UPPSALA UNIVERSITET

Machine learning – Block 5

Måns Magnusson
Department of Statistics, Uppsala University

Autumn 2025

This week's lectures

- Word embeddings
- (Recurrent Neural Networks)
- Transformers
- Encoder-based (BERT-type) models

Practicalities

- One lecture later this week on word embeddings (Väinö Yrjänäinen) and encoder models (Hannes Waldetoft)

UPPSALA UNIVERSITET

Section 2

Word embeddings

How do we represent words?

- One-hot encoding
  - A vector of length V (vocabulary size)

$$\text{Uppsala} = [0, ..., 1, ..., 0] = \mathbf{1}_i$$

How do we represent words?

- One-hot encoding
  - A vector of length V (vocabulary size)

$$\text{Uppsala} = [0, ..., 1, ..., 0] = \mathbf{1}_i$$

- Word embeddings
  - A vector of length D (embedding dimension)

$$\text{Uppsala} = [-0.1231, ..., 1.9001, ..., 0.012]$$

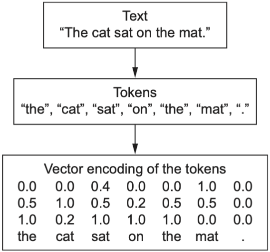

Figure: Representing words as word embeddings (Chollet and Allair, 2018, Fig. 6.1)

UPPSALA UNIVERSITET

- Practicalities
- Word embeddings
- Recurrent Neural Networks
- Transformers
  - Attention
  - Multi-Head Attention
  - Tokenization
  - Positional encoding
  - Add and Normalize
- Transformer-Encoder Models
  - BERT

Word embeddings vs. One-Hot

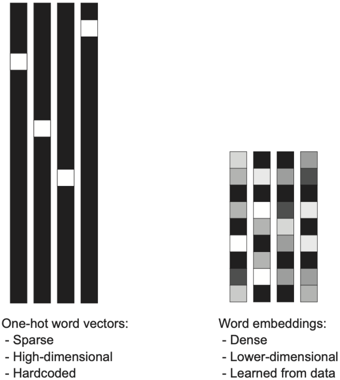

Figure: One-Hot vs. Word embeddings (Chollet and Allair, 2018, Fig. 6.2)

Word embeddings

The quick brown fox jumps over the lazy dog.

- A word type represents meaning in a low-dimensional semantic space
- The distributional hypothesis:
  - Harris (1954) and Firth (1957):
    "A word is characterized by the company it keeps"
  - Semantics (broadly defined) is captured by context
- Lots of different embeddings:
  `word2vec`, GloVe, Probabilistic Embeddings

UPPSALA UNIVERSITET

- Practicalities
- Word embeddings
- Recurrent Neural Networks
- Transformers
  - Attention
  - Multi-Head Attention
  - Tokenization
  - Positional encoding
  - Add and Normalize
- Transformer-Encoder Models
  - BERT

Word embeddings

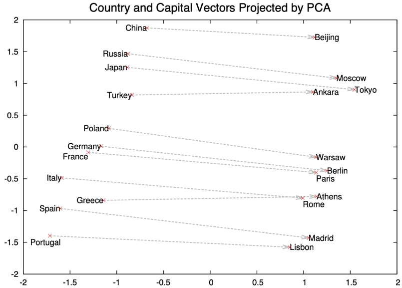

Figure: Word embedding properties (Mikolov et al, 2013)

$$\text{king} - \text{man} + \text{woman} \approx \text{queen}$$

UPPSALA UNIVERSITET

Word embeddings

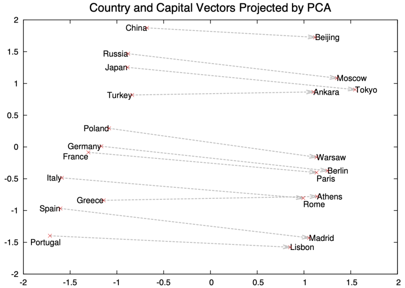

Figure: Word embedding properties (Mikolov et al, 2013)

$$\text{king} - \text{man} + \text{woman} \approx \text{queen}$$

But also (Bolukbasi et al., 2016):

$$\text{computer programmer} - \text{man} + \text{woman} \approx \text{homemaker}$$

Context Matters!

