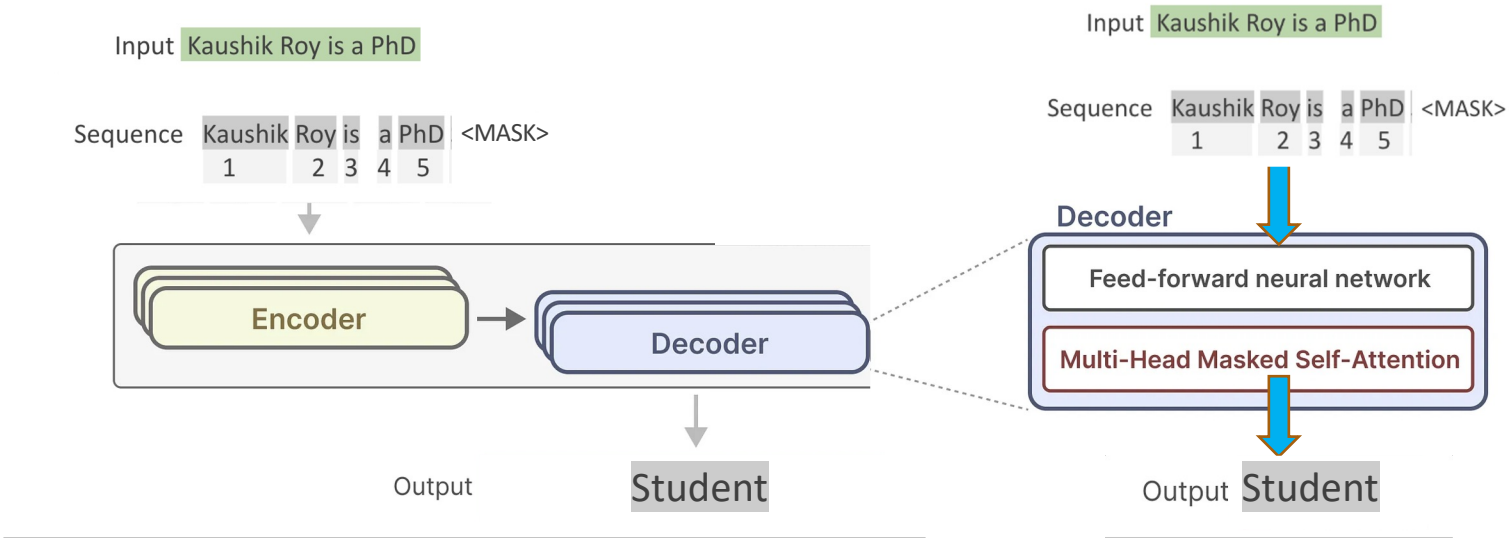
Input: the man went to the [MASK1] . he bought a [MASK2] of milk.
Labels: [MASK1] = store; [MASK2] = gallon

Sentence A: the man went to the store .
Sentence B: he bought a gallon of milk .
Label: IsNextSentence

Sentence A: the man went to the store .
Sentence B: penguins are flightless .
Label: NotNextSentence

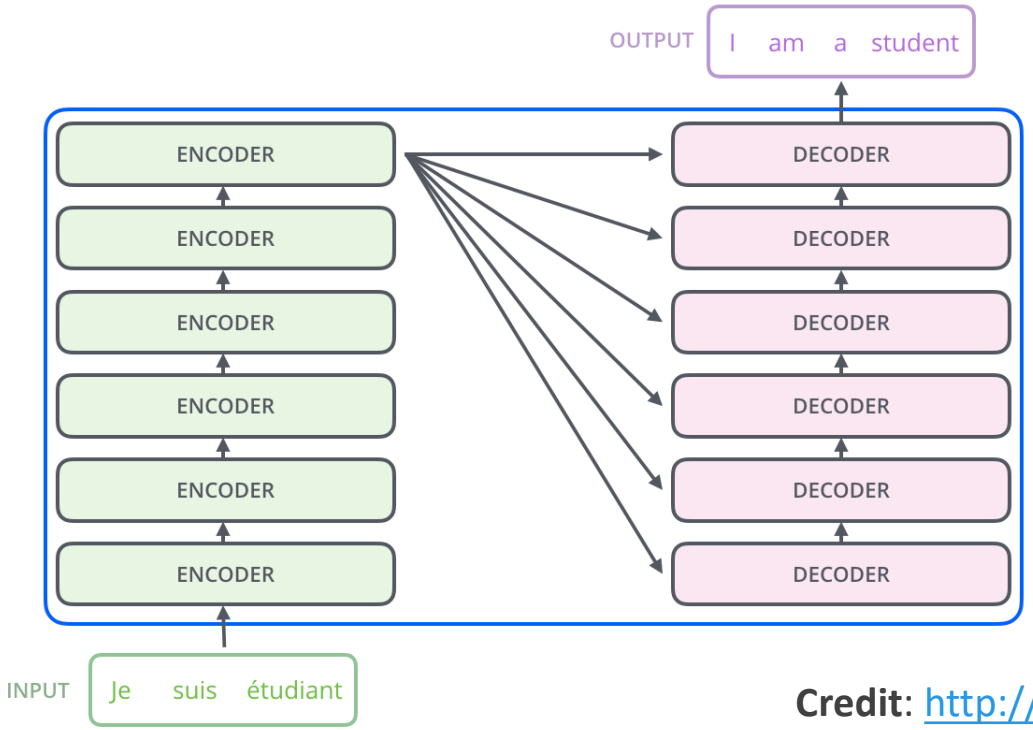
Credit and details: https://github.com/google-research/bert

BERT and family



Credit: Kaushik Roy, CSCE 771 Guest Lectures

Transformer: High-Level Idea

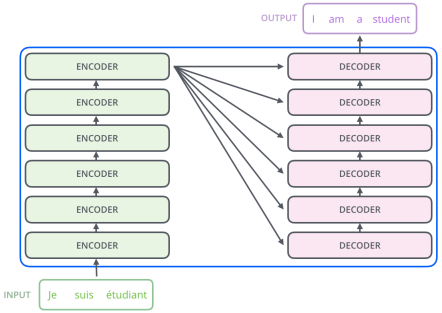


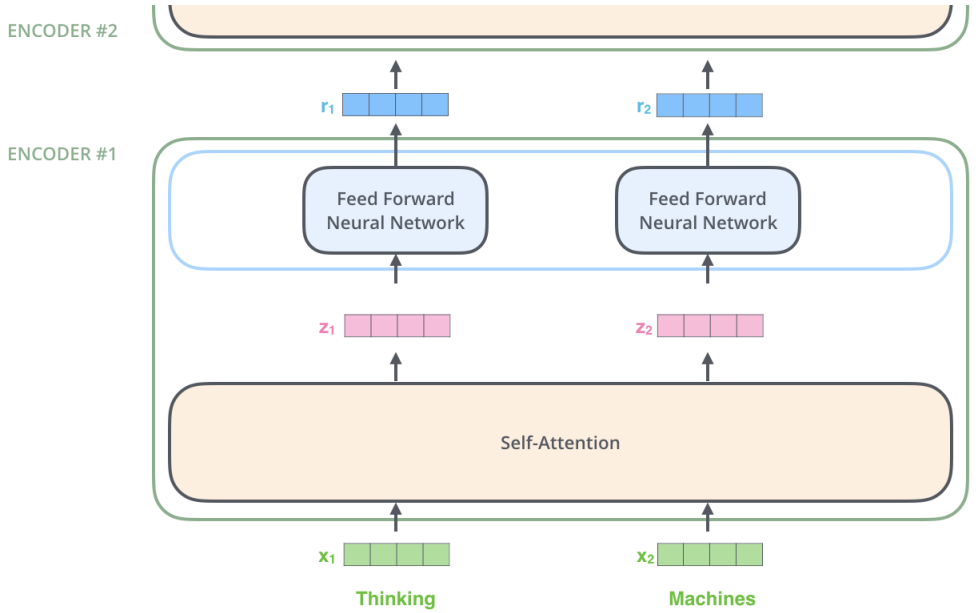
A block of encoders, and similar number of decoders

