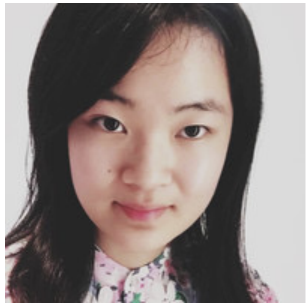
CPSC 670: Topics in Natural Language Processing

Yale University

SPRING 2023

Logistics



- Instructor: Arman Cohan
- Teaching Assistant: Simeng Han

- Time: Tuesdays and Thursdays 2:30pm-3:45pm
- Location: William L. Harkness Hall 120

- Office hours:
  - Arman’s office hours: Tuesday 4pm-5pm (appointment-based)
  - Simeng’s office hours: Friday 2:30-3:30pm

Course structure

- This a seminar course.
  - The course is primarily based on presentations & discussion of latest research papers

Course structure

- This a seminar course.
  - The course is primarily based on presentations & discussion of latest research papers

- Main goals of the course:
  - Get students up to speed with the latest developments in NLP
  - Prepare students to perform cutting-edge research in NLP
  - Help students build or improve research skills (from literature reviews and critiquing prior work, to brainstorming ideas and implementing them).

Course structure

- This a seminar course.
  - The course is primarily based on presentations & discussion of latest research papers
- Main goals of the course:
  - Get students up to speed with the latest developments in NLP
  - Prepare students to perform cutting-edge research in NLP
  - Help students build or improve research skills (from literature reviews and critiquing prior work, to brainstorming ideas and implementing them).
- All students are expected to participate in the class regularly and participate in presentations and discussions

Course structure - Prerequisites

- Students should feel comfortable with Machine Learning
  - Fundamentals of ML including supervised learning, linear models, basic neural networks
  - Training ML models including gradient based learning and backpropagation
- Having taken a Machine Learning (or similar) course is highly encouraged. Examples:
  - CPSC 452/ 552 Deep Learning Theory and Applications
  - CPSC 477/577 Natural Language Processing
  - CPSC 677 Advanced NLP
  - Or other related undergrad courses

Course structure - Prerequisites

- Familiarity with Natural Language Processing is a plus but not necessary

- Experience with reading research papers in NLP/ML:
  - Ability to read NLP research papers and develop a decent understanding of the paper

- Experience with implementing basic ML algorithms, understanding open-source code-base, setting up and running basic experiments

Course structure - Resources

- No required textbook. But if you are interested in textbooks or book chapters:
  - Natural Language Processing with Transformers https://transformersbook.com/
  - A Primer on Neural Network Models for Natural Language Processing. https://u.cs.biu.ac.il/~yogo/nnlp.pdf
  - On the Opportunities and Risks of Foundation Models https://arxiv.org/pdf/2108.07258.pdf
- We will be reading research papers from premier conferences in the field
  E.g., ACL, EMNLP, NAACL, ICLR, NeurIPS, ICML, …

Questions so far?

Course structure

- After first week, we will discuss 2 research papers in each session
- For each paper:
  - One student will present the paper with slides (15-20 minutes)
  - Other students will participate in class discussions with themselves and the presenter (~20 minutes)
- We will randomly assign presenter roles to students

Course structure

- **Before the class:** All students need to read the 2 papers
  - Students who are not presenting, need to prepare at least one question about each paper:
    - Could be anything you are confused about or something you'd like to hear discussed more, or an open-ended question
    - Submit your questions the night before the class
    - We will provide a form
  - We will use these questions partly as discussion points
  - You may come up with other questions in the class as the paper is being presented

Course structure

During the class

- Presentation – we will have 2 paper presentations each session (each 15-20 minutes)
  - The presenters may think of themselves as the lead author of the paper presenting it at a conference venue!
  - Discuss the main problem, contributions, methods, insights, and results
  - Discuss context or background necessary to understand the paper
  - Explain how the paper connects with papers discussed in previous sessions
  - Additionally prepare a few discussion points

Course structure

During the class

- **Discussion -** Other students who are not presenting have the role of reviewers and participate in discussions after the presentation

  - Think of yourselves as audience listening to the work or reviewers, and your goal is to prepare questions and evaluate the paper:

- What parts of the paper you found confusing? What are the main strengths? What are the weaknesses? How does the paper connect to other papers in the field?

Course structure

After the class

- Quiz: At the conclusion some of the class session (not all), a quiz may be distributed to assess understanding of the assigned paper and key discussion points covered during the session.
- These are due the day after the class

Guidelines for inclusive discussions

This is a **discussion-based course**, everyone should feel very welcome to participate in discussions and share their thoughts and opinions.

Example guidelines for promoting inclusive discussions:

- Be respectful and mindful of different opinions
- Try not to interrupt others, wait for them to finish
- Acknowledge that there are people with different expertise in the room
- Be positive, constructive, and polite

https://cse.ucsd.edu/sites/cse.ucsd.edu/files/Diversity/Inclusive_Seminar__LONG_.pdf

Final project

- Group projects (team size = 2 to 3 students)
  - 3 students are allowed for projects with a larger proposed scope

- What is the goal of the final project?
  - Conduct research on a specific NLP problem and submit a written report.

    Examples of possible projects
    - A novel investigation of existing methods to better understand their limitation or capabilities
    - Extending, training or fine-tuning an existing model for a new task, application, or domain
    - Exploratory projects on providing some insights about a specific modeling approach or a specific NLP problem/task

Class project and timeline

February 3
Form teams

February 25
Project proposal

April 1
Progress report

April 27
Presentation

May 10
Project report

Class project and timeline

- Project milestones:
  - **February 3:** Form teams (just send us an email and list your team members)

Class project and timeline

- Project milestones:
  - **February 3:** Form teams (just send us an email and list your team members)
  - **February 25:** project proposal (1-2 page)
    - Should describe what is the main goal of the project, the proposed research, and how it connects to existing work in the field
    - You will receive feedback in a week

Class project and timeline

- Project milestones:
  - **February 3:** Form teams (just send us an email and list your team members)
  - **February 25:** project proposal (1-2 page)
    - Should describe what is the main goal of the project, the proposed research, and how it connects to existing work in the field
    - You will receive feedback in a week
  - **April 1:** progress report (2 pages)
    - Describe the main problem, project goal and related work, what has been done so far, any initial results, and the plan continuing the project.
    - Receive feedback in a week

Class project and timeline

- Project milestones:
  - **February 3:** Form teams (just send us an email and list your team members)
  - **February 25:** project proposal (1-2 page)
    - Should describe what is the main goal of the project, the proposed research, and how it connects to existing work in the field
    - You will receive feedback in a week
  - **April 1:** progress report (2 pages)
    - Describe the main problem, project goal and related work, what has been done so far, any initial results, and the plan continuing the project.
    - Receive feedback in a week
  - **April 27:** project presentations
    - Projects will be presented in class

Class project and timeline

- Project milestones:
  - **February 3:** Form teams (just send us an email and list your team members)
  - **February 25:** project proposal (1-2 page)
    - Should describe what is the main goal of the project, the proposed research, and how it connects to existing work in the field
    - You will receive feedback in a week
  - **April 1:** progress report (2 pages)
    - Describe the main problem, project goal and related work, what has been done so far, any initial results, and the plan continuing the project.
    - Receive feedback in a week
  - **April 27:** project presentations
    - Projects will be presented in class
  - **May 10:** Final project report (6-8 pages)
    - The format of this report should be very similar to a conference paper
      - E.g., should include motivation, related work, proposed approach, results, and discussion

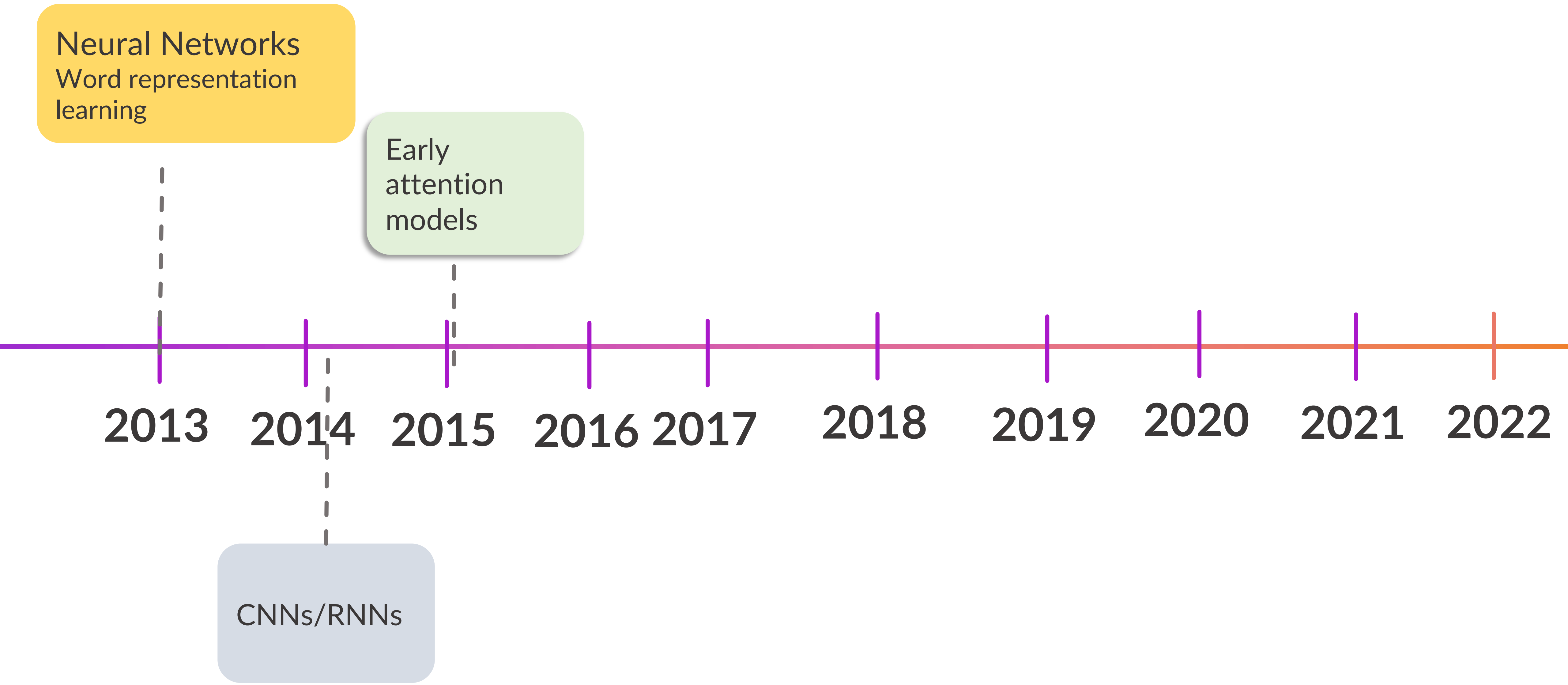
Grading

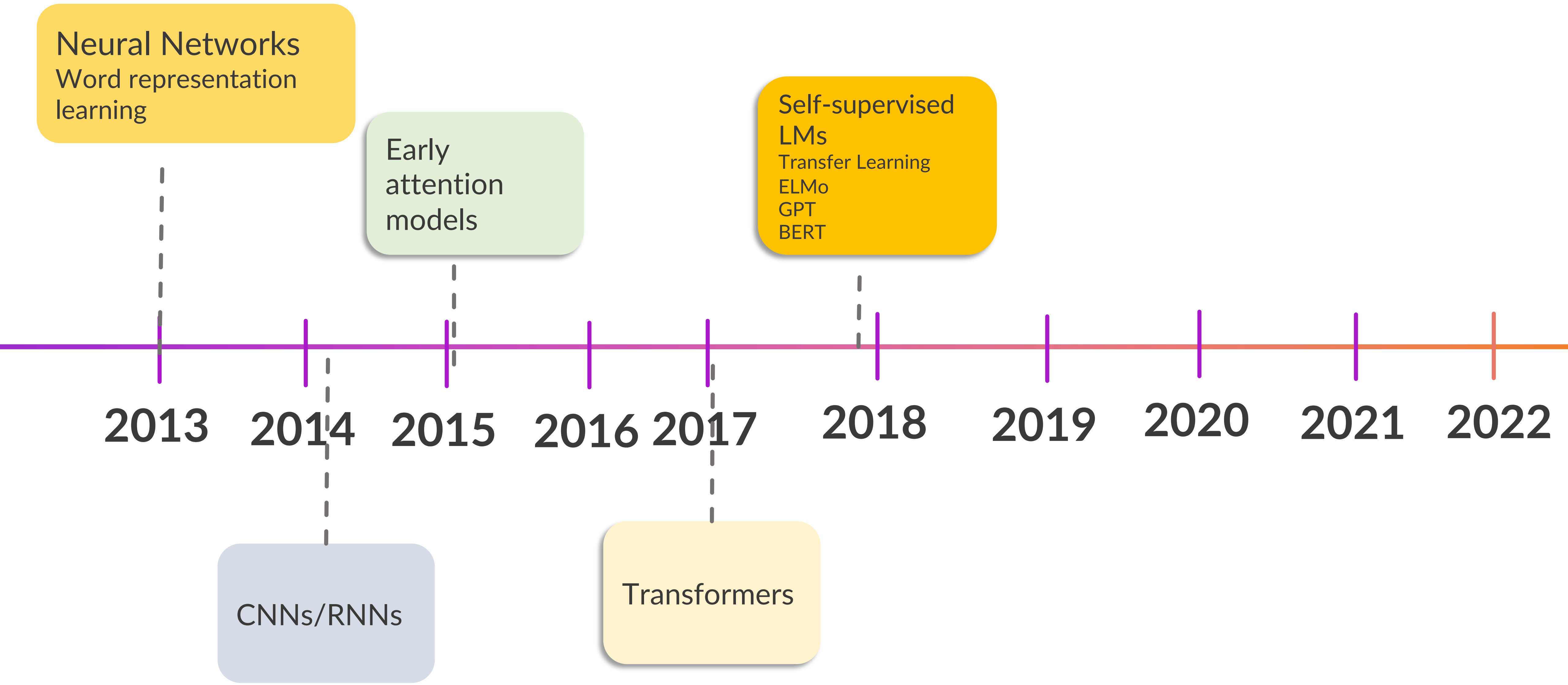
- Paper presentation and discussions (40%)
  - 20% Paper presentations
  - 10% Active participation in discussions
  - 10% question submissions and quiz
- Project (60%)
  - 10% Proposal
  - 10% Progress report
  - 10% Code
  - 10% Final presentation
  - 20% Final report
- If you’re engaged in class presentations/discussions and on top of your project, you should not be worried about the grade.