Figure: Context matters (Alammar, 2020)

- Static embeddings (word2vec, GloVe) assign *one vector per word type*.

Context Matters!

Figure: Context matters (Alammar, 2020)

- Static embeddings (word2vec, GloVe) assign *one vector per word type*.
- Contextual embeddings (BERT/Transformers) create *context-dependent vectors* per token.

Section 3

Recurrent Neural Networks

UPPSALA UNIVERSITET

Recurrent Neural Networks

- Recurrent Neural Networks, Recurrent Nets, RNN, ...
- Modeling of temporal data structures, such as
  - Time series data
  - Sequences of words (language models)

Recurrent Neural Networks

UPPSALA UNIVERSITET

- Recurrent Neural Networks, Recurrent Nets, RNN, ...
- Modeling of temporal data structures, such as
  - Time series data
  - Sequences of words (language models)
- Examples of applications (but gets rarer and rarer due to Transformers):
  - Time series predictions
  - Reinforcement learning

Recurrent Neural Networks

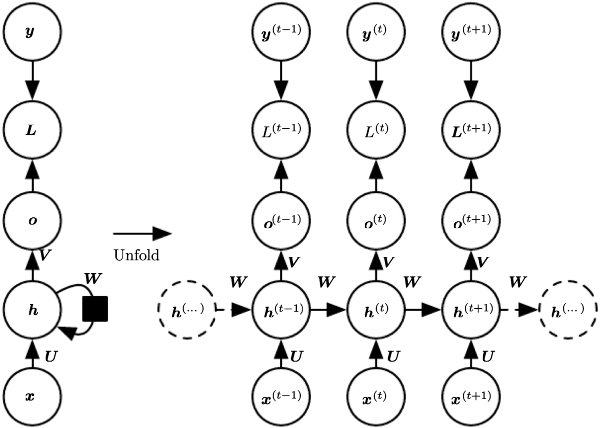

Figure: Recurrent Neural Network (Goodfellow et al, 2017, Fig. 10.3)

UPPSALA UNIVERSITET

Recurrent Neural Networks

$$
\begin{aligned}
a_t &= b + Wh_{t-1} + Ux_t \\
h_t &= \sigma_1(a_t) \\
o_t &= c + Vh_t \\
\hat{y}_t &= \sigma_{\text{output}}(o_t) = \text{softmax}(o_t)
\end{aligned}
$$

Think of h_t as the "state" at timepoint t and σ is an activation function (e.g. tanh or ReLU) and

$$
W \in \mathbb{R}^{H \times H}, \qquad U \in \mathbb{R}^{H \times D}, \qquad V \in \mathbb{R}^{D \times H} \text{ (if needed)}.
$$

where H is the number of hidden units and D is the input dimension.

UPPSALA UNIVERSITET

Recurrent network with one output

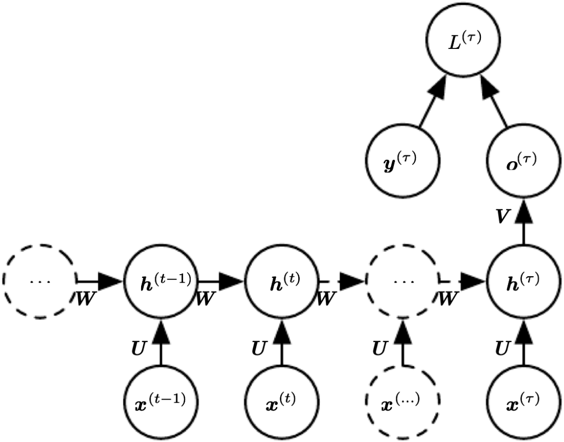

Figure: Recurrent Neural Network with one output (Goodfellow et al, 2017, Fig. 10.5)

Sequence to Sequence: Encoder-Decoder

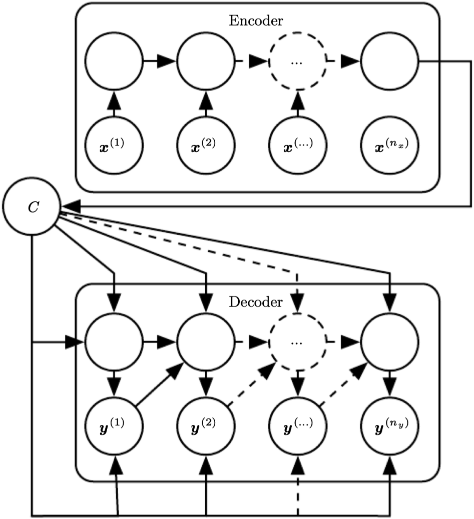

Figure: Encoder-Decoder Recurrent Networks (Goodfellow et al, 2017, Fig. 10.12)