Credit: http://jalammar.github.io/illustrated-transformer/

Transformer: Encoder

An encoder

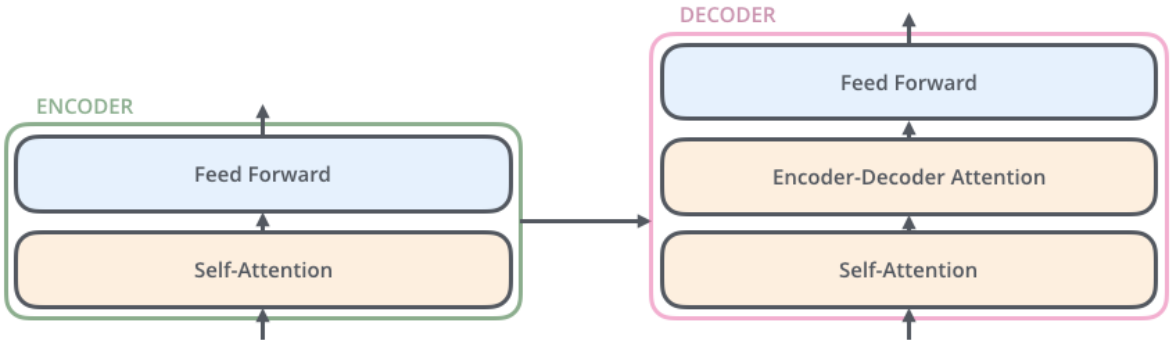


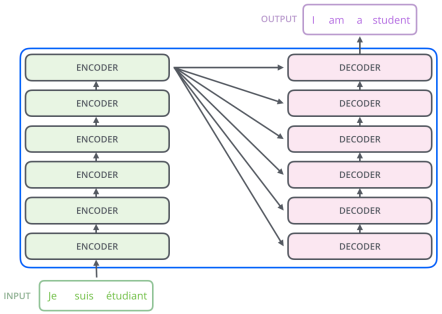


Credit: http://jalammar.github.io/illustrated-transformer/

Transformer: Decoder

A decoder



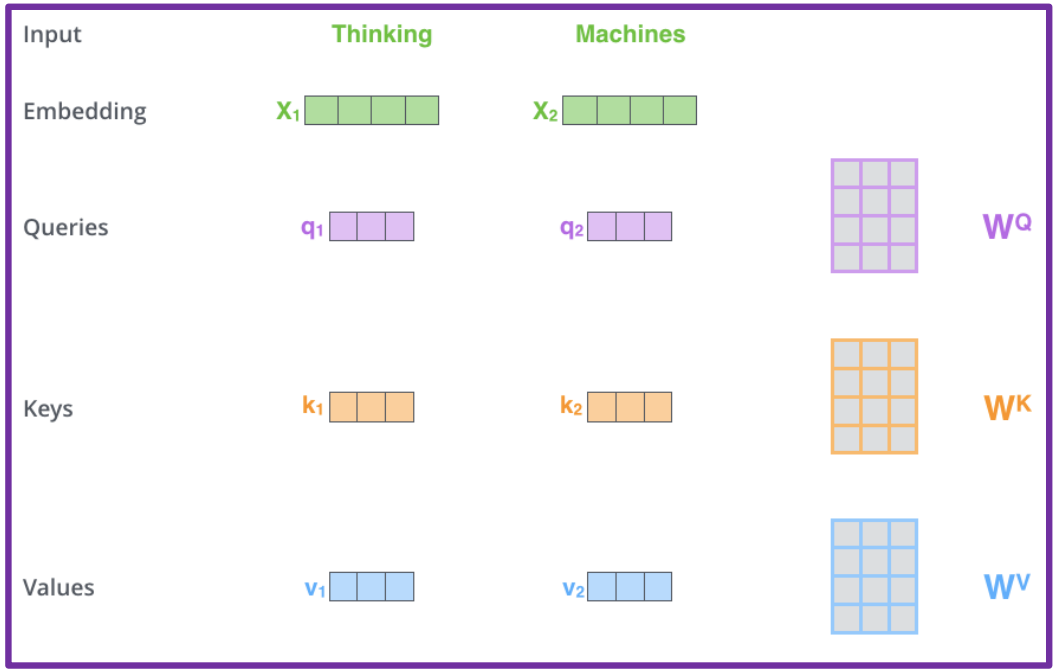


Credit: http://jalammar.github.io/illustrated-transformer/

Attention, Query, Key, Value

- **Attention**: concept (e.g., person, book, video)
- **Query**: expressed in input (e.g., words)
- **key**: features (e.g., names)
- **value**: (e.g., “Robert Redford”)

Transformer: Self-Attention



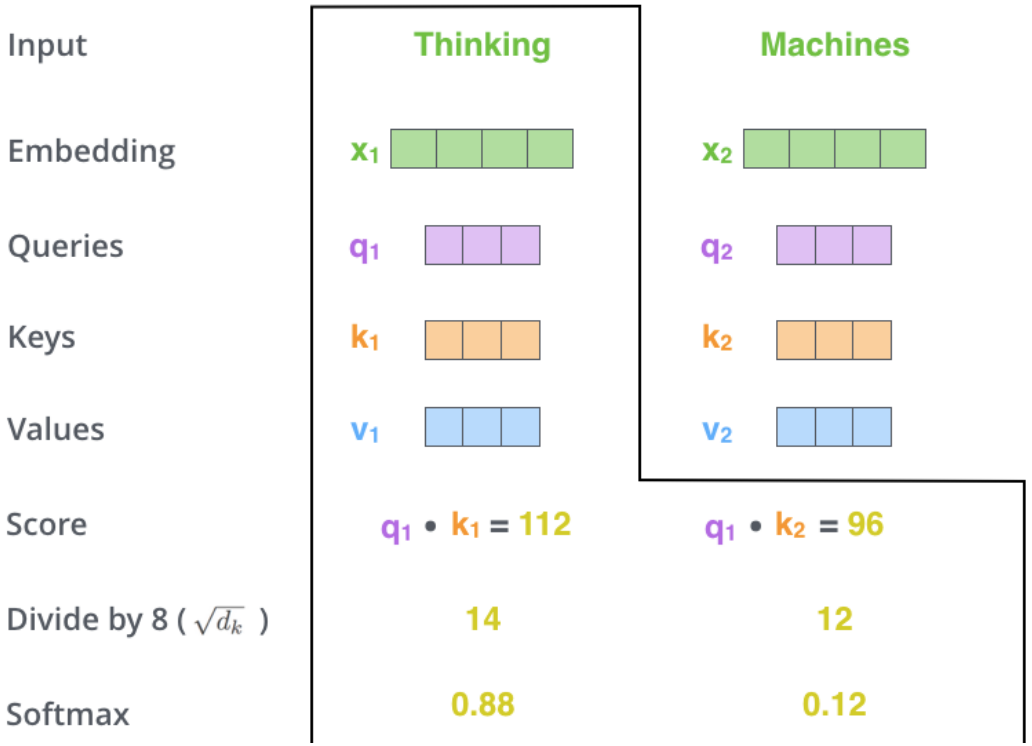
The **first step** in calculating self-attention is to create three vectors from each of the encoder's input vectors (for word). Size: 512 .

Multiplying x1 by the WQ weight matrix produces q1, the "query" vector associated with that word.

We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.

Credit: http://jalammar.github.io/illustrated-transformer/

Transformer: Self-Attention

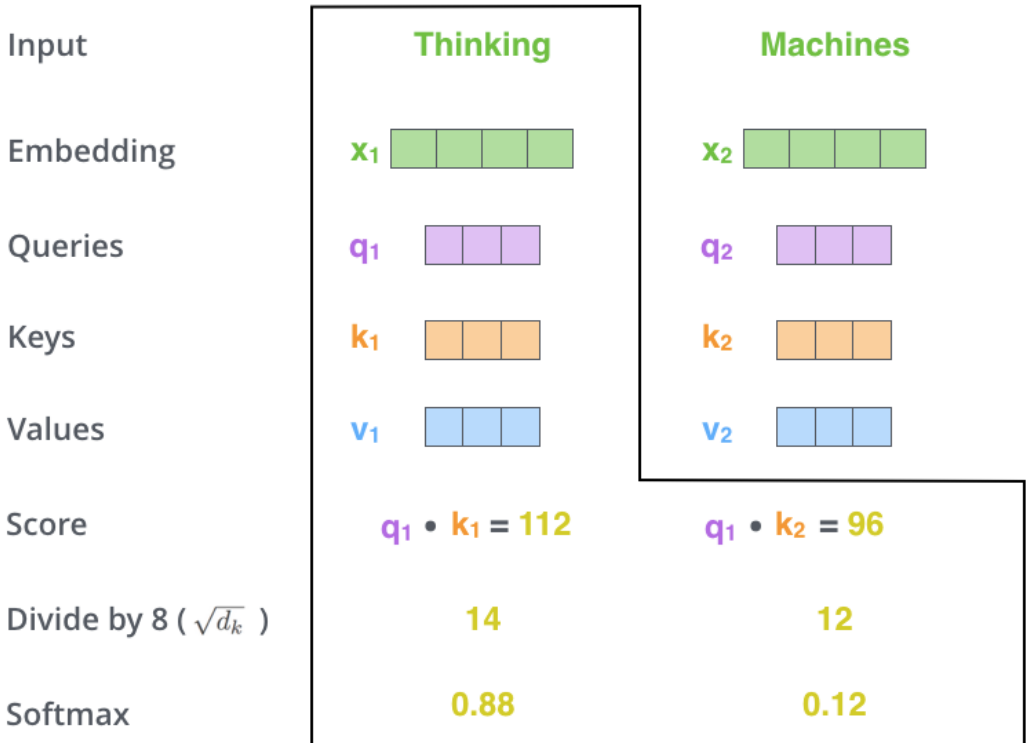


The **second step** in calculating self-attention is to calculate a score. The score is calculated by taking the dot product of the query vector with the key vector of the respective word we're scoring.