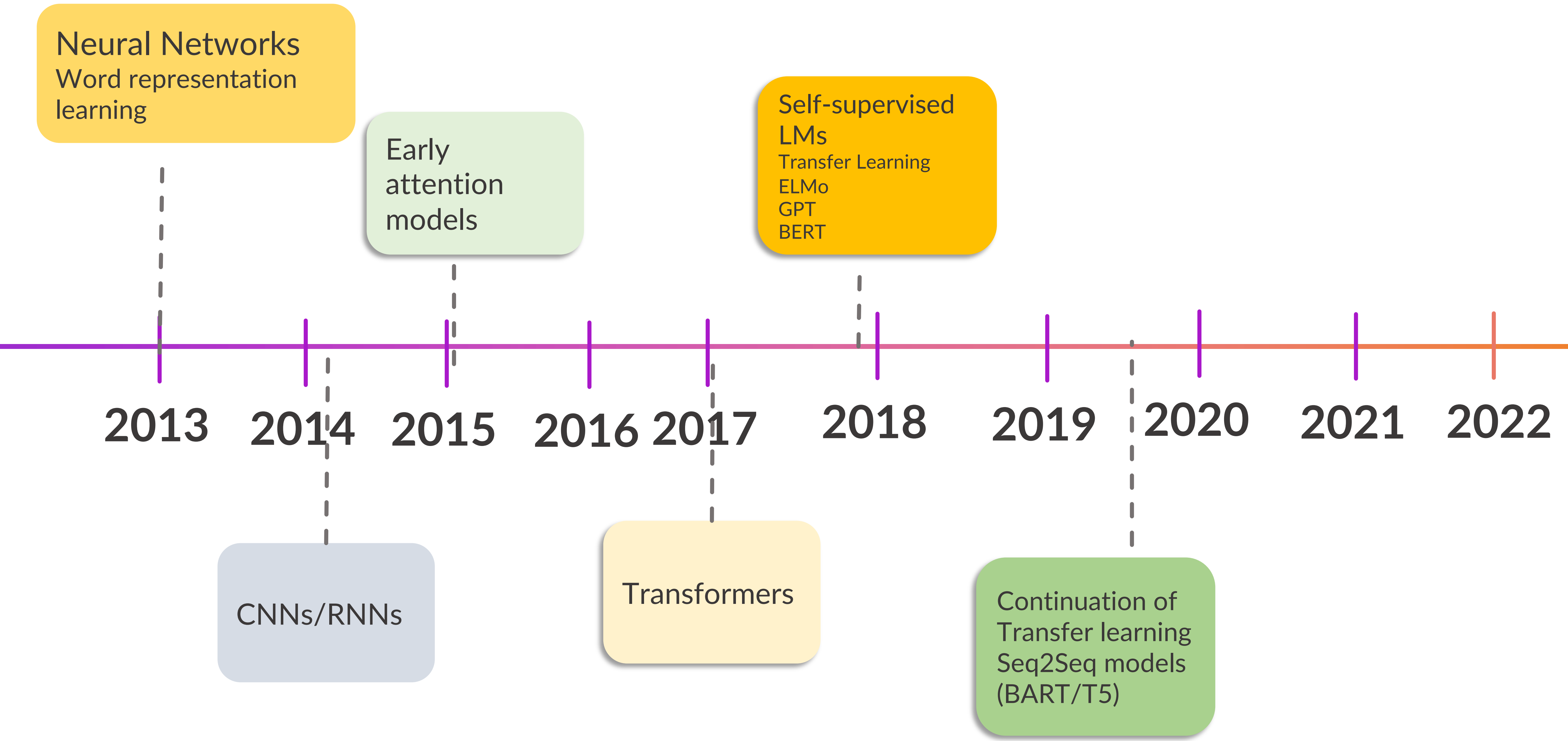
Questions?

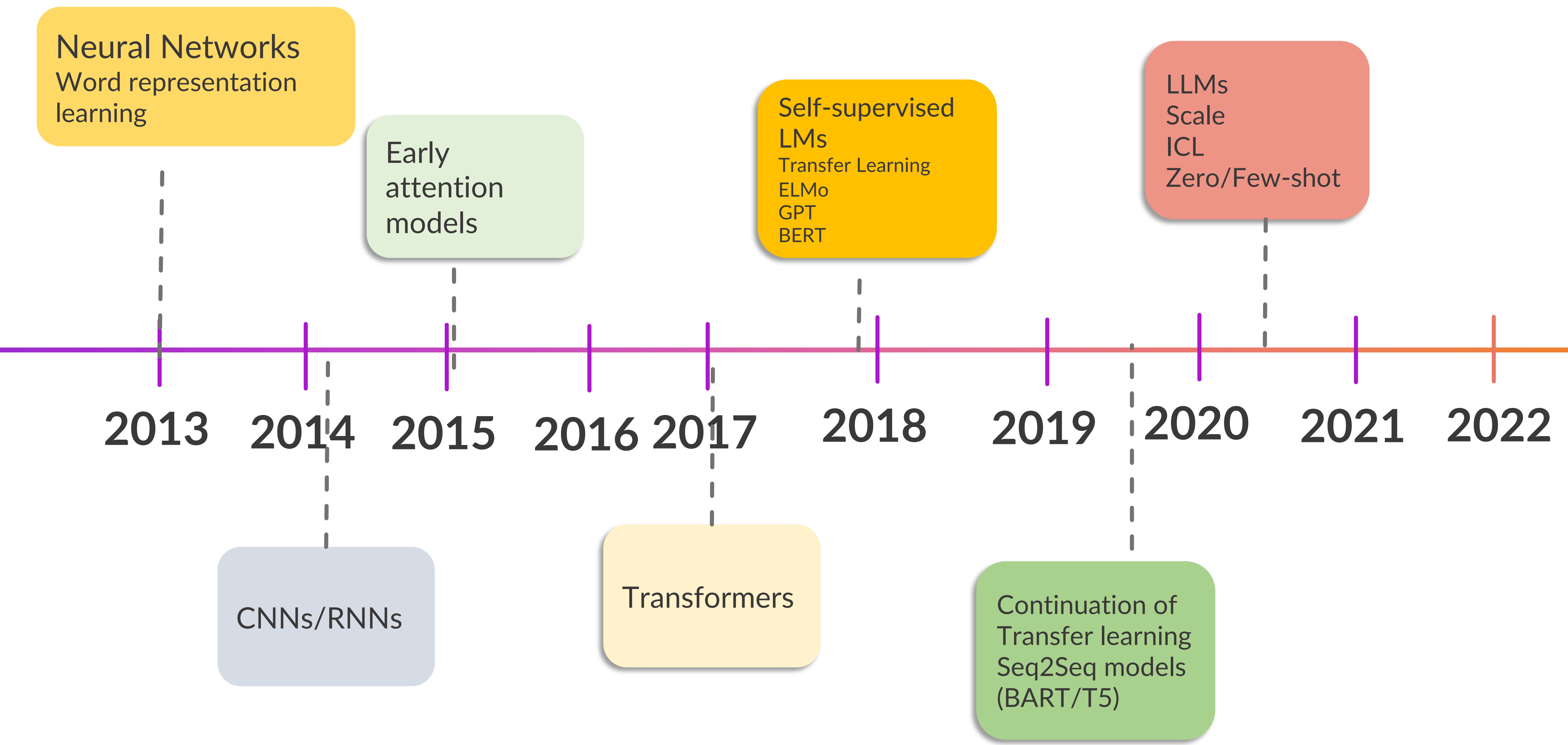
Current state of NLP

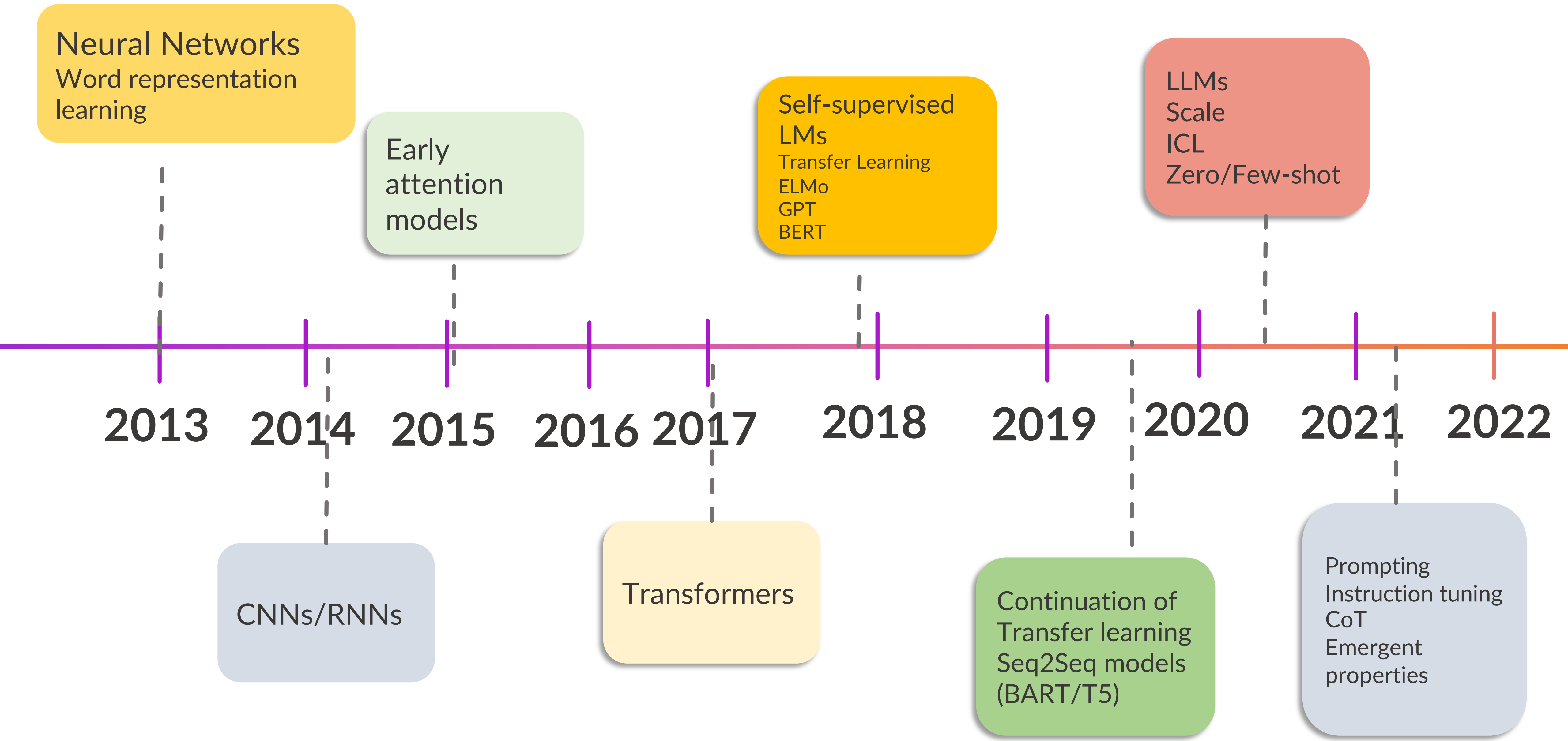
… and a brief history

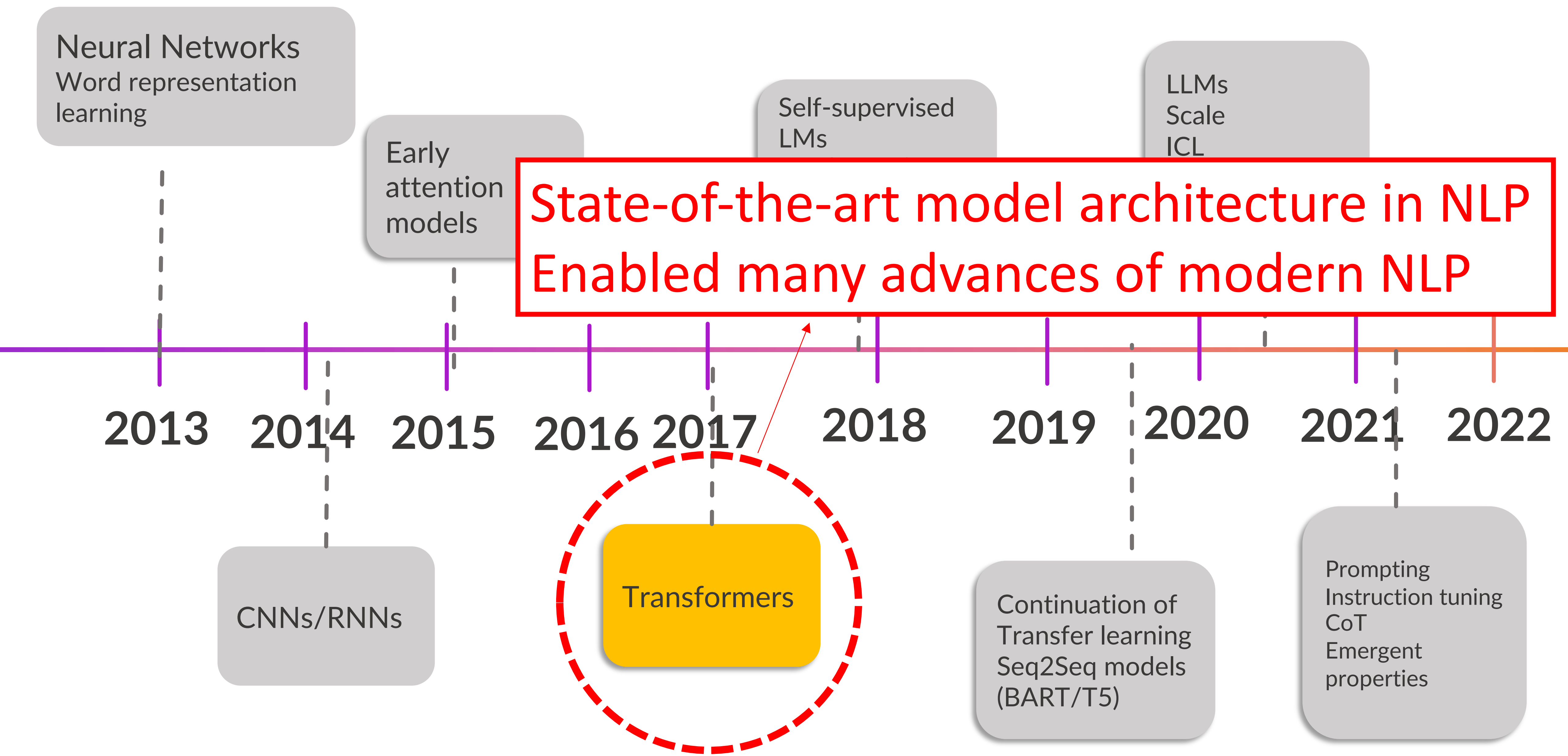

Neural Networks
Word representation learning
Early attention models
CNNs/RNNs
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022