Problems with RNN

- Exploding and vanishing gradients
- Long-term dependencies

Section 4

Transformers

The Transformer

- Introduced in 2017 in Vaswani et al. (2017)
- Behind the recent progress in NLP: BERT, GPT-series, Gemini etc.

The Transformer

- Introduced in 2017 in Vaswani et al. (2017)
- Behind the recent progress in NLP: BERT, GPT-series, Gemini etc.
- De-facto standard in industry and academia

The Transformer

- Introduced in 2017 in Vaswani et al. (2017)
- Behind the recent progress in NLP: BERT, GPT-series, Gemini etc.
- De-facto standard in industry and academia
- Four benefits:
  - Enables more parallelism
  - Better handling of long-range dependencies
  - Brings transfer learning to text data
  - Enables deeper networks

A Sequence-to-Sequence Model

Figure: Attention (Allamar, 2018)

Stacked Encoder-Decoder Structure

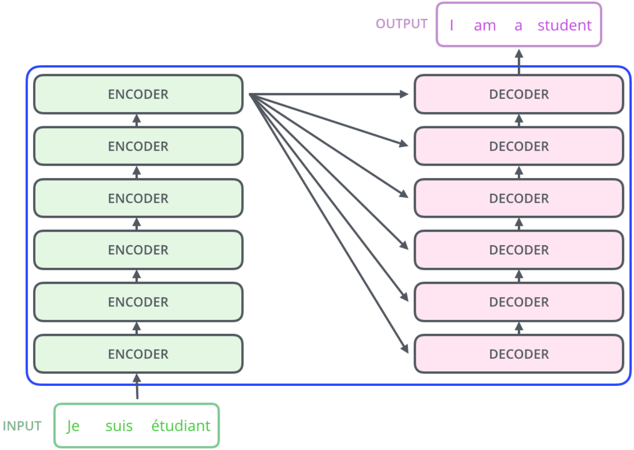

Figure: Attention (Allamar, 2018)

Transformer

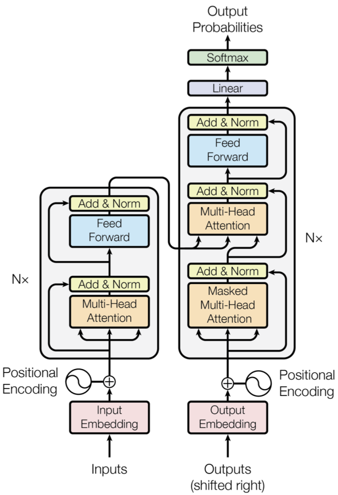

Figure: The Transformer Architecture (Vaswani et al., 2017)

The encoder vs. the decoder

- Encoder:
  - Input: words (embeddings)
  - Output: contextualized embeddings
- Decoder:
  - Input: contextualized embeddings and previous words (embeddings)
  - Output: words (embeddings)

The Transformer Layer (Encoder layer)

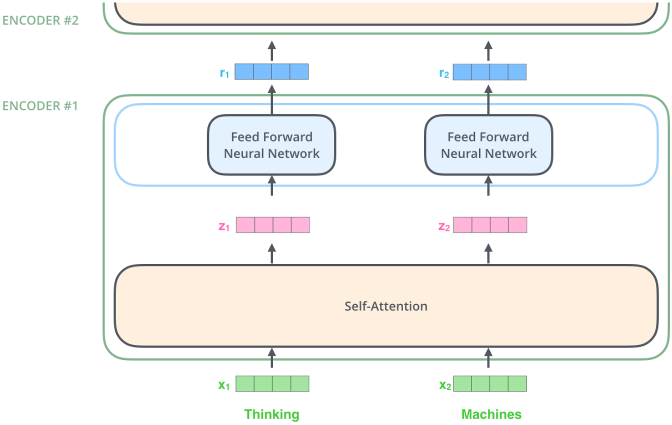

Figure: The Encoder Layer (Alammar, 2018b)

Scaled Dot-Product Attention

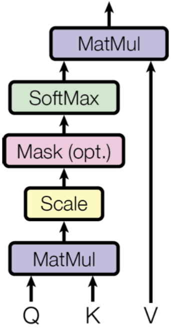

Figure: Scaled Dot-Product Attention (Vaswani et al., 2017)

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V},$$

where the scaling factor $\sqrt{d_k}$ stabilizes gradients.