The **third and fourth steps** are to divide the scores by 8 (the square root of the dimension of the key vectors used in the paper – 64

Credit: http://jalammar.github.io/illustrated-transformer/

Transformer: Self-Attention

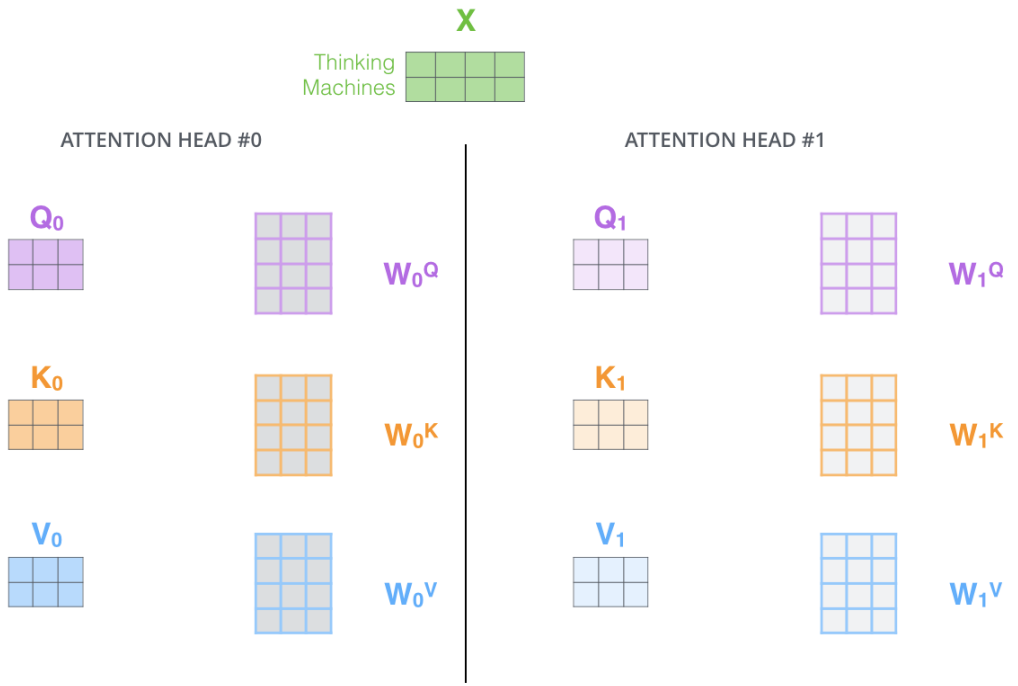


The **fifth step** is to multiply each value vector by the softmax score (in preparation to sum them up). The intuition here is to keep intact the values of the word(s) we want to focus on, and drown-out irrelevant word.

The **sixth step** is to sum up the weighted value vectors. This produces the output of the self-attention layer at this position (for the first word).

Credit: http://jalammar.github.io/illustrated-transformer/

Transformer: Multiple Heads

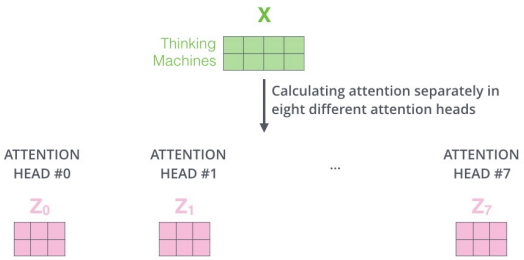


With multi-headed attention, we maintain separate Q/K/V weight matrices for each head resulting in different Q/K/V matrices.

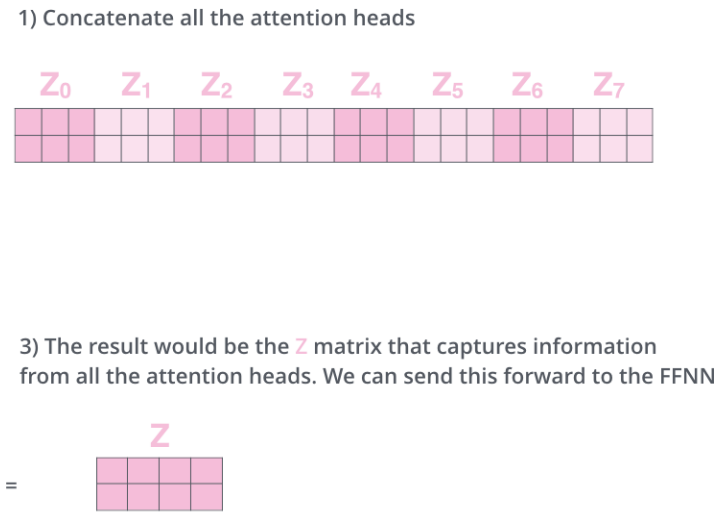
We multiply X by the WQ/WK/WV matrices to produce Q/K/V matrices.