Neural Networks
Word representation learning
Early attention models
Self-supervised LMs
Transfer Learning
ELMo
GPT
BERT
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
CNNs/RNNs
Transformers


Neural Networks
Word representation learning
Early attention models
Self-supervised LMs
Transfer Learning
ELMo
GPT
BERT
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
CNNs/RNNs
Transformers
Continuation of Transfer learning
Seq2Seq models
(BART/T5)


Neural Networks
Word representation learning
Early attention models
Self-supervised LMs
Transfer Learning
ELMo
GPT
BERT
LLMs
Scale
ICL
Zero/Few-shot
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
CNNs/RNNs
Transformers
Continuation of Transfer learning
Seq2Seq models
(BART/T5)


Neural Networks
Word representation learning
Early attention models
Self-supervised LMs
Transfer Learning
ELMo
GPT
BERT
LLMs
Scale
ICL
Zero/Few-shot
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
CNNs/RNNs
Transformers
Continuation of Transfer learning
Seq2Seq models
(BART/T5)
Prompting
Instruction tuning
CoT
Emergent properties

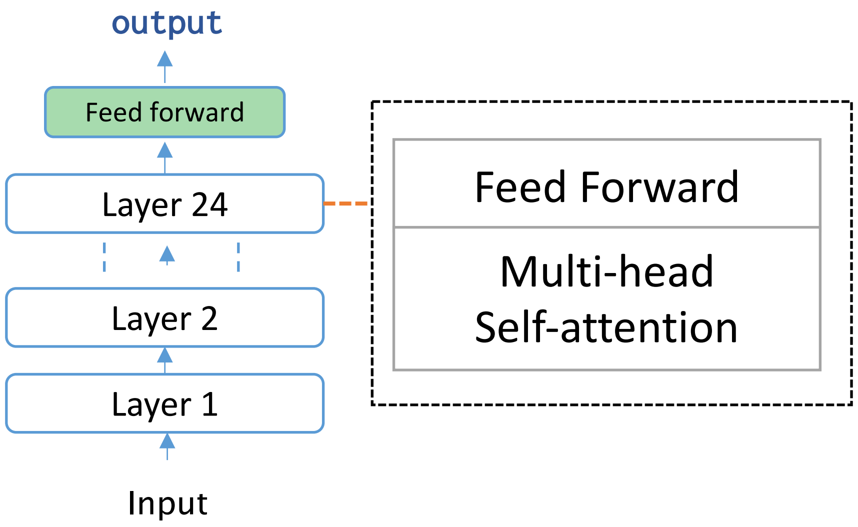

Neural Networks
Word representation learning
Early attention models
Self-supervised LMs
LLMs
Scale
ICL
State-of-the-art model architecture in NLP
Enabled many advances of modern NLP
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
CNNs/RNNs
Transformers
Continuation of Transfer learning
Seq2Seq models (BART/T5)
Prompting
Instruction tuning
CoT
Emergent properties

Transformer (Vaswani, et al 2017)

- A type of neural network designed primarily for sequence modeling
- Gradually build representations of the input sequence in each layer of the network
- A mechanism to let the model learn how to combine representations across the sequence (Attention)

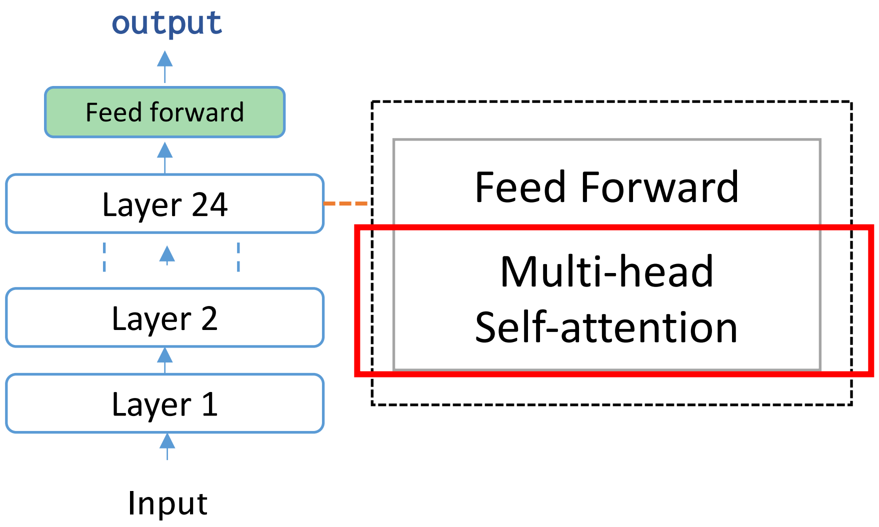
Transformer (Vaswani et al., 2017)

- Each layer consists of two major components
  - Self-attention and feed forward



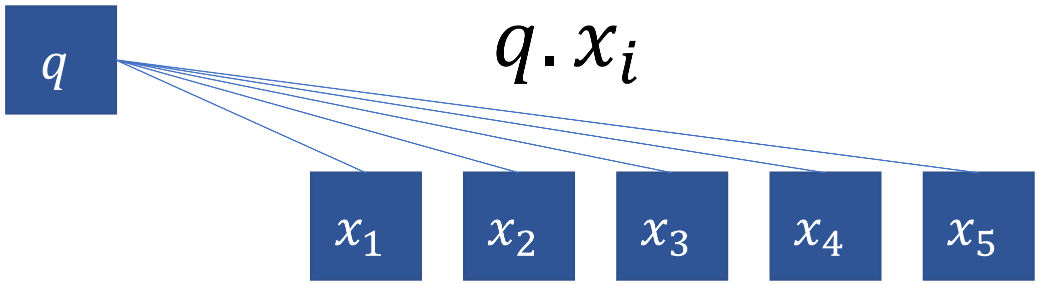
Transformer (Vaswani et al., 2017)

- Each layer consists of two major components
  - Self-attention and feed forward



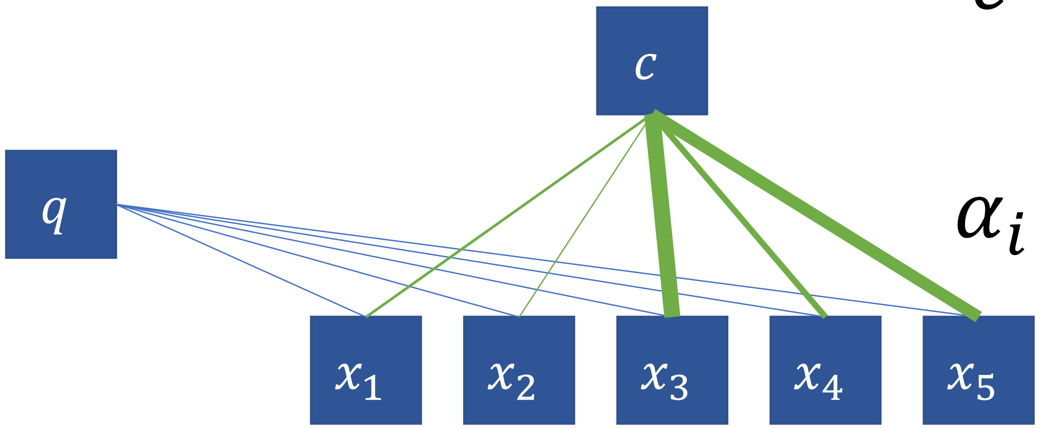
Attention mechanism (Bahdanau et al 2015)

- On high-level allows the model to weigh the importance of different parts of the input to build a contextual representation with respect to a query



Attention mechanism (Bahdanau et al 2015)

- On high-level allows the model to weigh the importance of different parts of the input to build a contextual representation with respect to a query



$$c = \sum_i \alpha_i x_i$$

$$\alpha_i = \mathrm{softmax}(q . x_i)$$

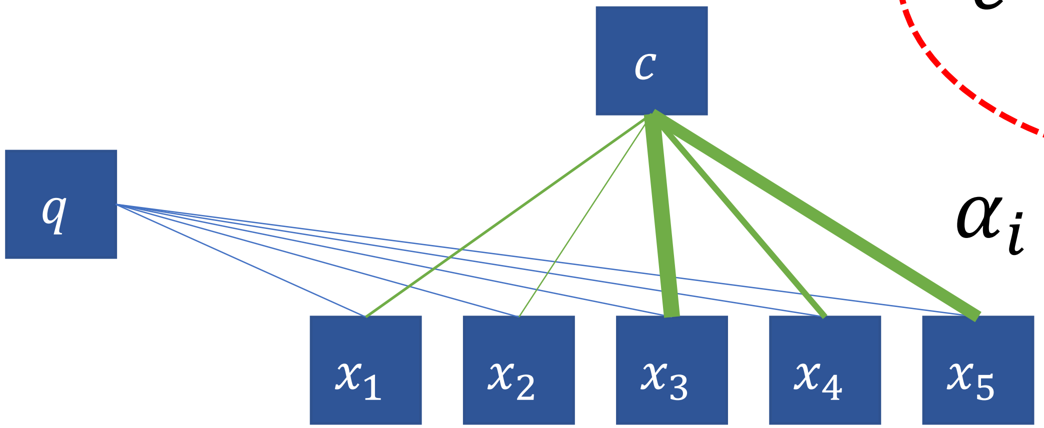
$$\mathrm{softmax}(z_i) = \frac{e_i^z}{\sum_j e^j}$$

Attention mechanism (Bahdanau et al 2015)

- On high-level allows the model to weigh the importance of different parts of the input to build a **contextual representation** with respect to a **query**

contextual representation of the sequence w.r.t the query

$$c = \sum_i \alpha_i x_i$$

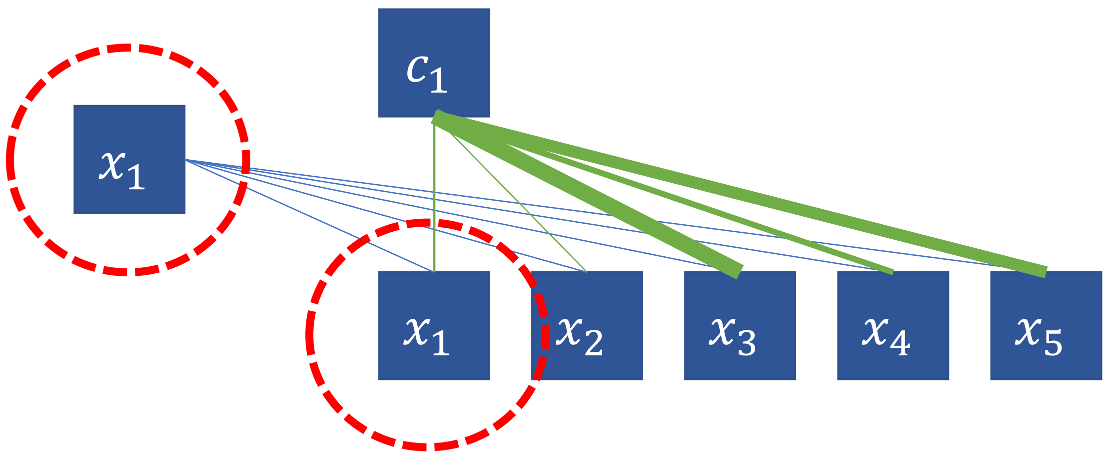


$$\alpha_i = \mathrm{softmax}(q . x_i)$$

$$\mathrm{softmax}(z_i) = \frac{e_i^z}{\sum_j e^j}$$

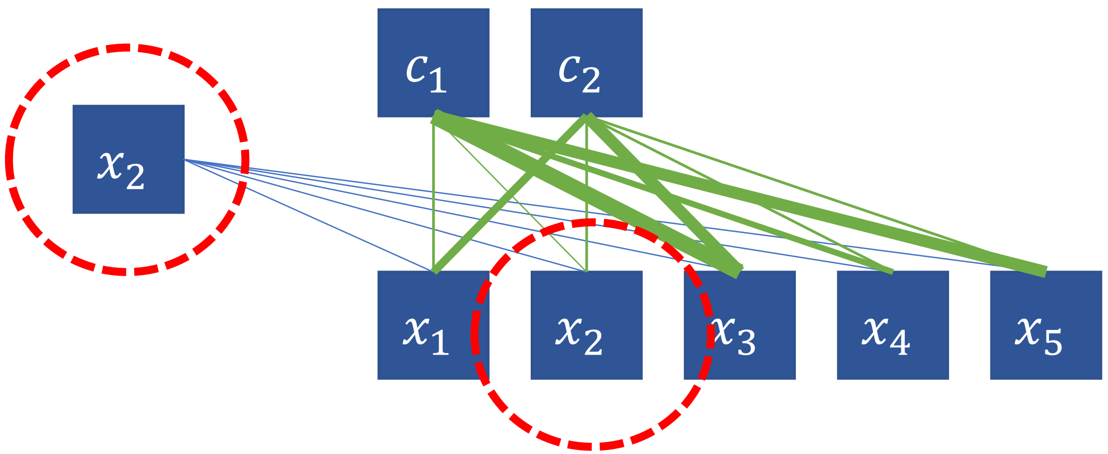
Transformer and self-attention

- Self-attention
- The **query** is the same sequence
  - **For each element x_i in the sequence we compute the contextual representation of the sequence w.r.t. x_i**