Attention components

Self-attention allows each word to decide which other words are most relevant to interpreting its meaning.

- (Q)uery: Word i query other words
- (K)ey: The other words return their key to i
- (V)alue: The value of the other words to i

UPPSALA UNIVERSITET

Computing Q, V and K

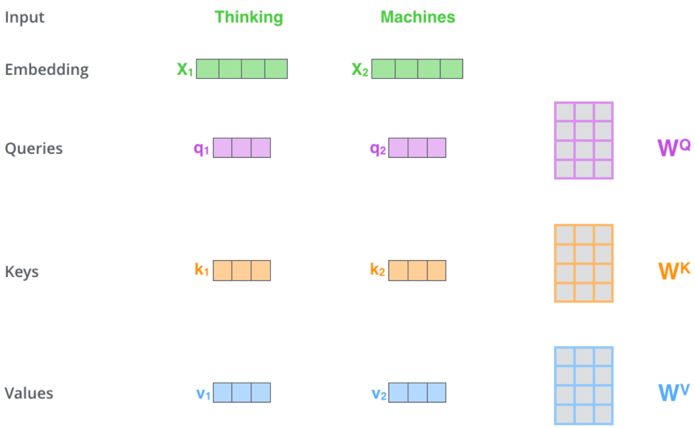

Figure: Attention heads (Alammar, 2018b)

UPPSALA UNIVERSITET

Computing Self-Attention

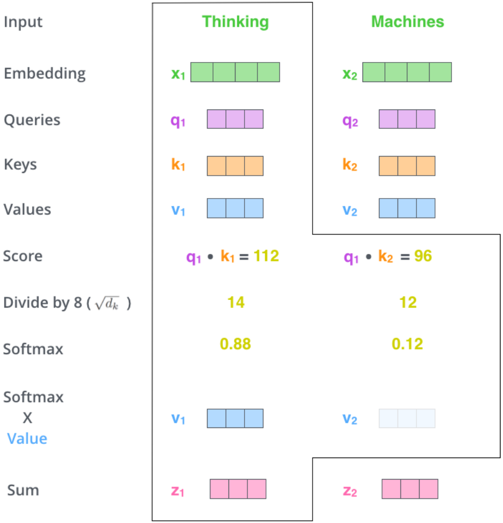

Figure: Attention (Alammar, 2018b)

Self-Attention Example: Setup

Sentence: "The cat sat"

Use simple 2D embeddings:

$$\text{the} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \text{cat} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \text{sat} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

We compute self-attention for the token "cat".

Parameter matrices (toy example):

$$W_Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad W_K = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad W_V = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

Step 1: Compute Q, K, and V

Query for "cat":

$$Q_{\text{cat}} = W_Q \cdot \text{cat} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Keys:

$$K_{\text{the}} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad K_{\text{cat}} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad K_{\text{sat}} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Values:

$$V_{\text{the}} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad V_{\text{cat}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad V_{\text{sat}} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Step 2: Attention Scores and Weights

Dot-product scores for "cat":

$$\left[Q_{\text{cat}}^{\top}K_{\text{the}},\, Q_{\text{cat}}^{\top}K_{\text{cat}},\, Q_{\text{cat}}^{\top}K_{\text{sat}}\right] = [0,\ 1,\ 1]$$

Softmax weights (**QK**):

$$\text{weights} = \mathrm{softmax}([0,\ 1,\ 1]) = [0.21,\ 0.39,\ 0.39]$$

Step 3: Output Representation for "cat"

UPPSALA UNIVERSITET

Weighted sum of values:

$$\text{Output}_{\text{cat}} = 0.21 \cdot V_{\text{the}} + 0.39 \cdot V_{\text{cat}} + 0.39 \cdot V_{\text{sat}}$$

$$= 0.21 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 0.39 \begin{bmatrix} 0 \\ 1 \end{bmatrix} + 0.39 \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0.60 \\ 1.59 \end{bmatrix}$$

This is the new embedding for the token "cat" after self-attention.