Credit: http://jalammar.github.io/illustrated-transformer/

Transformer: Multiple Heads

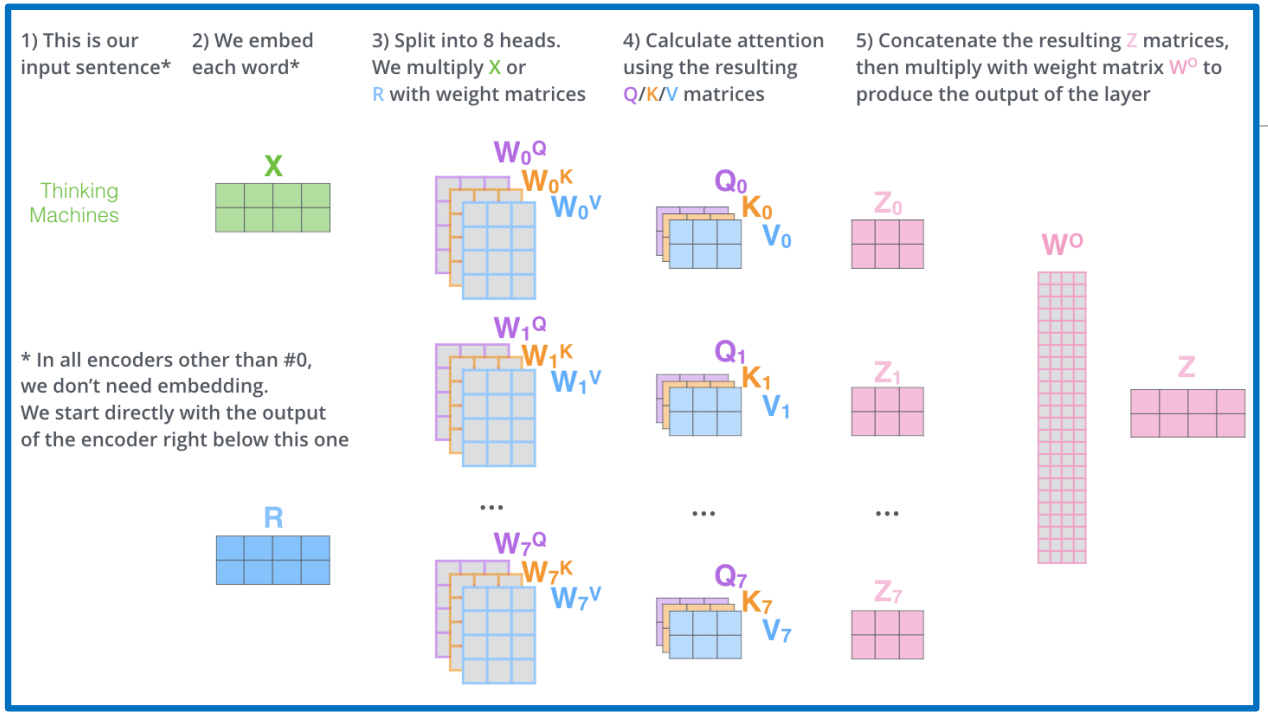


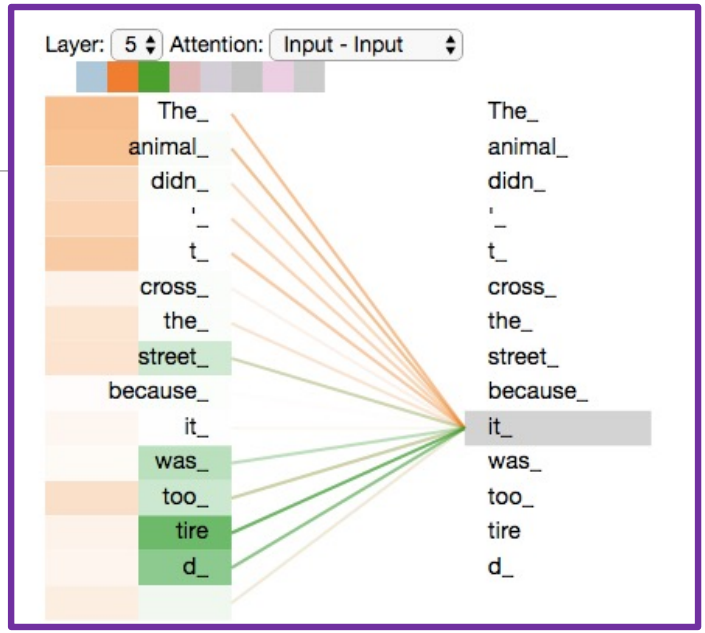
From multiple (=8) matrices to a single matrix (Z)



Credit: http://jalammar.github.io/illustrated-transformer/

Transformer

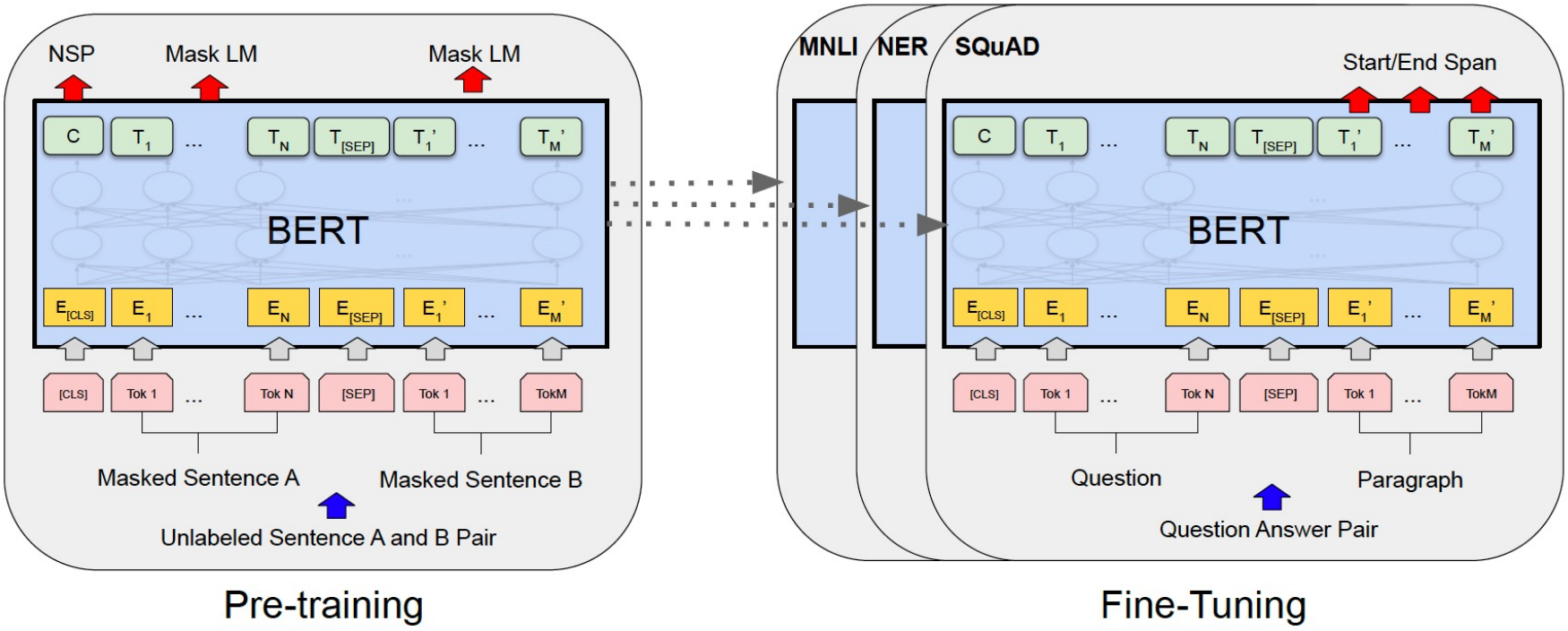




Credit: http://jalammar.github.io/illustrated-transformer/

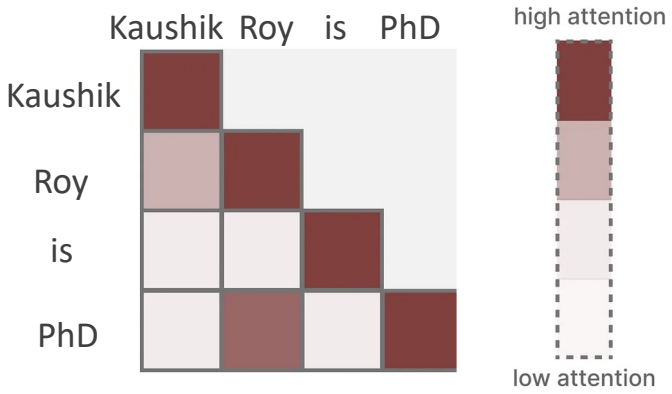
BERT: High-Level Idea

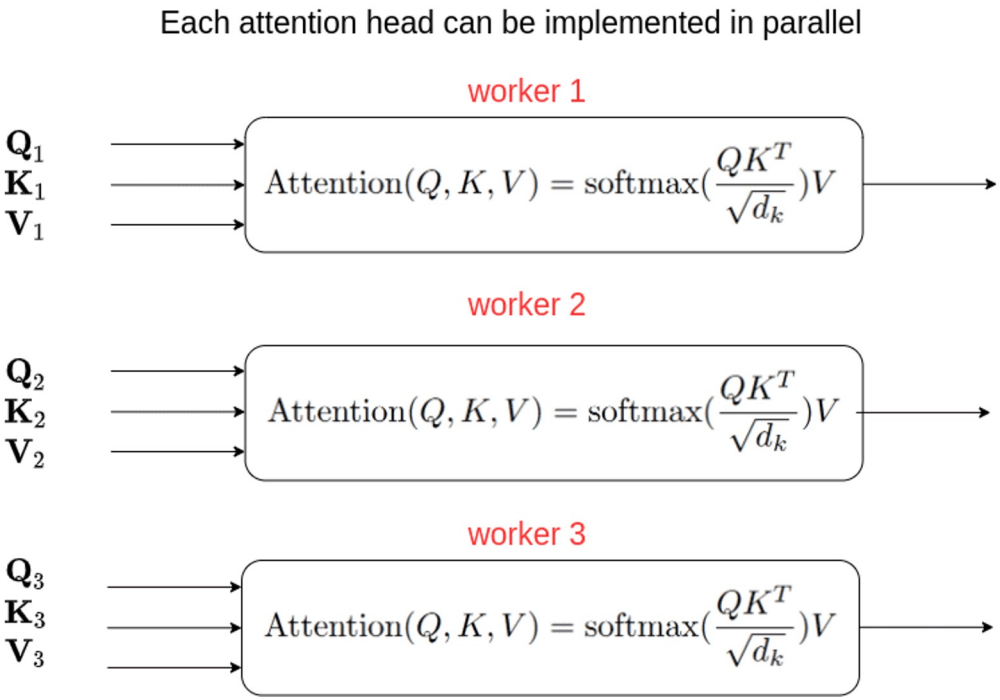
BERT: Before and During Usage



Credit and details: **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**
Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, 2018