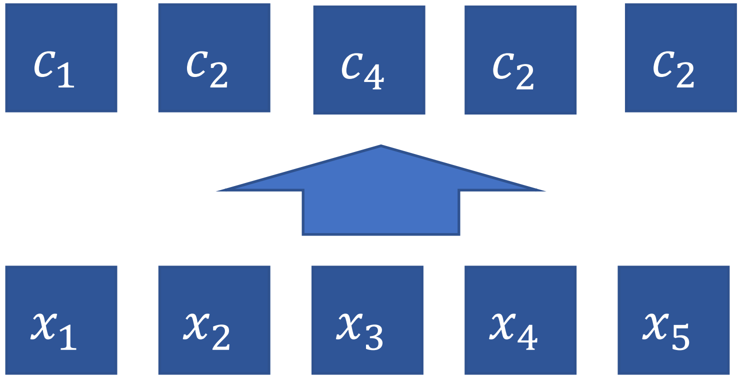
Transformer and self-attention

- Self-attention
- The **query** is the same sequence
  - **For each element x_i in the sequence we compute the contextual representation of the sequence w.r.t. x_i**



Transformer and self-attention

- Self-attention
- The **query** is the same sequence
  - **For each element x_i in the sequence we compute the contextual representation of the sequence w.r.t. x_i**



$$c_i = \sum_j \alpha_{ij} x_j$$

$$\alpha_{ij} = \mathrm{softmax}_j(x_i . x_j)$$

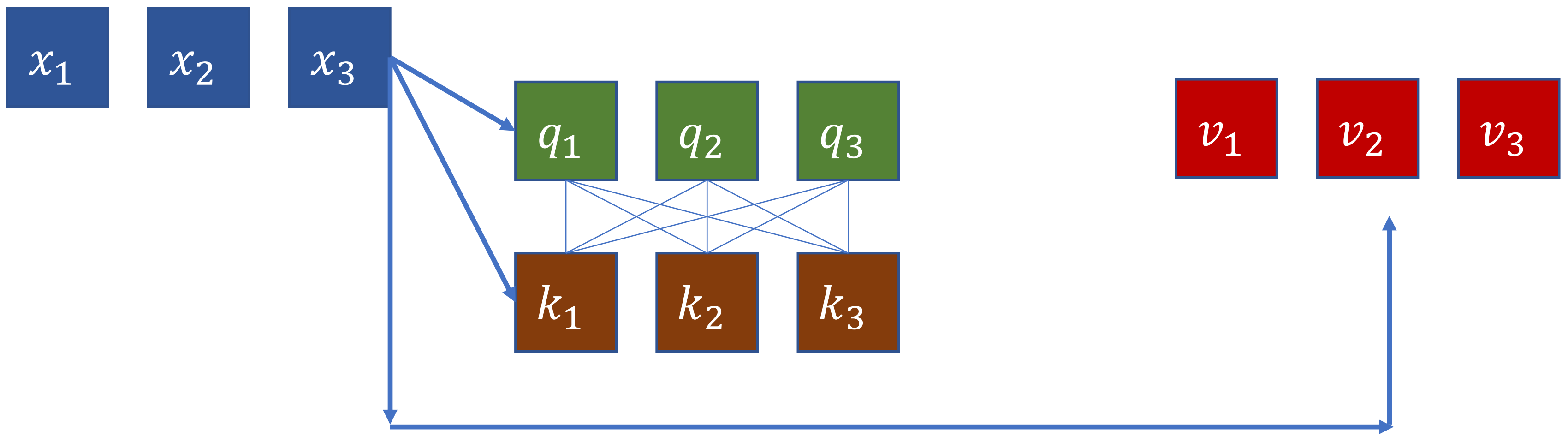
attention scores

Transformer

- Instead of using one vector x_i for each input location
  - Each input vector is projected into three vectors
    - Query vector $q_i = W_q x_i$
    - Key vector $k_i = W_k x_i$
    - Value vector $v_i = W_v x_i$

Transformer

- Instead of using one vector x_i for each input location
  - Each input vector is projected into three vectors
    - Query vector $q_i = W_q x_i$
    - Key vector $k_i = W_k x_i$
    - Value vector $v_i = W_v x_i$

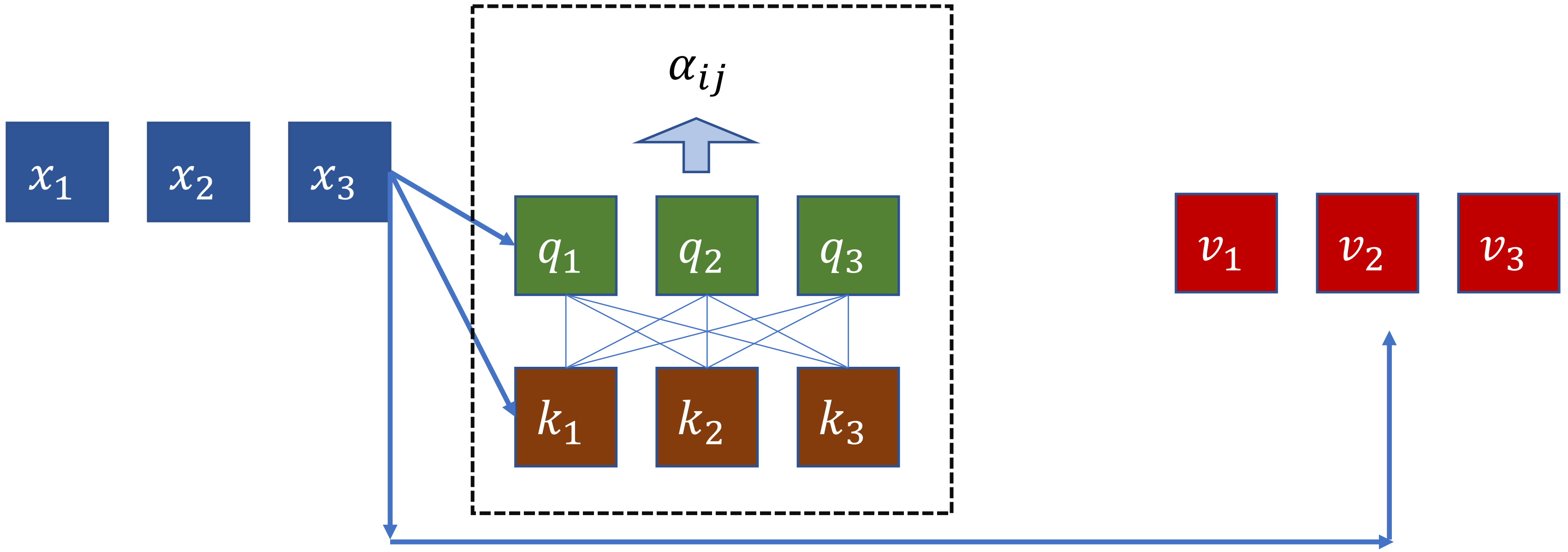


Transformer

- Instead of using one vector x_i for each input location
  - Each input vector is projected into three vectors
    - Query vector $q_i = W_q x_i$
    - Key vector $k_i = W_k x_i$
    - Value vector $v_i = W_v x_i$

$$\alpha_{ij} = \text{softmax}_j(q_i . k_j)$$

attention scores

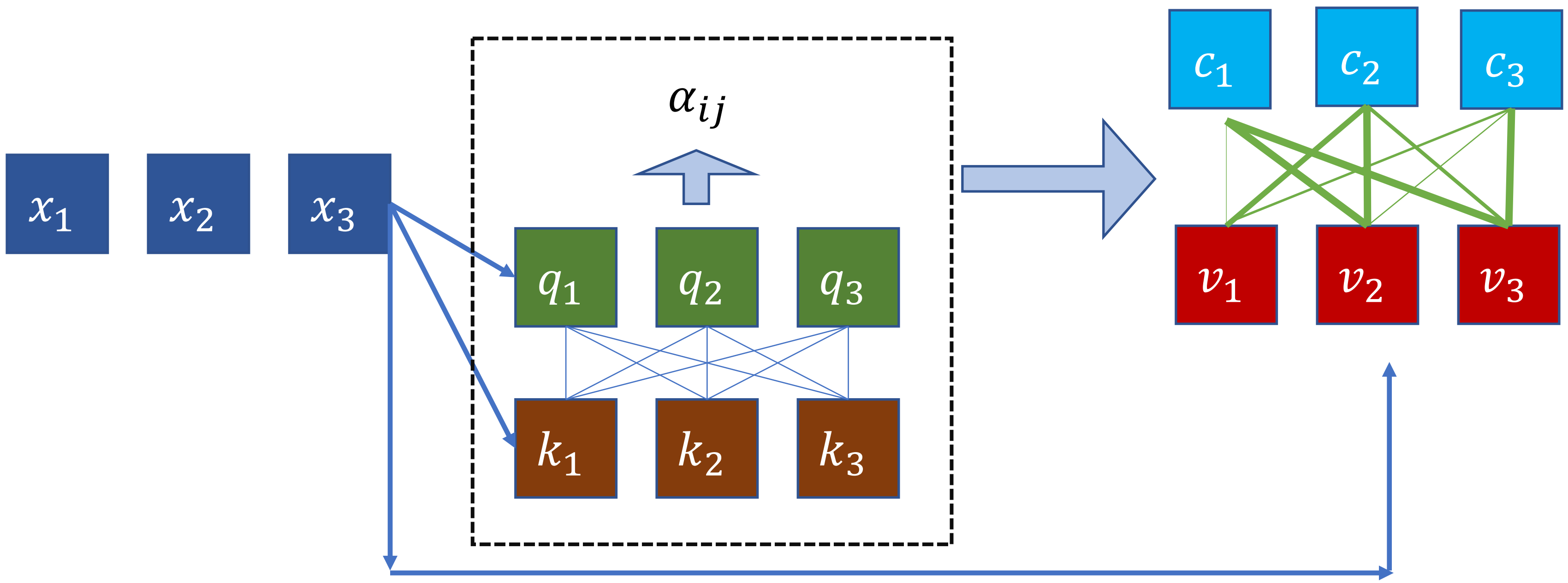


Transformer

- Instead of using one vector x_i for each input location
  - Each input vector is projected into three vectors
    - Query vector $q_i = W_q x_i$
    - Key vector $k_i = W_k x_i$
    - Value vector $v_i = W_v x_i$

$$\alpha_{ij} = \mathrm{softmax}_j(q_i . k_j)$$

$$c_i = \sum_j \alpha_{ij} v_j$$



Transformer

- Vaswani et al., 2017 used a slightly modified version of attention
- They re-scale the logits by a factor of $\frac{1}{\sqrt{d_k}}$ where d_k is the dimension of the key vector

$$\alpha_{ij} = \mathrm{softmax}_j(q_i . k_j) \longrightarrow \alpha_{ij} = \mathrm{softmax}_j\left(\frac{q_i . k_j}{\sqrt{d_k}}\right)$$

Putting it all together: Implementation

```python
def self_attention(query, key, value):
    d_k = query.size(-1)
    attn_logits = torch.matmul(query, key.transpose(-2, -1)) / math.sqrt(d_k)
    attn_scores = F.softmax(attn_logits, dim=-1)
    context_vectors = torch.matmul(attn_scores, value)
    return context_vectors, attn_scores
```

Multi-head attention

- Vaswani et al., (2017) use multiple attention heads instead of one
  - Perform self_attention times: N number of heads
    - Query vector $q_i^n = W_q^n x_i$ (W_q^n: Query projection weights for head $1 \leq n \leq N$)
    - Key vector $k_i^n = W_k^n x_i$
    - Value vector $v_i^n = W_v^n x_i$

  - Concatenate the resulting context vectors and project the output

$$c_i^n = \sum_j \mathrm{softmax}\left(\frac{q_i^n k_j^n}{\sqrt{d_k}}\right) v_j^n$$

$$c_i = \mathrm{concat}\left(c_i^1, \ldots, c_i^N\right). W^o$$

Multi-head attention

- Example dimensions (used in original Transformer)
  - $d = 512$ (input vector dimension)
  - $h = 8$ (number of attention heads)
  - $d_v = d_k = d_q = \frac{512}{8} = 64$

$$c_i^n = \sum_j \operatorname{softmax}\left(\frac{q_i^n k_j^n}{\sqrt{d_k}}\right) v_j^n$$

$$c_i = \operatorname{concat}\left(c_i^1, \ldots, c_i^N\right). W^o \qquad W^o \in \mathbb{R}^{h.d_v \times d}$$

  - What is the output vector dimension c_i?