Multi-Head Attention

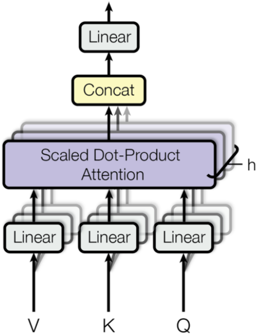

Figure: Scaled Dot-Product Attention (Vaswani et al., 2017)

Attentions Heads

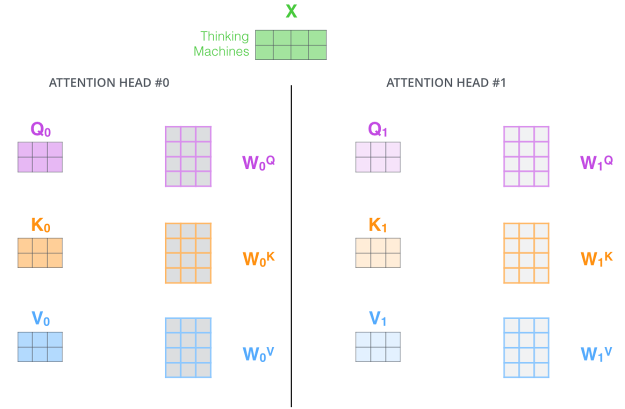

Figure: Attention heads (Alammar, 2018b)

Multi-head attention

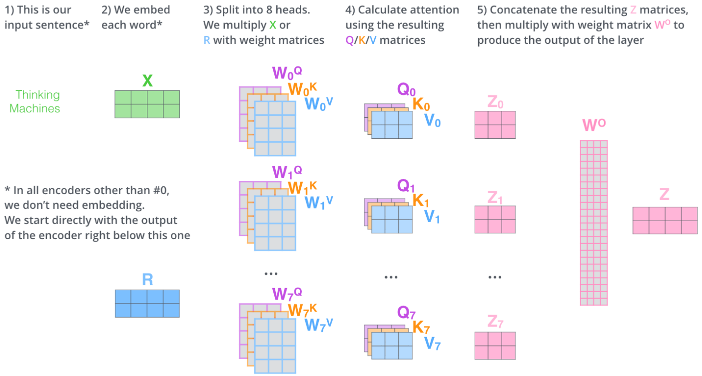

Figure: Attention heads (Alammar, 2018b)

Multi-Head Attention example

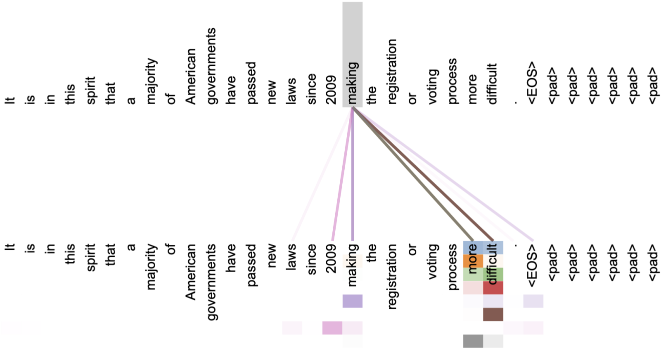

Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb ‘making’, completing the phrase ‘making...more difficult’. Attentions here shown only for the word ‘making’. Different colors represent different heads. Best viewed in color.

Figure: Attention (Vaswani et al., 2017)

Attention weight = how much word i focuses on word j when computing its contextual representation.