BERT and family - Multi-headed Self-Attention





Credit: Kaushik Roy, CSCE 771 Guest Lectures

Self Attention Snippet and Live Coding - BERT from Scratch

```python
class SelfAttention(nn.Module):
    def __init__(self, embed_size, heads):
        super(SelfAttention, self).__init__()
        self.embed_size = embed_size
        self.heads = heads
        self.head_dim = embed_size // heads

        assert (
            self.head_dim * heads == embed_size
        ), "Embedding size needs to be divisible by heads"

        self.values = nn.Linear(self.head_dim, embed_size, bias=False)
        self.keys = nn.Linear(self.head_dim, embed_size, bias=False)
        self.queries = nn.Linear(self.head_dim, embed_size, bias=False)
        self.fc_out = nn.Linear(embed_size, embed_size)
```

Github: https://github.com/kauroy1994/CSCE-771-NLP-Class-11/tree/main (BERT-based CV Processing)

Credit: Kaushik Roy, CSCE 771 Guest Lectures

Live Coding - BERT using Libraries

- Use the transformers library to extract information from the CV

```python
import pdfplumber
from transformers import AutoTokenizer, AutoModelForTokenClassification
from transformers import pipeline

# Step 1: Load the CV PDF and extract text
def extract_text_from_pdf(pdf_path):
    with pdfplumber.open(pdf_path) as pdf:
        pages = [page.extract_text() for page in pdf.pages]
    return ''.join(pages)

# Extract text from the CV
pdf_path = "CV.pdf"
cv_text = extract_text_from_pdf(pdf_path)

# Step 2: Load pre-trained BERT model and tokenizer
tokenizer = AutoTokenizer.from_pretrained("dslim/bert-base-NER")
model = AutoModelForTokenClassification.from_pretrained("dslim/bert-base-NER")

# Step 3: Use pipeline for Named Entity Recognition
nlp = pipeline("ner", model=model, tokenizer=tokenizer, grouped_entities=True)

# Step 4: Extract entities from the CV text
ner_results = nlp(cv_text)

# Display the recognized entities
for entity in ner_results:
    print(f"Entity: {entity['word']}, Label: {entity['entity_group']}")

# Step 5: Post-process the entities for CV data extraction (optional)
# For example, grouping entities like degree, institution, and dates
def extract_education_details(ner_results):
    education = []
    current_education = {}
    for entity in ner_results:
        if entity['entity_group'] == 'ORG':
            current_education['institution'] = entity['word']
        elif entity['entity_group'] == 'MISC':  # Assuming degrees are labeled as MISC
            current_education['degree'] = entity['word']
        elif entity['entity_group'] == 'DATE':
            current_education['year'] = entity['word']

        # Save the current education entry
        if 'institution' in current_education and 'degree' in current_education and 'year' in current_education:
            education.append(current_education)
            current_education = {}
    return education

education_details = extract_education_details(ner_results)

# Display the structured education data
print("Extracted Education Details:", education_details)
```

Github: https://github.com/kauroy1994/CSCE-771-NLP-Class-11/tree/main (BERT-based CV Processing)

Credit: Kaushik Roy, CSCE 771 Guest Lectures

Transformer

- RNN/ LSTM with
  - Attention
    - attention layer can access all previous states and weighs them according to some learned measure of relevancy to the current token, providing sharper information about far-away relevant tokens
    - **Query** vector, **Key** vector, and **Value** vectors introduced during encoding and decoding phase
  - Parallelization of learning
  - See Dr. Amitava Das's slide for Attention/ BERT video
    - https://prezi.com/view/amx5hBo8UhMOn1rPyJ02/

Source and details: https://en.wikipedia.org/wiki/Transformer_(machine_learning_model), http://jalammar.github.io/illustrated-transformer/

Major LM Types

- Large
  - Large training dataset
  - Large number of parameters
- General purpose
  - Commonality of human languages
  - Resource restriction
- Pre-trained and fine-tuned

Credits: Google Cloud Skills Boost

LLMs have three different architectures - (a) encoder-only, (b) decoder-only, and (c) encoder-decoder, each with their own benefits.

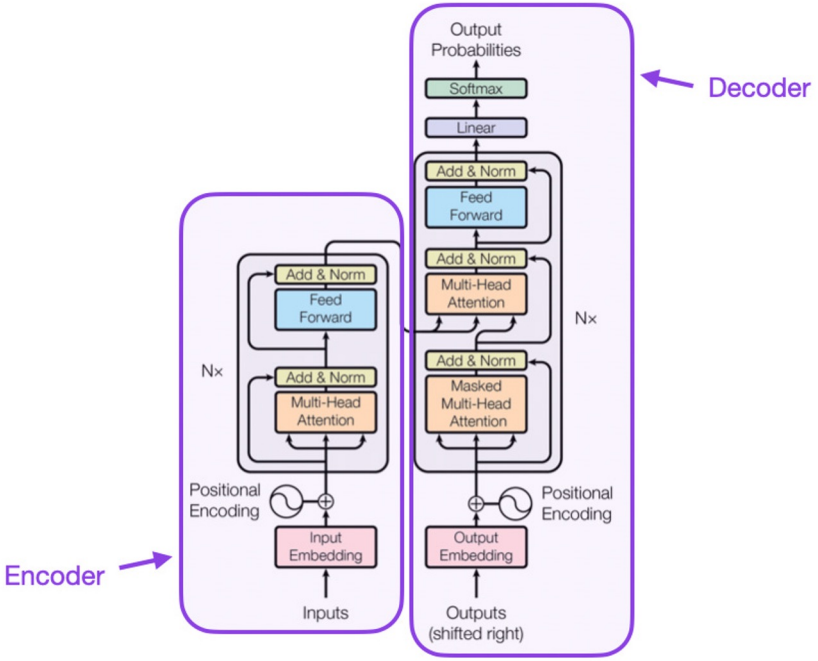
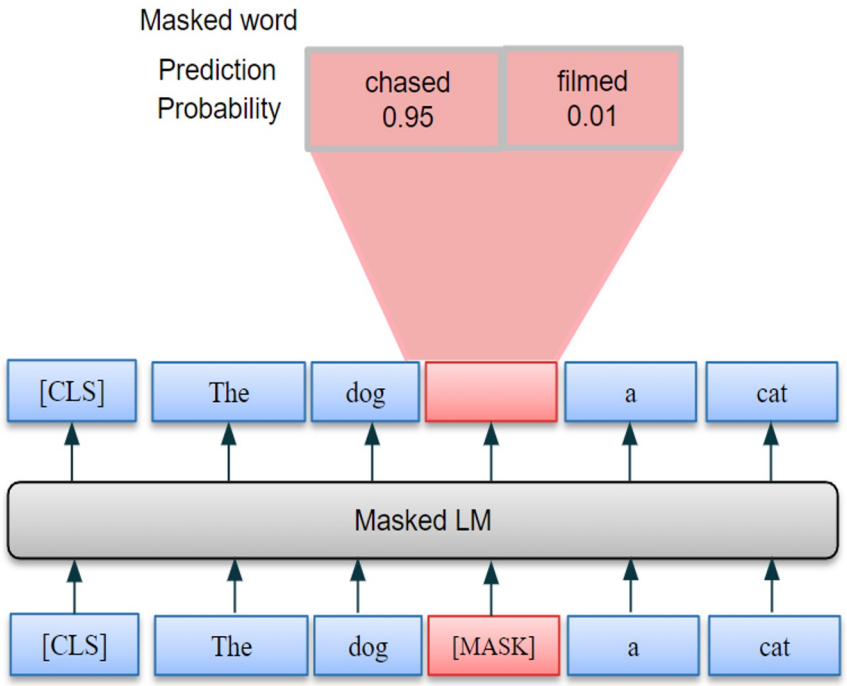


Figure. The Transformer - model architecture.