Multi-head attention

- Example dimensions (used in original Transformer)
  - $d = 512$ (input vector dimension)
  - $h = 8$ (number of attention heads)
  - $d_v = d_k = d_q = \frac{512}{8} = 64$

$$c_i^n = \sum_j \mathrm{softmax}\left(\frac{q_i^n k_j^n}{\sqrt{d_k}}\right) v_j^n$$

$$c_i = \mathrm{concat}\left(c_i^1, \ldots, c_i^N\right). W^o \qquad W^o \in \mathbb{R}^{h.d_v \times d}$$

- What is the output vector dimension c_i? Output dimension is *d=512*

Multi-head attention

```python
class TransformeMultiHeadAttention:
    def __init__(self, d_model, n_heads, dropout=0.1):
        self.d_model = d_model
        self.n_heads = n_heads
        self.d_k = d_model // n_heads
        self.w_q = nn.Linear(d_model, d_model)
        self.w_k = nn.Linear(d_model, d_model)
        self.w_v = nn.Linear(d_model, d_model)
        self.w_o = nn.Linear(d_model, d_model)
        self.dropout = nn.Dropout(dropout)

    def forward(self, q, k, v):
        batch_size = q.size(0)
        q = self.w_q(q).view(batch_size, -1, self.n_heads, self.d_k).transpose(1, 2)
        k = self.w_k(k).view(batch_size, -1, self.n_heads, self.d_k).transpose(1, 2)
        v = self.w_v(v).view(batch_size, -1, self.n_heads, self.d_k).transpose(1, 2)
        context_vectors, attn_scores = self_attention(q, k, v)
        context_vectors = context_vectors.transpose(1, 2).contiguous().view(batch_size, -1, self.d_model)
        return self.w_o(context_vectors), attn_scores
```

Multi-head attention

```
class TransformeMultiHeadAttention:
    def __init__(self, d_model, n_heads, dropout=0.1):
        self.d_model = d_model
        self.n_heads = n_heads
        self.d_k = d_model // n_heads
        self.w_q = nn.Linear(d_model, d_model)
        self.w_k = nn.Linear(d_model, d_model)
        self.w_v = nn.Linear(d_model, d_model)
        self.w_o = nn.Linear(d_model, d_model)
        self.dropout = nn.Dropout(dropout)

    def forward(self, q, k, v):
        batch_size = q.size(0)
        q = self.w_q(q).view(batch_size, -1, self.n_heads, self.d_k).transpose(1, 2)
        k = self.w_k(k).view(batch_size, -1, self.n_heads, self.d_k).transpose(1, 2)
        v = self.w_v(v).view(batch_size, -1, self.n_heads, self.d_k).transpose(1, 2)
        context_vectors, attn_scores = self_attention(q, k, v)
        context_vectors = context_vectors.transpose(1, 2).contiguous().view(batch_size, -1, self.d_model)
        return self.w_o(context_vectors), attn_scores
```

q,k,v projections

output projection

Multi-head attention

```python
class TransformeMultiHeadAttention:
    def __init__(self, d_model, n_heads, dropout=0.1):
        self.d_model = d_model
        self.n_heads = n_heads
        self.d_k = d_model // n_heads
        self.w_q = nn.Linear(d_model, d_model)
        self.w_k = nn.Linear(d_model, d_model)
        self.w_v = nn.Linear(d_model, d_model)
        self.w_o = nn.Linear(d_model, d_model)
        self.dropout = nn.Dropout(dropout)

    def forward(self, q, k, v):
        batch_size = q.size(0)
        q = self.w_q(q).view(batch_size, -1, self.n_heads, self.d_k).transpose(1, 2)
        k = self.w_k(k).view(batch_size, -1, self.n_heads, self.d_k).transpose(1, 2)
        v = self.w_v(v).view(batch_size, -1, self.n_heads, self.d_k).transpose(1, 2)
        context_vectors, attn_scores = self_attention(q, k, v)
        context_vectors = context_vectors.transpose(1, 2).contiguous().view(batch_size, -1, self.d_model)
        return self.w_o(context_vectors), attn_scores
```

q,k,v projections

output projection

Main self-attention function

Final projection

Feed forward layer

- A position-wise transformation consisting of:
  - A linear transformation, non-linear activation f (e.g., ReLU), and another linear transformation.

$$FF(c) = f(cW_1 + b_1)W_2 + b_2$$

Feed forward layer

- A position-wise transformation consisting of:
  - A linear transformation, non-linear activation f (e.g., ReLU), and another linear transformation.

$$FF(c) = f(cW_1 + b_1)W_2 + b_2$$

  - This allows the model to apply another transformation to the contextual representations (or "post-process" them)
  - Usually the dimensionality of the hidden feedforward layer is 2-8 times larger than the input dimension

Feed forward layer

```
class TransformerFeedForward:

    def __init__(self, d_model, d_ff, dropout=0.1):
        self.w_1 = nn.Linear(d_model, d_ff)
        self.w_2 = nn.Linear(d_ff, d_model)
        self.dropout = nn.Dropout(dropout)

    def forward(self, x):
        x = self.w_2(F.relu(self.w_1(x)))
        return self.dropout(x)
```

How to encode position information?

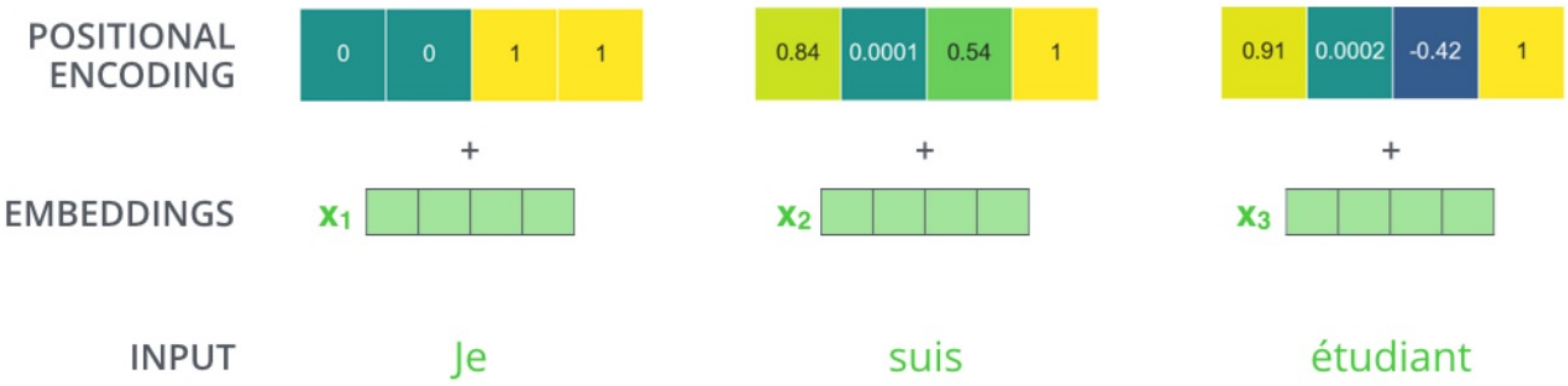
- Multi-head attention doesn't have a way to know whether an input token comes before or after another
  - Position is important in sequence modeling in NLP

- One way to introduce position information is add individual position encodings to the input for each position in the sequence

$$x_j = x_j + pos_j$$

Where pos_j is a position vector

How to encode position information?



The positional encodings can be a functional form (sinusoidal encoding) or can be parameters that we learn

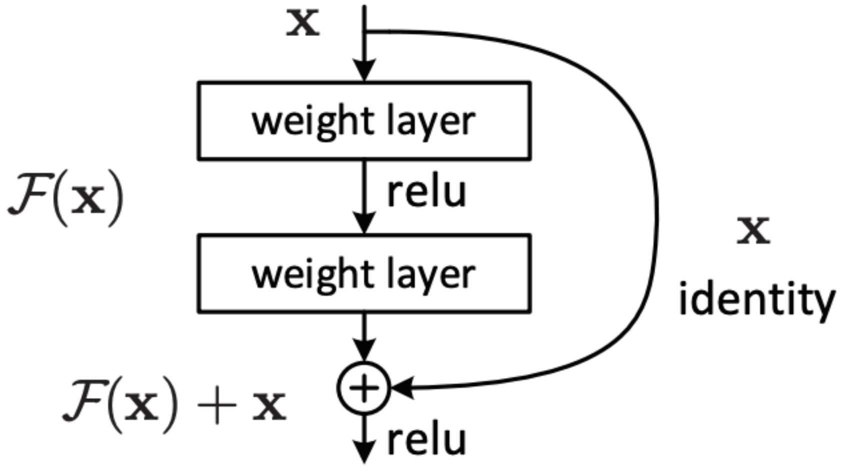
Image from https://jalammar.github.io/illustrated-transformer/

Residual connection and layer norm

- Recall residual connection

Residual connection and layer norm

- Recall residual connection



Makes training more stable by allowing gradients to flow more easily through multiple layers

https://arxiv.org/pdf/1512.03385v1.pdf

Residual connection and layer norm

- Recall layer normalization

Residual connection and layer norm

- Recall layer normalization
  - Normalize the activations of a neural network layer.
  - Helps to stabilize the training of deep networks

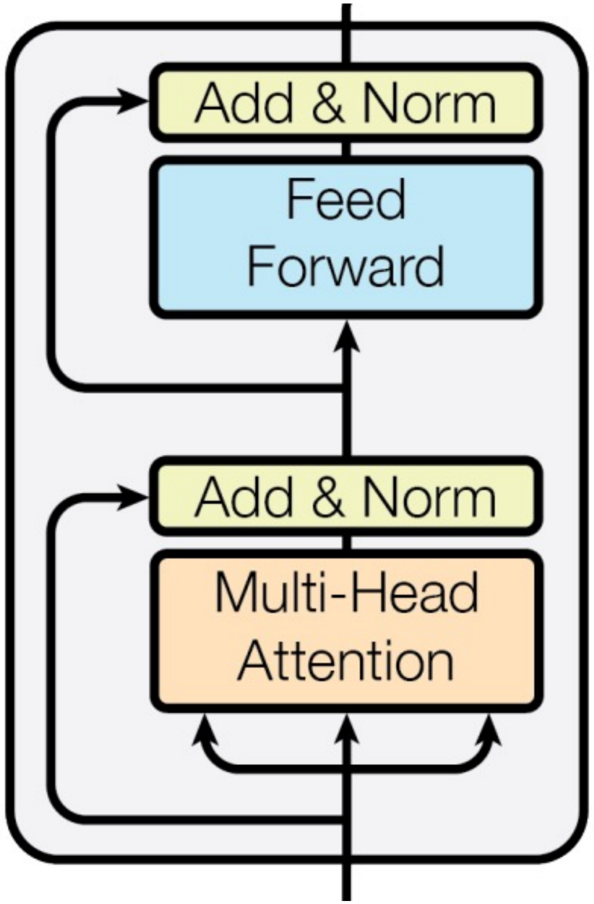
$$\hat{x}_{i,k} = \frac{x_{i,k} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}$$

  - Subtract the mean of features μ_i from input and divide by standard deviation
  - Scale and shift by learnable parameters β and γ
  - $$y_i = \gamma \hat{x}_i + \beta \equiv \mathrm{LN}_{\gamma,\beta}(x_i)$$