Tokenization

- Subword tokenization is commonly used

UPPSALA UNIVERSITET

Tokenization

- Subword tokenization is commonly used
- The main problem with tokenization
  1. Very large vocabulary size
  2. Out-of-vocabulary (OOV) tokens
  3. Different meanings of very similar words

UPPSALA UNIVERSITET

Tokenization

- Subword tokenization is commonly used
- The main problem with tokenization
  1. Very large vocabulary size
  2. Out-of-vocabulary (OOV) tokens
  3. Different meanings of very similar words
- Two common approaches:
  1. Byte-pair encoding (GPT, RoBERTa)
  2. WordPiece (BERT)

Byte-pair encoding

- Gage, Philip (1994). "A New Algorithm for Data Compression"

Byte-pair encoding

- Gage, Philip (1994). "A New Algorithm for Data Compression"
- Encode the most common pairs iteratively
  1. look for the most frequent pairing
  2. merge them
  3. repeat (until token or iteration limit)

Byte-pair encoding

- Gage, Philip (1994). "A New Algorithm for Data Compression"
- Encode the most common pairs iteratively
  1. look for the most frequent pairing
  2. merge them
  3. repeat (until token or iteration limit)
- Example (Wikipedia): `aaabdaaabac`

Byte-pair encoding

- Gage, Philip (1994). "A New Algorithm for Data Compression"
- Encode the most common pairs iteratively
  1. look for the most frequent pairing
  2. merge them
  3. repeat (until token or iteration limit)
- Example (Wikipedia): `aaabdaaabac`

  Step 1: `ZabdZabac`
  `Z=aa`

Byte-pair encoding

- Gage, Philip (1994). "A New Algorithm for Data Compression"
- Encode the most common pairs iteratively
  1. look for the most frequent pairing
  2. merge them
  3. repeat (until token or iteration limit)
- Example (Wikipedia): `aaabdaaabac`

  Step 1: `ZabdZabac`
  `Z=aa`

  Step 2: `ZYdZYac`
  `Z=aa`
  `Y=ab`

Byte-pair encoding

- Gage, Philip (1994). "A New Algorithm for Data Compression"
- Encode the most common pairs iteratively
  1. look for the most frequent pairing
  2. merge them
  3. repeat (until token or iteration limit)
- Example (Wikipedia): `aaabdaaabac`

  Step 1: `ZabdZabac`
  `Z=aa`

  Step 2: `ZYdZYac`
  `Z=aa`
  `Y=ab`

  Step 3: `XdXac`
  `Z=aa`
  `Y=ab`
  `X=ZY`

Byte-pair encoding

1. look for the most frequent pairing
2. merge them
3. repeat (until token or iteration limit)

- Example : 9:`text_`, 10:`texting_`,11:`context_`

Byte-pair encoding

1. look for the most frequent pairing
2. merge them
3. repeat (until token or iteration limit)

- Example : 9:`text_`, 10:`texting_`,11:`context_`

  Step 1 (most common: "te"): {`te`}

Byte-pair encoding

1. look for the most frequent pairing
2. merge them
3. repeat (until token or iteration limit)

- Example : 9:`text_`, 10:`texting_`,11:`context_`

  Step 1 (most common: "te"): {`te`}

  ...

  Step i (most common: "text_"): {`text_`}

Byte-pair encoding

1. look for the most frequent pairing
2. merge them
3. repeat (until token or iteration limit)

- Example : 9:`text_`, 10:`texting_`,11:`context_`

  Step 1 (most common: "te"): {`te`}

  ...

  Step i (most common: "text_"): {`text_`}

  ...

  Step j (most common: "con"): {`text_`,con}

UPPSALA UNIVERSITET

Byte-pair encoding

1. look for the most frequent pairing
2. merge them
3. repeat (until token or iteration limit)

- Example : 9:`text_`, 10:`texting_`,11:`context_`

  Step 1 (most common: "te"): {`te`}

  ...

  Step i (most common: "text_"): {`text_`}

  ...

  Step j (most common: "con"): {`text_`,con}

  ...

  Step k (most common: "texting"): {`text_`,con,`texting`}

Byte-pair encoding

1. look for the most frequent pairing
2. merge them
3. repeat (until token or iteration limit)

- Example : 9:`text_`, 10:`texting_`,11:`context_`

  Step 1 (most common: "te"): {`te`}

  ...
  Step i (most common: "text_"): {`text_`}

  ...
  Step j (most common: "con"): {`text_`,con}

  ...
  Step k (most common: "texting"): {`text_`,con,`texting`}

WordPiece

- BPE difficulty: Which pair to choose (if they are approx. equally frequent)?