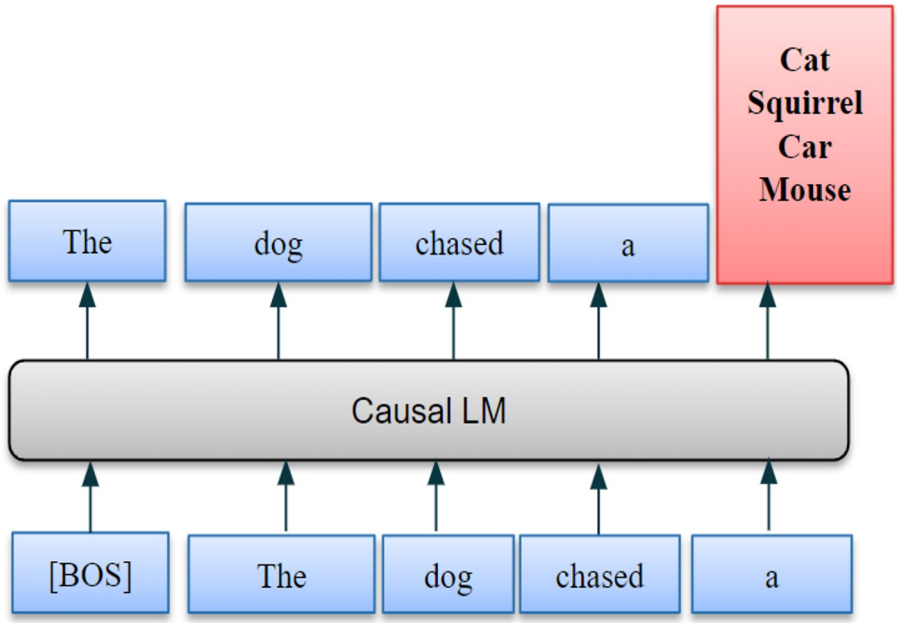
Encoder-only

- Encoder-only architectures are trained to understand the bidirectional context by predicting words randomly masked in a sentence.
- **Example:** BERT
- **Effective for:** sentiment analysis, classification, entailment.



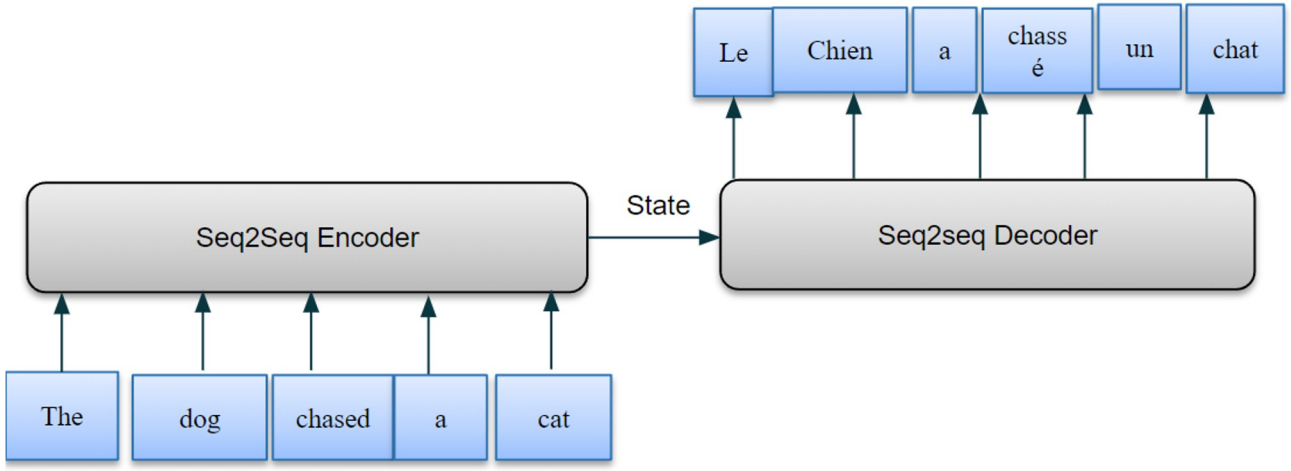
Decoder-only

- Decoder-only architectures are designed for tasks where text generation is sequential and dependent on the preceding context.
- They predict each subsequent word based on the preceding words, modeling the probability of a word sequence in a forward direction.
- **Example:** GPT-4, Llama series, Claude, Vicuna.
- **Effective for:** Content generation.



Encoder-Decoder

- Encoder-Decoder architectures are designed to transform an input sequence into a related output sequence.
- **Example:** T5, CodeT5, FlanT5
- **Effective for:** summarization, language translation.



Summary: Large Language Models (LLMs)

- **Large Language Models**.
  - **What are LLMs?**: Large Language Models are deep learning models trained on massive amounts of text data to understand and generate human-like language.
  - **Importance**: LLMs power many modern AI applications such as chatbots, machine translation, text summarization, and code generation.
- **Example Models**:
  - **GPT-3**, with 175 billion parameters, is a well-known example of an LLM used in tools like OpenAI's ChatGPT.
  - **Llama, Mixtral, Gemma,** by Meta, Mistral, and Google, respectively.
- **Key Advantages**:
  - Understanding context across long sequences of text.
  - Generating coherent and contextually accurate language outputs.

Autoregressive Models

- **Autoregressive Architecture**:
  - **How GPT works**: GPT predicts the next word in a sequence based on the words that came before it. This makes it **unidirectional** or **autoregressive**.
  - **Transformer Backbone**: GPT uses transformer layers to model long-range dependencies between words, which is a significant improvement over older models like RNNs and LSTMs.
- **Key Features**:
  - **Self-Attention Mechanism**: GPT uses self-attention to focus on important words in a sequence, allowing it to learn context.
  - **Scaling GPT**: GPT-2 had 1.5 billion parameters, and GPT-3 scaled up to 175 billion parameters, greatly improving performance on a wide range of tasks.
- **GPT Variants**:
  - **GPT-2**: 1.5 billion parameters, trained on 8 million web pages.
  - **GPT-3**: 175 billion parameters, trained on hundreds of billions of tokens.

Exercise for Class

Task	Traditional (NLP) Method	LLM-Based (Programmatic)	LLM-Based (UI/ Chat)
Entity Extraction	Regex, …	Distilbert	GPT/ChatGPT
Sentiment Detection	Vader, TextBlob, …	Distilbert	GPT/ChatGPT
…			

- **Entity**
  - Regex based: https://github.com/biplav-s/course-nl-f22/blob/main/sample-code/l17-eventextr/SimpleEntitySearch.ipynb
  - Using LLM: https://github.com/biplav-s/course-tai-s25/blob/main/sample-code/LLM(Distilbert)-Entity%20Recognition.ipynb
- **Sentiment**
  - Using lexicon-based methods, https://github.com/biplav-s/course-d2d-ai/blob/7f90f154729115a31f449702dbdf84d63be7a844/sample-code/l23-textrepresent/Basic%20Sentiment.ipynb
  - Using Language Models, https://github.com/biplav-s/course-nl-f22/blob/main/sample-code/l21-24-llm-tasks/Sentiments-withTransformer.ipynb

The **Nobel Peace Prize** is one of the five **Nobel Prizes** established by the **will** of Swedish industrialist, inventor, and armaments manufacturer **Alfred Nobel**, along with the prizes in Chemistry, Physics, Physiology or Medicine, and Literature

Credit: From Wikipedia

Lecture 9: Concluding Comments

We discussed on

- NN archs: RNNs/ LSTMS
- LMs
  - Evaluation: perplexity
- LLMs
  - Transformers
  - BERT

Lecture 10: Using LLMs, Trust Issues

Lecture 10 Outline

- Revisit Transformer discussion (slides 40-50)
- Major LLM Types
- Using LLMs v/s traditional methods (entity recognition, sentiment detection)
- Trust of technology, AI and with LLMs
- Prompting for gaining explanations

Using BERT in Practice – Huggingface Libraries

- Transformers – https://github.com/huggingface/transformers
- APIs to download and use pre-trained models, fine-tune them on own datasets and tasks
  - Code Sample

```
# Loading BERT
model_class, tokenizer_class, pretrained_weights =  (ppb.DistilBertModel, ppb.DistilBertTokenizer, 'distilbert-base-uncased')

# Load pretrained model/tokenizer
tokenizer = tokenizer_class.from_pretrained(pretrained_weights)
model = model_class.from_pretrained(pretrained_weights)
```

- Provides pretrained models in 100+ languages.
- Use with popular deep learning libraries, PyTorch and TensorFlow,
  - Possible to train / fine-tune models with one, and load it for inference with another

Using BERT in Practice – Huggingface Libraries

- DistilBERT
  - Details: https://medium.com/huggingface/distilbert-8cf3380435b5
  - Teacher-student learning, also called model distillation
    - Teacher: bert-base-uncased
    - Student: dstilBERT - BERT without *the token-type embeddings and the pooler* , and half the layers
  - **"DistilBERT,** has **about half** the total number of parameters of BERT base and retains 95% of BERT's performances on the language understanding benchmark GLUE"

- Sample code of usage for sentiment classification: https://github.com/biplav-s/course-nl/blob/master/l12-langmodel/UsingLanguageModel.ipynb
- Also see: https://huggingface.co/blog/sentiment-analysis-python