https://arxiv.org/pdf/1512.03385v1.pdf

A transformer block

out



x: input sequence

$$\text{out} = \text{LN}(c' + \text{FF}(c'))$$

$$FF(c') = f(c'W_1 + b_1)W_2 + b_2$$

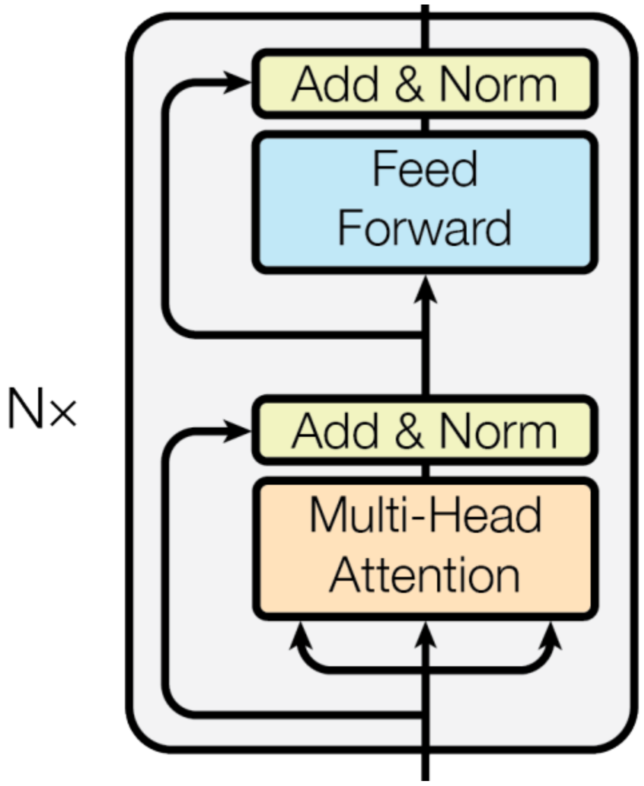
$$c' = LN(c + x)$$

$$c = \text{MultiHeadAttention}(\text{q}, \text{k}, \text{v})$$

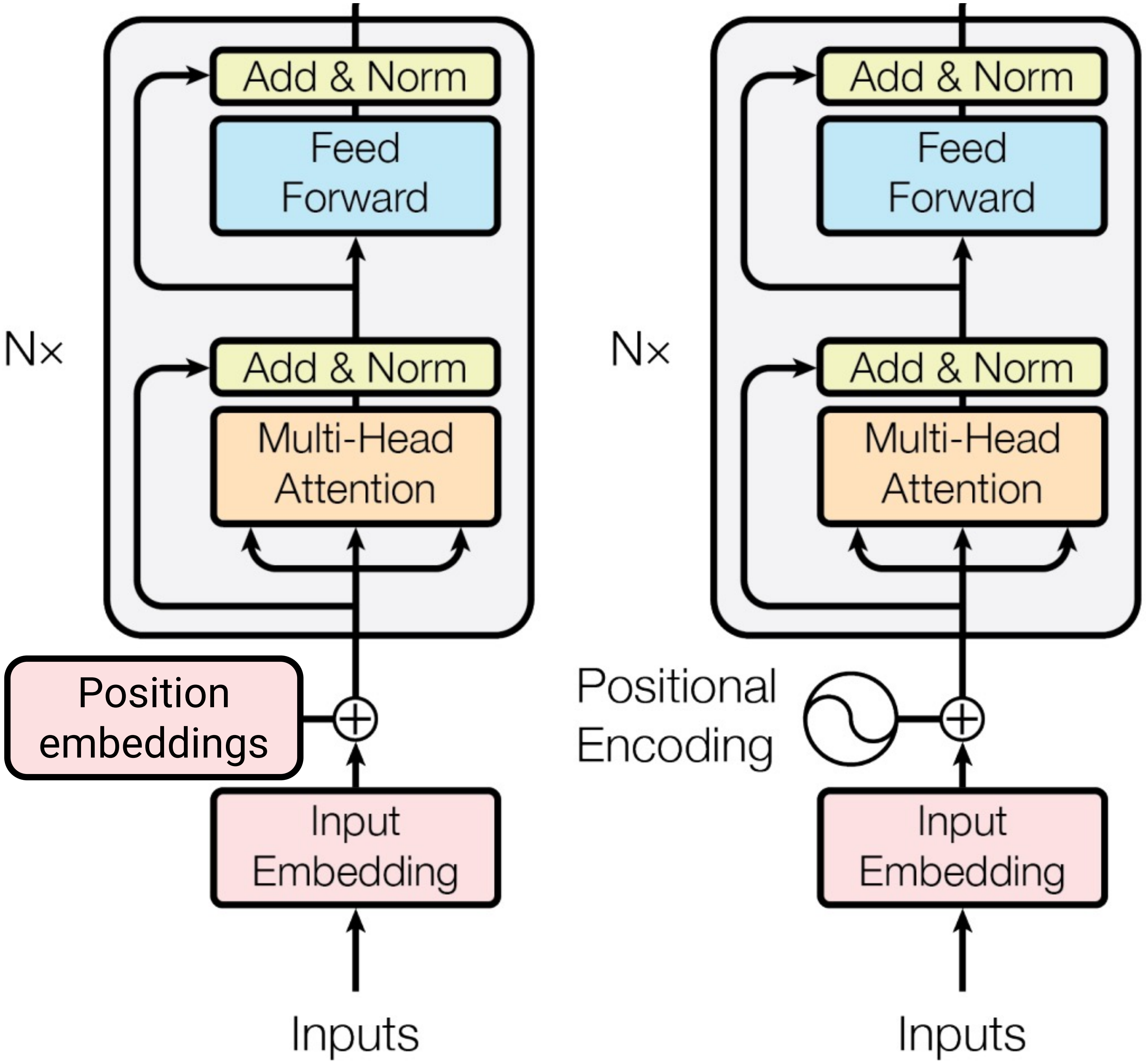
$$q, k, v = \text{QKV\_Projection}(x)$$

Transformer stack

- A stack of N transformer blocks (organized in N layers)



Position info



Encoder-decoder

- It is possible to have two stacks of transformer layers
- The encoder is as we've seen
- We can also add a decoder layer that is identical to the encoder but we give it the ability to also attend to the input

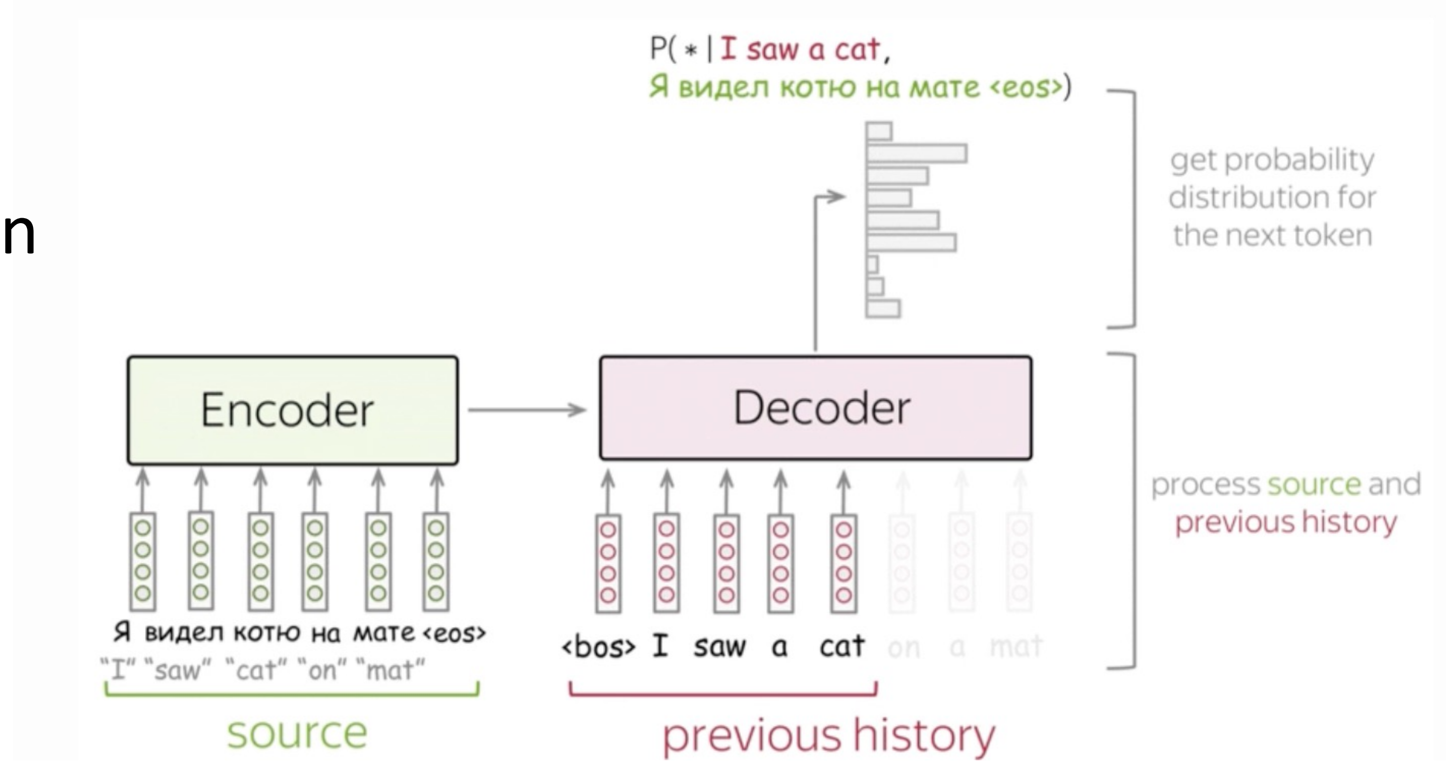
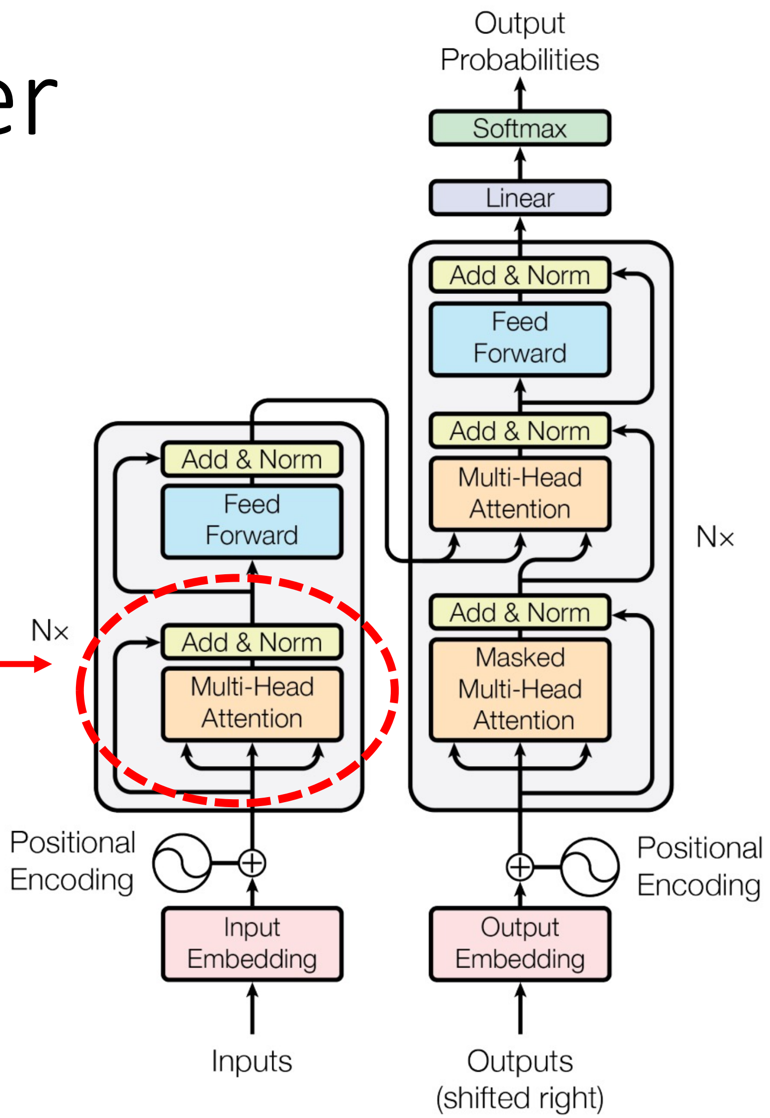


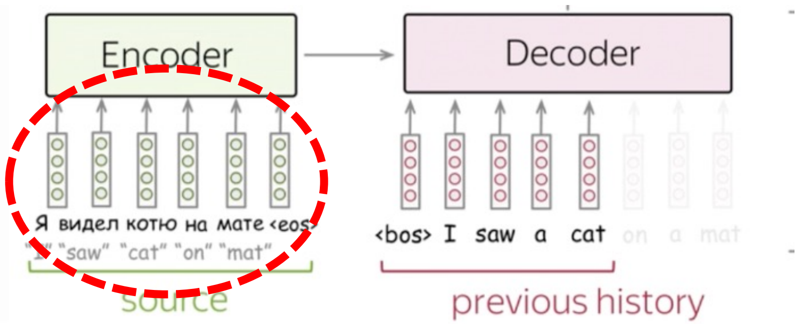
Fig from: https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Encoder-decoder

Example:
Machine translation

Self-attention in encoder





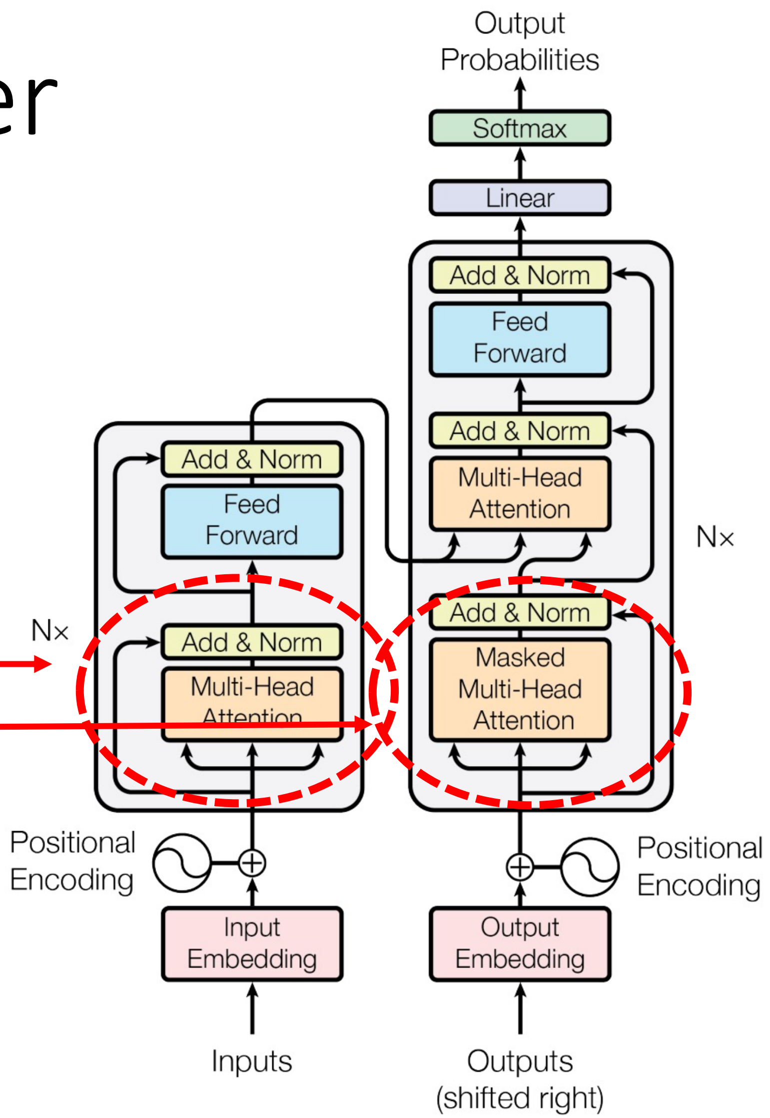
En: The cat is sleeping. → **Fr:** Le chat dort.