UPPSALA UNIVERSITET

WordPiece

- BPE difficulty: Which pair to choose (if they are approx. equally frequent)?
- Schuster and Nakajima (2012) present the WordPiece model

WordPiece

- BPE difficulty: Which pair to choose (if they are approx. equally frequent)?
- Schuster and Nakajima (2012) present the WordPiece model
- Let $P(i, j)$ be the probability of observing the pair ij and $P(i)$ observing i.

WordPiece

- BPE difficulty: Which pair to choose (if they are approx. equally frequent)?
- Schuster and Nakajima (2012) present the WordPiece model
- Let $P(i, j)$ be the probability of observing the pair ij and $P(i)$ observing i.
- BPE: Choose highest $P(i, j)$

WordPiece

- BPE difficulty: Which pair to choose (if they are approx. equally frequent)?
- Schuster and Nakajima (2012) present the WordPiece model
- Let $P(i, j)$ be the probability of observing the pair ij and $P(i)$ observing i.
- BPE: Choose highest $P(i, j)$
- Wordpiece: Choose highest $P(i, j)/(P(i)P(j))$

Positional Encoding

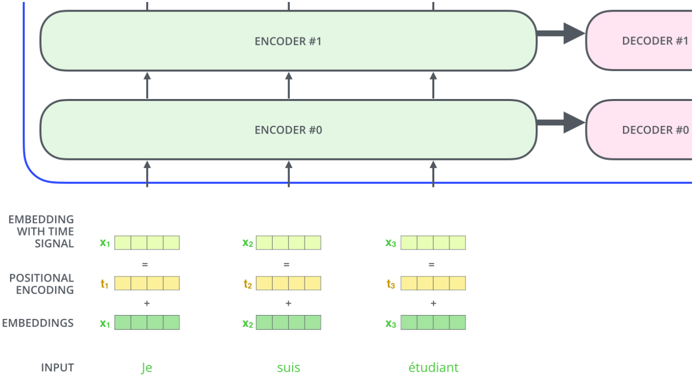

Figure: Attention heads (Alammar, 2018b)

UPPSALA UNIVERSITET

- Practicalities
- Word embeddings
- Recurrent Neural Networks
- Transformers
  - Attention
  - Multi-Head Attention
  - Tokenization
  - Positional encoding
  - Add and Normalize
- Transformer-Encoder Models
  - BERT

Positional Encoding

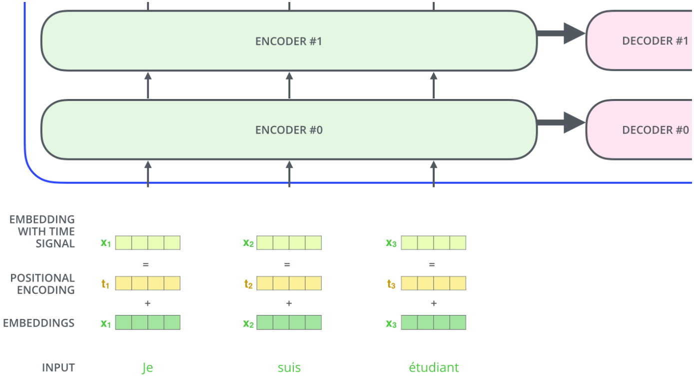

Figure: Attention heads (Alammar, 2018b)

Transformer attention is order-invariant, so positional encodings inject order information.

Positional Encoding

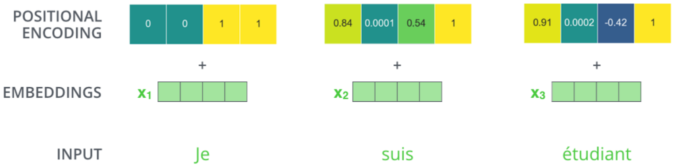

Figure: Adding positional encodings to embeddings (Alammar, 2018b)

Add and Normalize

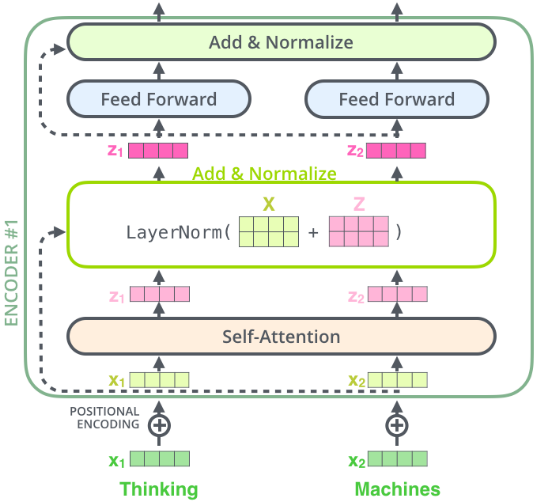

Figure: Add and Normalize (Alammar, 2018b)