HF / DistilBERT and LLM Project

- **Resources**: https://huggingface.co/docs/transformers/en/model_doc/distilbert
- Create notebooks for
  - Data cleaning
  - LLM finetuning
  - Conducting (task) evaluation on
    - LLM
    - Fine-tuned LLM
    - GPT (ChatGPT

Exercise for Class

Task	Traditional (NLP) Method	LLM-Based (Programmatic)	LLM-Based (UI/ Chat)
Entity Extraction	Regex, …	Distilbert	GPT/ChatGPT
Sentiment Detection	Vader, TextBlob, …	Distilbert	GPT/ChatGPT
…			

- **Entity**
  - Regex based: https://github.com/biplav-s/course-nl-f22/blob/main/sample-code/l17-eventextr/SimpleEntitySearch.ipynb
  - Using LLM: https://github.com/biplav-s/course-tai-s25/blob/main/sample-code/LLM(Distilbert)-Entity%20Recognition.ipynb
- **Sentiment**
  - Using lexicon-based methods, https://github.com/biplav-s/course-d2d-ai/blob/7f90f154729115a31f449702dbdf84d63be7a844/sample-code/l23-textrepresent/Basic%20Sentiment.ipynb
  - Using Language Models, https://github.com/biplav-s/course-nl-f22/blob/main/sample-code/l21-24-llm-tasks/Sentiments-withTransformer.ipynb

The **Nobel Peace Prize** is one of the five **Nobel Prizes** established by the **will** of Swedish industrialist, inventor, and armaments manufacturer **Alfred Nobel**, along with the prizes in Chemistry, Physics, Physiology or Medicine, and Literature

Credit: From Wikipedia

Example Pre-Trained Models

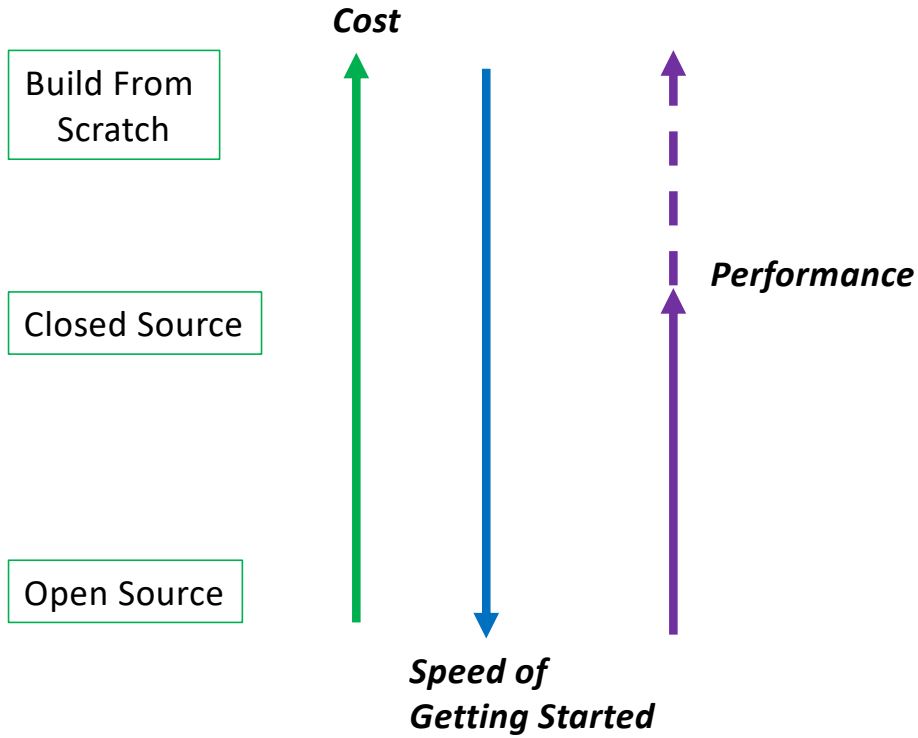
1. ALBERT (from Google Research and the Toyota Technological Institute at Chicago) released with the paper ALBERT: A Lite BERT for Self-supervised Learning of Language Representations, by Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, Radu Soricut.
2. BART (from Facebook) released with the paper BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension by Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov and Luke Zettlemoyer.
3. BERT (from Google) released with the paper BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding by Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova.
4. BERT For Sequence Generation (from Google) released with the paper Leveraging Pre-trained Checkpoints for Sequence Generation Tasks by Sascha Rothe, Shashi Narayan, Aliaksei Severyn.
5. CamemBERT (from Inria/Facebook/Sorbonne) released with the paper CamemBERT: a Tasty French Language Model by Louis Martin*, Benjamin Muller*, Pedro Javier Ortiz Suárez*, Yoann Dupont, Laurent Romary, Éric Villemonte de la Clergerie, Djamé Seddah and Benoît Sagot.
6. CTRL (from Salesforce) released with the paper CTRL: A Conditional Transformer Language Model for Controllable Generation by Nitish Shirish Keskar*, Bryan McCann*, Lav R. Varshney, Caiming Xiong and Richard Socher.
7. DeBERTa (from Microsoft Research) released with the paper DeBERTa: Decoding-enhanced BERT with Disentangled Attention by Pengcheng He, Xiaodong Liu, Jianfeng Gao, Weizhu Chen.
8. DialoGPT (from Microsoft Research) released with the paper DialoGPT: Large-Scale Generative Pre-training for Conversational Response Generation by Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, Bill Dolan.
9. DistilBERT (from HuggingFace), released together with the paper DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter by Victor Sanh, Lysandre Debut and Thomas Wolf. The same method has been applied to compress GPT2 into DistilGPT2, RoBERTa into DistilRoBERTa, Multilingual BERT into DistilmBERT and a German version of DistilBERT.
10. DPR (from Facebook) released with the paper Dense Passage Retrieval for Open-Domain Question Answering by Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih.
11. ELECTRA (from Google Research/Stanford University) released with the paper ELECTRA: Pre-training text encoders as discriminators rather than generators by Kevin Clark, Minh-Thang Luong, Quoc V. Le, Christopher D. Manning.
12. FlauBERT (from CNRS) released with the paper FlauBERT: Unsupervised Language Model Pre-training for French by Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoît Crabbé, Laurent Besacier, Didier Schwab.
13. Funnel Transformer (from CMU/Google Brain) released with the paper Funnel-Transformer: Filtering out Sequential Redundancy for Efficient Language Processing by Zihang Dai, Guokun Lai, Yiming Yang, Quoc V. Le.
14. GPT (from OpenAI) released with the paper Improving Language Understanding by Generative Pre-Training by Alec Radford, Karthik Narasimhan, Tim Salimans and Ilya Sutskever.
15. GPT-2 (from OpenAI) released with the paper Language Models are Unsupervised Multitask Learners by Alec Radford*, Jeffrey Wu*, Rewon Child, David Luan, Dario Amodei** and Ilya Sutskever**.
16. LayoutLM (from Microsoft Research Asia) released with the paper LayoutLM: Pre-training of Text and Layout for Document Image Understanding by Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, Ming Zhou.
17. Longformer (from AllenAI) released with the paper Longformer: The Long-Document Transformer by Iz Beltagy, Matthew E. Peters, Arman Cohan.
18. LXMERT (from UNC Chapel Hill) released with the paper LXMERT: Learning Cross-Modality Encoder Representations from Transformers for Open-Domain Question Answering by Hao Tan and Mohit Bansal.
19. MarianMT Machine translation models trained using OPUS data by Jörg Tiedemann. The Marian Framework is being developed by the Microsoft Translator Team.
20. MBart (from Facebook) released with the paper Multilingual Denoising Pre-training for Neural Machine Translation by Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, Luke Zettlemoyer.
21. MMBT (from Facebook), released together with the paper a Supervised Multimodal Bitransformers for Classifying Images and Text by Douwe Kiela, Suvrat Bhooshan, Hamed Firooz, Davide Testuggine.
22. Pegasus (from Google) released with the paper PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization> by Jingqing Zhang, Yao Zhao, Mohammad Saleh and Peter J. Liu.
23. Reformer (from Google Research) released with the paper Reformer: The Efficient Transformer by Nikita Kitaev, Łukasz Kaiser, Anselm Levskaya.
24. RoBERTa (from Facebook), released together with the paper a Robustly Optimized BERT Pretraining Approach by Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov. ultilingual BERT into DistilmBERT and a German version of DistilBERT.
25. SqueezeBert released with the paper SqueezeBERT: What can computer vision teach NLP about efficient neural networks? by Forrest N. Iandola, Albert E. Shaw, Ravi Krishna, and Kurt W. Keutzer.
26. T5 (from Google AI) released with the paper Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer by Colin Raffel and Noam Shazeer and Adam Roberts and Katherine Lee and Sharan Narang and Michael Matena and Yanqi Zhou and Wei Li and Peter J. Liu.
27. Transformer-XL (from Google/CMU) released with the paper Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context by Zihang Dai*, Zhilin Yang*, Yiming Yang, Jaime Carbonell, Quoc V. Le, Ruslan Salakhutdinov.
28. XLM (from Facebook) released together with the paper Cross-lingual Language Model Pretraining by Guillaume Lample and Alexis Conneau.
29. XLM-RoBERTa (from Facebook AI), released together with the paper Unsupervised Cross-lingual Representation Learning at Scale by Alexis Conneau*, Kartikay Khandelwal*, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer and Veselin Stoyanov.
30. XLNet (from Google/CMU) released with the paper XLNet: Generalized Autoregressive Pretraining for Language Understanding by Zhilin Yang*, Zihang Dai*, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, Quoc V. Le.