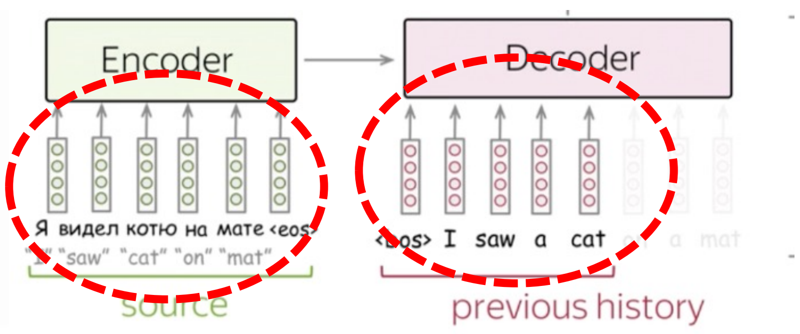
Encoder-decoder

Example:
Machine translation

Self-attention in encoder

Self-attention in decoder





En: The cat is sleeping. → **Fr:** Le chat dort.

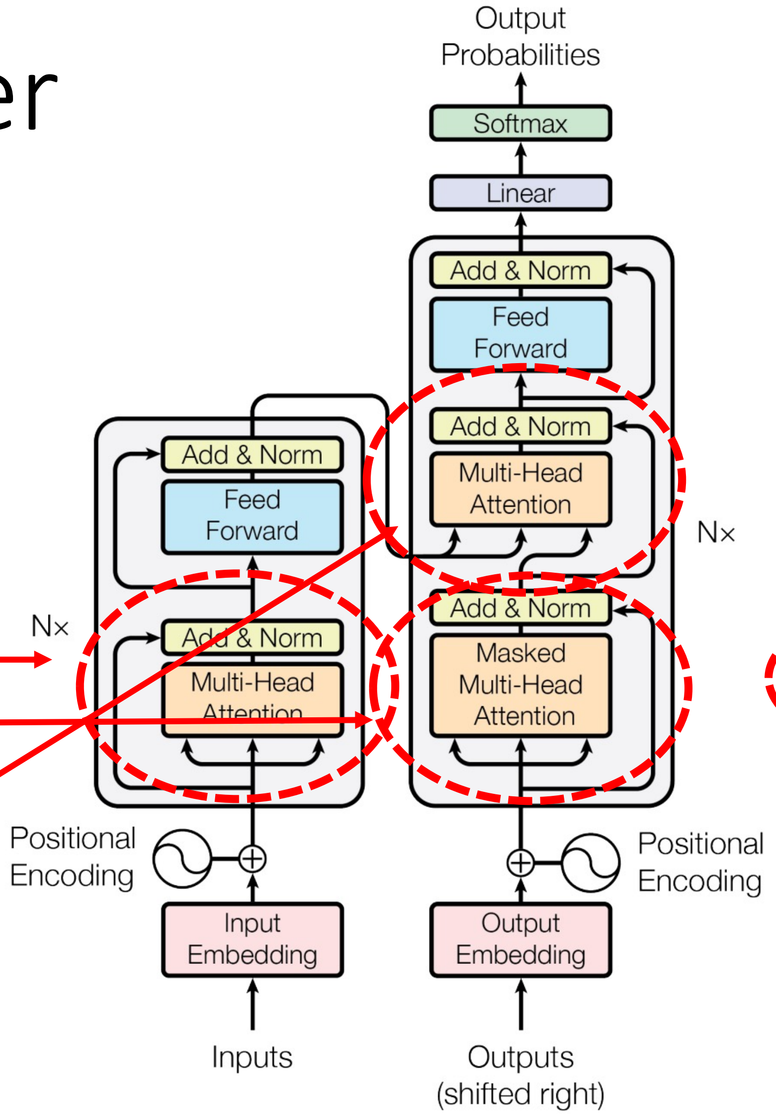
Encoder-decoder

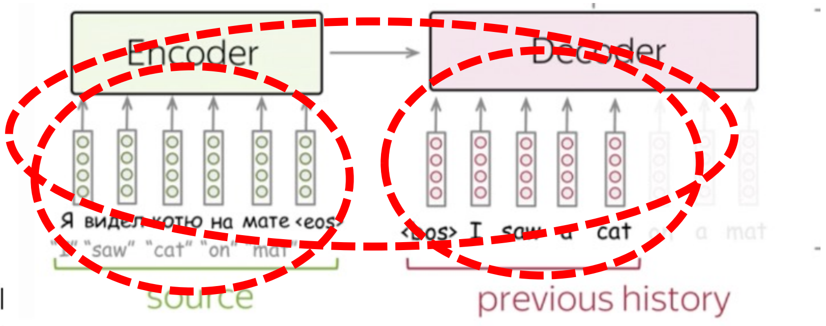
Example:
Machine translation

Self-attention in encoder

Self-attention in decoder

Cross attention





En: The cat is sleeping. → **Fr:** Le chat dort.

How did transformers change NLP?

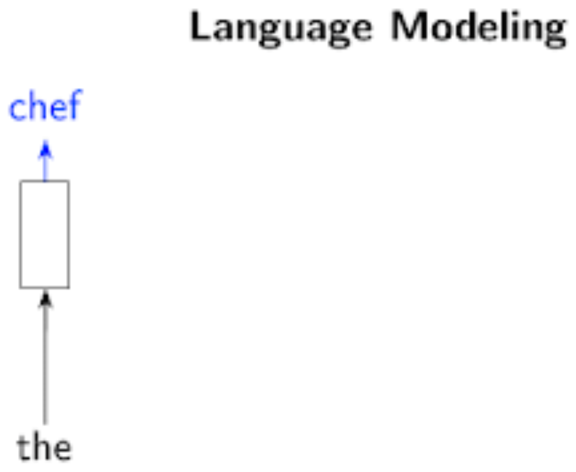
How did transformers change NLP?

Transformers + Large-scale self-supervised learning or language modeling

Self-supervised learning

- Using unlabeled data to train the model

Predict the next word



Predict the masked word

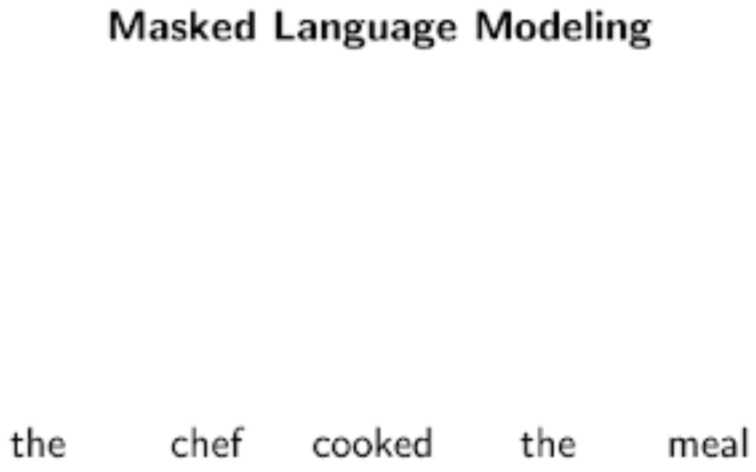
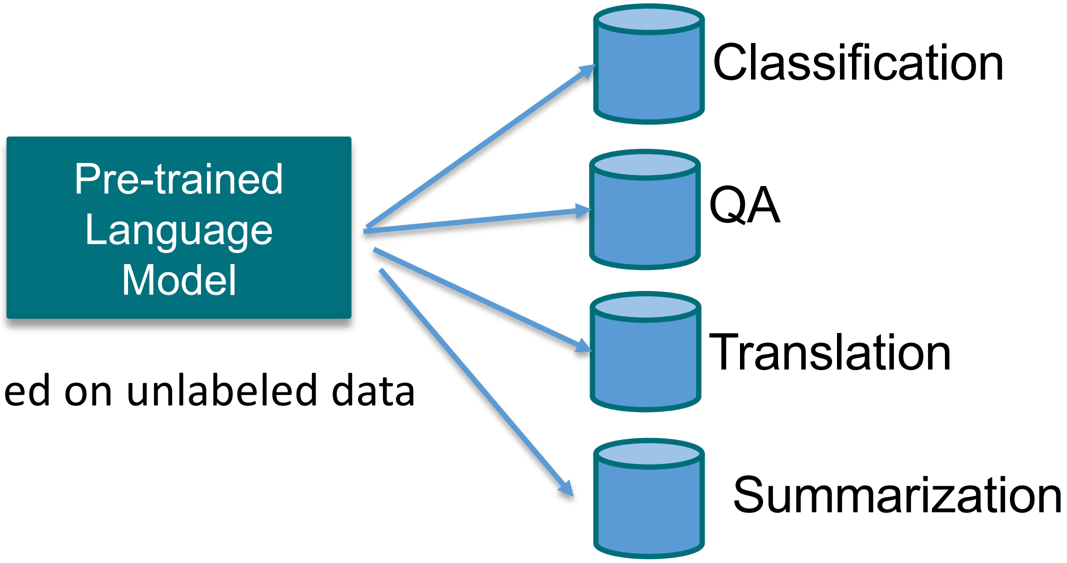


Figure from: https://ai.googleblog.com/2020/03/more-efficient-nlp-model-pre-training.html

Self-supervised learning - Transfer learning

- Pre-train then fine-tune paradigm
- After training on unlabeled data, the model works much better on labeled data → Transfer learning



Trained on unlabeled data

Transfer to downstream tasks

Scale

- 2 main dimensions:
- Model size, pretraining data size

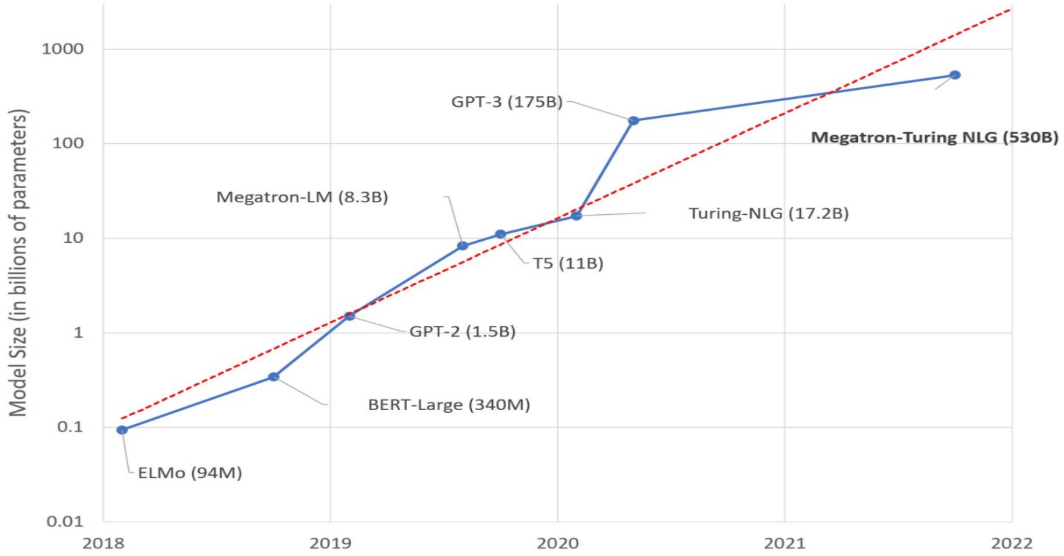
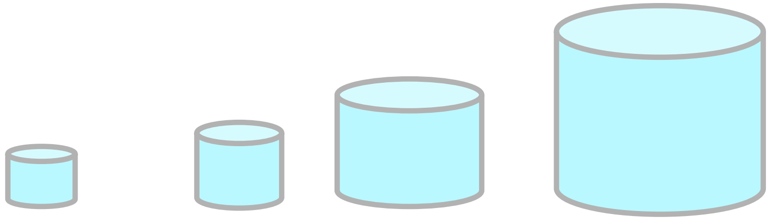


Photo credit: https://www.microsoft.com/en-us/research/blog/using-deepspeed-and-megatron-to-train-megatron-turing-nlg-530b-the-worlds-largest-and-most-powerful-generative-language-model/



Wiki+books (BERT, 2018)

~3B tokens

Wiki+books+news (RoBERTa 2019)

~30B tokens

C4 (T5, 2020)

360B tokens

Chinchilla (2022)

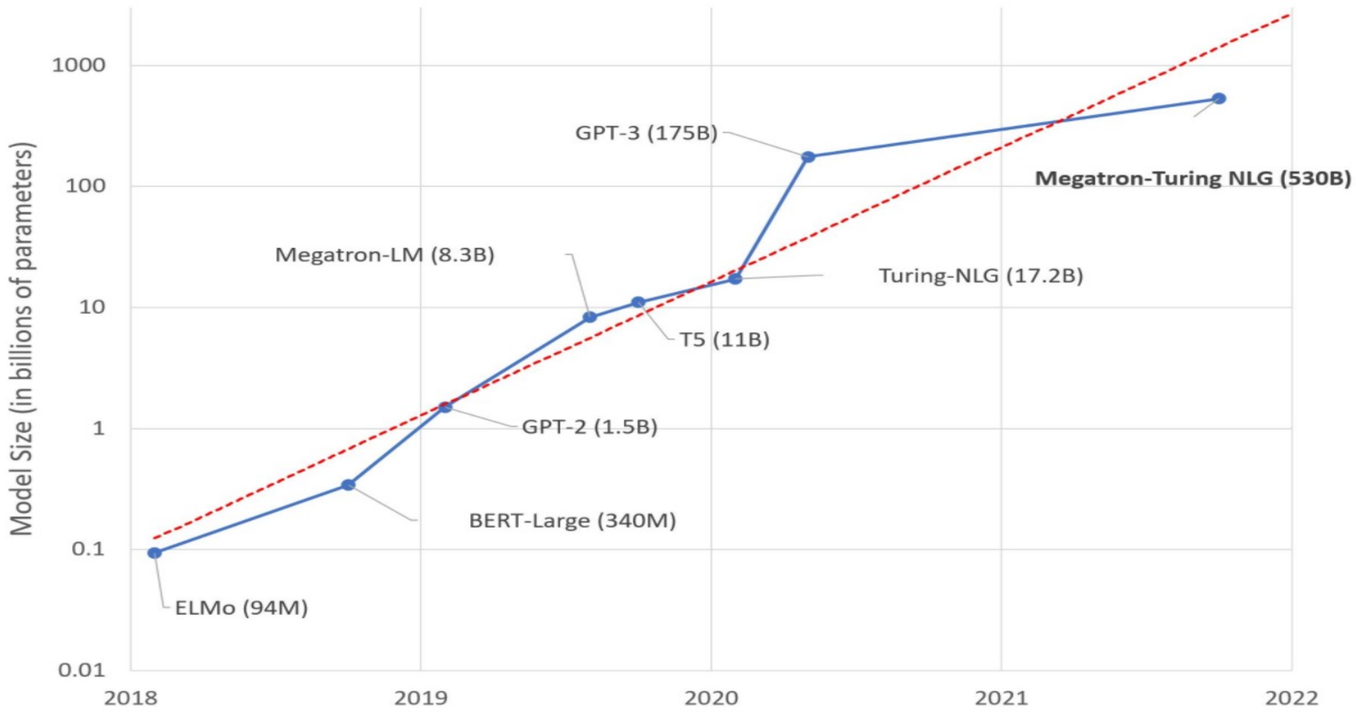
1.4T tokens

Impact of scale

Large Language Models

Large Language Models

- Not only they improved performance on many NLP tasks, but exhibited new capabilities