https://github.com/huggingface/transformers

Creating and Using One's Own LLMs

Practical Considerations

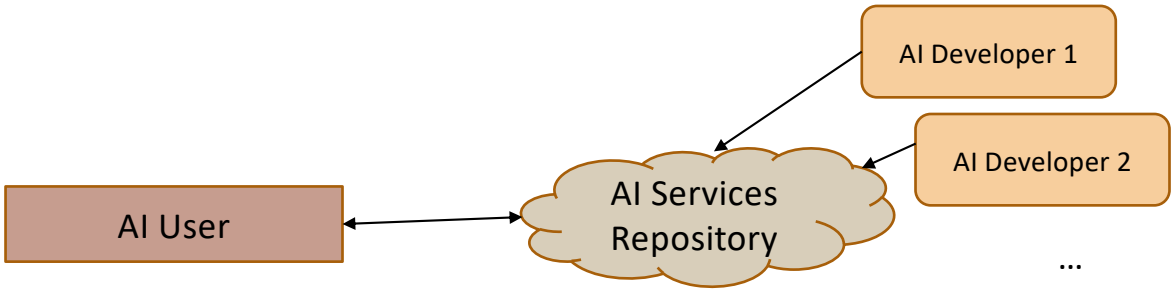
- How much can you pay?
  - Free open-source models, i.e., weights are available for download (Llama, Mistral, …) v/s Closed models (GPT, Gemini, Claude, …)
- Can you afford to share your data? Yes/ No
- Will others be using your model? Yes / No



Options

- Assumption: Already tried a pre-trained model and know performance on one's tasks
- Option 1: Fine-tune a pretrained model
- Option 2: Use someone-else's pretrained model
  - Creating mini-GPT (OpenAI): https://help.openai.com/en/articles/8554397-creating-a-gpt
- Option 3: Build one's own on specialized data, tasks and optimizing performance metrics of interest

AI: The Problem of Trust



Components of Trust - Illustration

1. Competent – does what it is supposed to do
2. Reliable – including, well tested
3. Upholds human values
    1. Fairly and ethically used
    2. Adequate data management & preserves privacy
4. Allows human-technology interaction
    1. Explainable, transparent
    2. How does the system give its result?

	Car – cruise control	Nuclear Energy
Competent	X	X
Reliable	X	X
Upholds human values	-	?
Allows human interaction	X	-

x: yes; -: not applicable; ?: questionable

Instability of AI is Well Recorded

[Text] Su Lin Blodgett, Solon Barocas, Hal Daumé III, Hanna Wallach, Language (Technology) is Power: A Critical Survey of "Bias" in NLP, Arxiv - https://arxiv.org/abs/2005.14050, 2020 [NLP Bias]

[Image] Vegard Antun, Francesco Renna, Clarice Poon, Ben Adcock, and Anders C. Hansen, On instabilities of deep learning in image reconstruction and the potential costs of AI, https://doi.org/10.1073/pnas.1907377117, PNAS, 2020

[Audio] Allison Koenecke, Andrew Nam, Emily Lake, Joe Nudell, Minnie Quartey, Zion Mengesha, Connor Toups, John R. Rickford, Dan Jurafsky, and Sharad Goel, Racial disparities in automated speech recognition, PNAS April 7, 2020 117 (14) 7684-7689, https://doi.org/10.1073/pnas.1915768117, March 23, 2020

Explanation in Neural Network

- Visualize neural network activation for explanation
  - Illustration via code (Credit: Kausik Lakkaraju) - https://colab.research.google.com/drive/1ABWbuQ09qLJzITkKtYrMGUNCw-inQPDn?usp=sharing
- Mechanistic explanation
  - Mechanistic Interpretability for AI Safety -- A Review, Leonard Bereska, Efstratios Gavves, https://arxiv.org/abs/2404.14082
  - Illustrative code samples:
    - https://www.neelnanda.io/mechanistic-interpretability/quickstart
    - https://transformerlensorg.github.io/TransformerLens/content/getting_started_mech_interp.html

Explanation in LLMs

- Setting: one prompts a LLM and gets an output. How does one explain the result?
- Ideas
  - Prompt again with variation; analyze output, infer
    - Question: how to create variations?
    - Question: how to analyze outputs?
    - Question: how to infer about LLMs understanding?
  - Prompt another LLM, analyze output, infer
    - Question: which other LLM(s)
    - …
- Prompting methods
  - https://www.promptingguide.ai/introduction
  - See reading list on LLMs (https://github.com/biplav-s/course-tai-s25/blob/main/reading-list/Readme-LLMs.md)

Prompting of LLMs

A prompt contains any of the following elements:

- **Instruction** - a specific task or instruction you want the model to perform
- **Context** - external information or additional context that can steer the model to better responses
- **Input Data** - the input or question that we are interested to find a response for
- **Output Indicator** - the type or format of the output.

Credit:
https://www.promptingguide.ai/introduction/elements

Advanced reading:
The Prompt Report: A Systematic Survey of Prompting Techniques (June 2024, Feb 2025)
https://arxiv.org/pdf/2406.06608

Lecture 8: Concluding Comments

- We talked about
  - LLMs
  - Usage
  - Trust Issues

Week 5: Concluding Comments

We talked about

- Transformers (slides 40-50)
- Major LLM Types
- Using LLMs v/s traditional methods (entity recognition, sentiment detection)
- Trust of technology, AI and with LLMs
- Prompting for gaining explanations

- Week 1: Introduction, Aim: Chatbot / Intelligence Agent
- Weeks 2-3: Data: Formats, Representation and the Trust Problem
- Week 3: Machine Learning – Supervised (Classification)
- Week 4: Machine Learning - Unsupervised (Clustering) –
- Topic 5: Learning neural network, deep learning, Adversarial attacks
- Week 6: Large Language Models – Representation and Usage issues
- Weeks 7-8: Search, Heuristics - Decision Making
- Week 9: Constraints, Optimization – Decision Making
- Topic 10: Markov Decision Processes, Hidden Markov models - Decision making
- Topic 11-12: Planning, Reinforcement Learning – Sequential decision making
- Week 13: Trustworthy Decision Making: Explanation, AI testing
- Week 14: AI for Real World: Tools, Emerging Standards and Laws; Safe AI/ Chatbots

Upcoming Evaluation Milestones

- Projects B: Sep 30 – Nov 20
- Quiz 2: Oct 7
- Quiz 3: Nov 11
- Paper presentation (grad students only) : Nov 18

About Week 6 – Lectures 11 and 12

Week 6 – Lectures 11 and 12

- L11: Overcoming ML Trust Issues – Explainability, Rating
- L12: Representation and Logic - Propositional

- Week 1: Introduction, Aim: Chatbot / Intelligence Agent
- Weeks 2: Data: Formats, Representation, ML Basics
- Week 3: Machine Learning – Supervised (Classification)
- Week 4: Machine Learning - Unsupervised (Clustering) –
- Topic 5: Learning neural network, deep learning, Adversarial attacks
- Week 6: Large Language Models – Representation and Usage issues
- Weeks 7-8: Search, Heuristics - Decision Making
- Week 9: Constraints, Optimization – Decision Making
- Topic 10: Markov Decision Processes, Hidden Markov models - Decision making
- Topic 11-12: Planning, Reinforcement Learning – Sequential decision making
- Week 13: Trustworthy Decision Making: Explanation, AI testing
- Week 14: AI for Real World: Tools, Emerging Standards and Laws; Safe AI/ Chatbots

Note: exact schedule changes slightly to accommodate for exams and holidays.