Power of scale and Large Language Models

- Paradigm shift:
  - Learn from only few examples
  - No need for transfer learning or additional finetuning ->
    - We can learn “in-context”

Classify the following review as positive or negative:

Review: The movie wasn't the worse but I wouldn't watch it again.
Label: Negative

Review: The acting was superb and the plot was well thought out.
Label: Positive

Review: The special effects were amazing, but the story was lacking.
Label: Mixed

Review: The pacing was slow and the ending was unsatisfying.
Label: Negative

Power of scale and Large Language Models

- Paradigm shift:
  - Learn from only few examples
  - No need for transfer learning or additional finetuning ->
    - We can learn “in-context”
  - Learning from task instructions:
    - No need for training examples, just describe the task

Instruction:

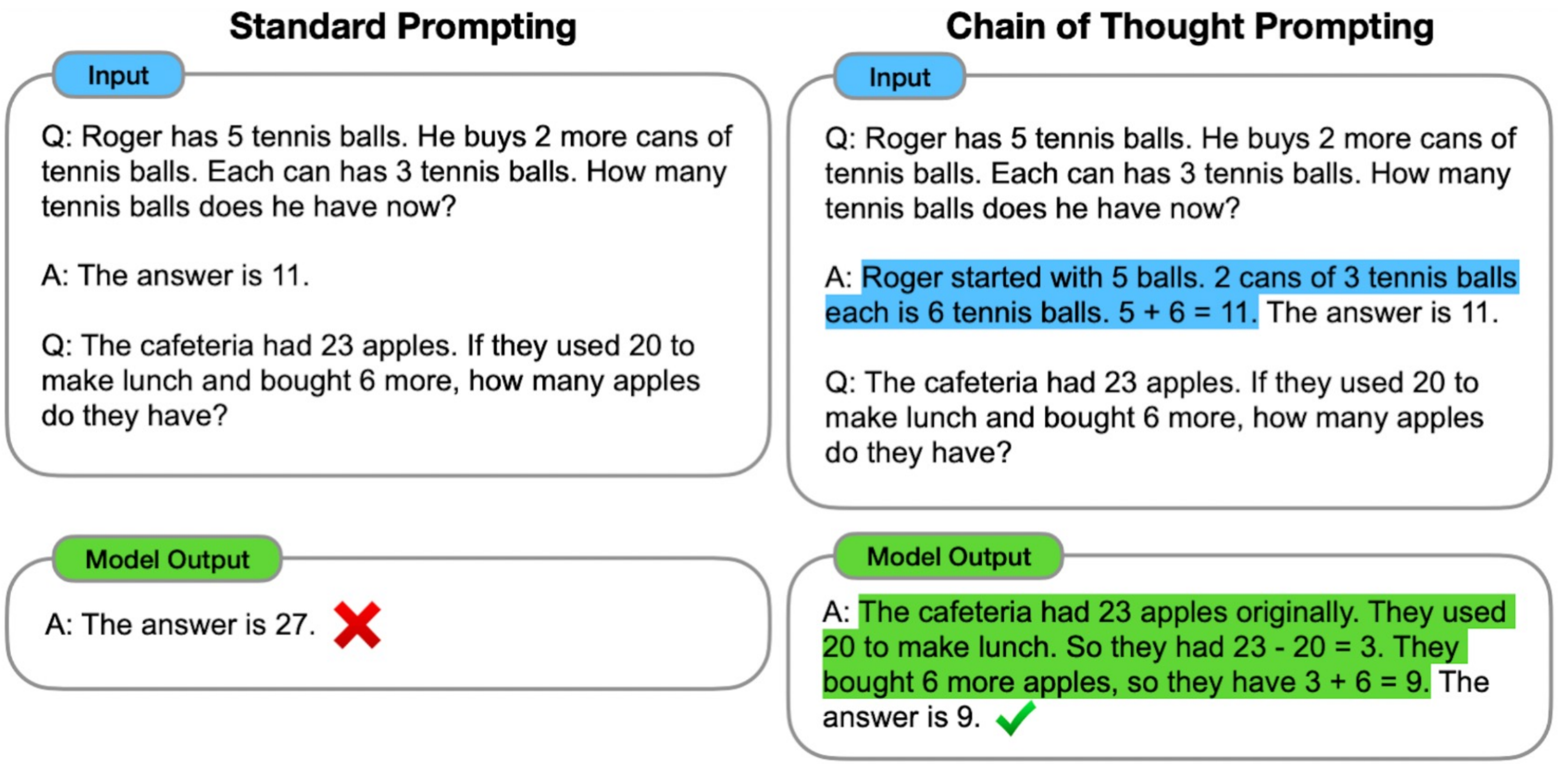
Classify the following review as positive, negative, or mixed:

Review: The movie wasn't the worse but I wouldn't watch it again.

Label: Mixed

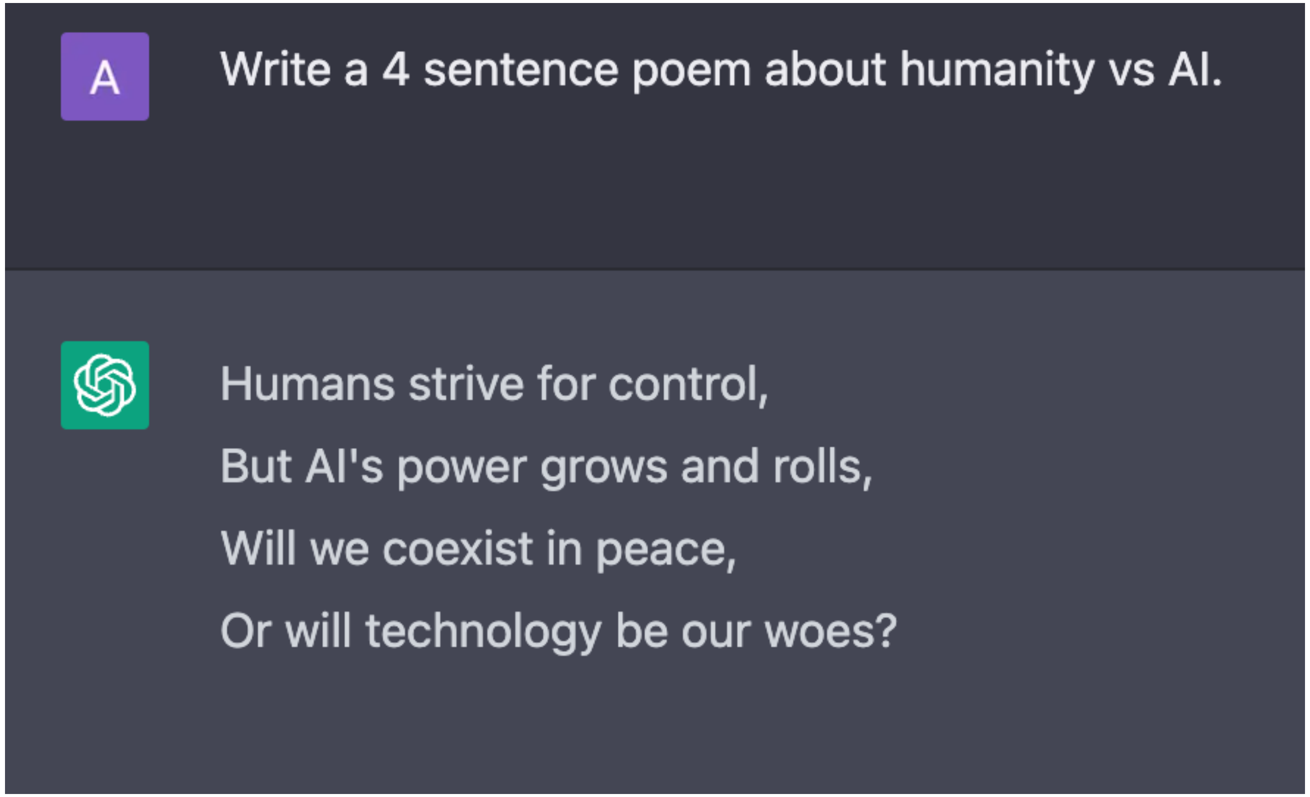
Power of scale and Large Language Models

- Chain of Thought prompting (Wei et al, 2022)



ChatGPT!

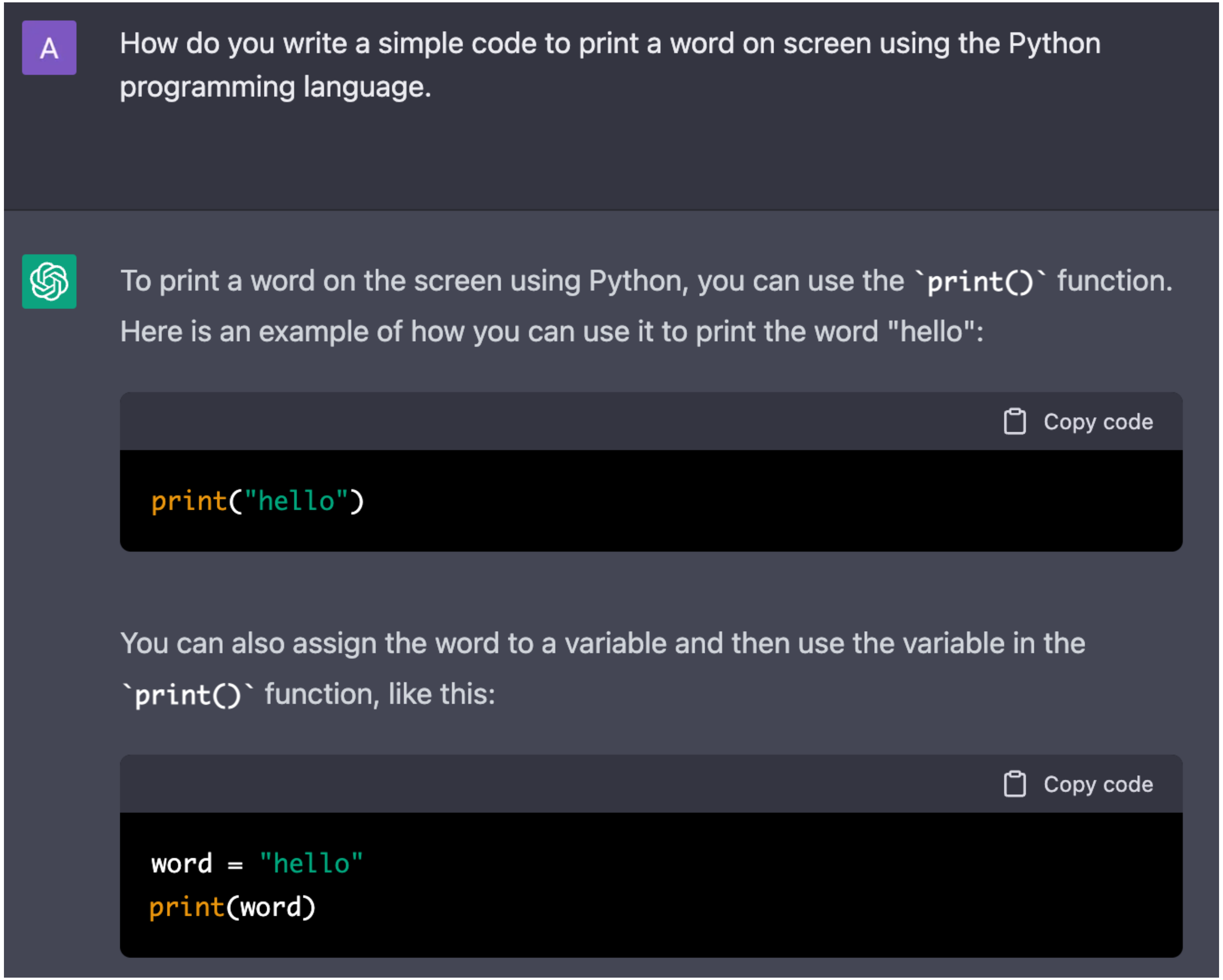
- Interaction and much higher quality generation



https://chat.openai.com/chat

ChatGPT!

•



https://chat.openai.com/chat

A

Why should I take the CPSC 670 (Advanced Topics in NLP) course?

The CPSC 670 (Advanced Topics in NLP) course can provide you with a deeper understanding of natural language processing techniques and their applications. It can help you develop advanced skills in areas such as language modeling, machine translation, and text summarization. Additionally, the course can expose you to current research in the field and provide you with an opportunity to work on projects that apply NLP techniques to real-world problems. Overall, taking CPSC 670 can help you gain expertise in NLP and be well-prepared for a career or further research in the field.

Coming up

- Transfer learning details
  - ULMFit, ELMo, BERT, T5
- We will post the exact schedule and papers later today
- We will assign the papers for presentation after next session