Transformer-Encoder Models

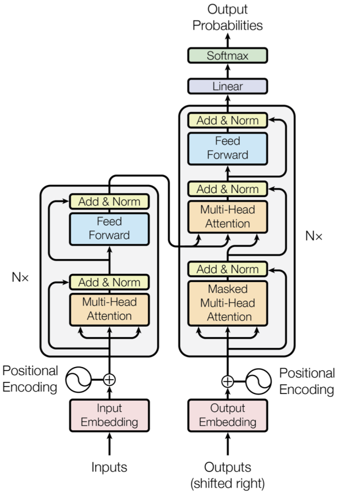

Figure: The Transformer Architecture (Vaswani et al., 2017)

Transformer-Encoder Models

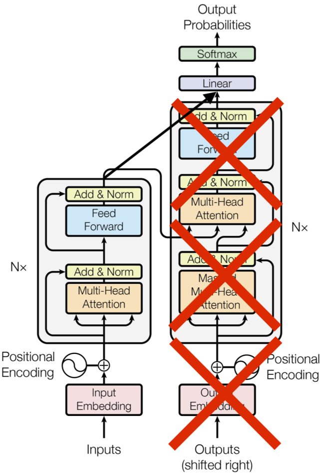

Figure: The Transformer Architecture (Vaswani et al., 2017)

Transformer-Encoder Models

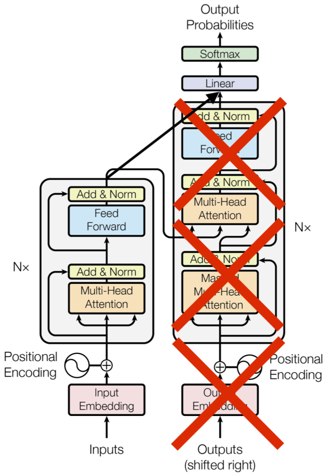

Figure: The Transformer Architecture (Vaswani et al., 2017)

- Common models are BERT, RoBERTa and ModernBERT

BERT

- Bidirectional Encoder Representations from Transformers (BERT)
- Introduced in Devlin et al. (2018)

BERT

- Bidirectional Encoder Representations from Transformers (BERT)
- Introduced in Devlin et al. (2018)
- (Still) State-of-the-Art in many text prediction tasks, such as
  - Named-Entity Recognition
  - Text Classification

BERT

- Bidirectional Encoder Representations from Transformers (BERT)
- Introduced in Devlin et al. (2018)
- (Still) State-of-the-Art in many text prediction tasks, such as
  - Named-Entity Recognition
  - Text Classification
- Many flavors, such as RoBERTa, AlBERT, ModernBERT etc.

UPPSALA UNIVERSITET

BERT

- Bidirectional Encoder Representations from Transformers (BERT)
- Introduced in Devlin et al. (2018)
- (Still) State-of-the-Art in many text prediction tasks, such as
  - Named-Entity Recognition
  - Text Classification
- Many flavors, such as RoBERTa, AlBERT, ModernBERT etc.
- Pre-trained on a large corpus

BERT

- Bidirectional Encoder Representations from Transformers (BERT)
- Introduced in Devlin et al. (2018)
- (Still) State-of-the-Art in many text prediction tasks, such as
  - Named-Entity Recognition
  - Text Classification
- Many flavors, such as RoBERTa, AlBERT, ModernBERT etc.
- Pre-trained on a large corpus
- Then fine-tuned for a specific problem

BERT (continued)

- Available in English, Swedish, and many other languages (e.g. The National Library)

BERT (continued)

- Available in English, Swedish, and many other languages (e.g. The National Library)
- Note!
  - BERT = encoder-only (no generation)
  - As we will see, GPT = decoder-only (generation)

BERT (continued)

- Available in English, Swedish, and many other languages (e.g. The National Library)
- Note!
  - BERT = encoder-only (no generation)
  - As we will see, GPT = decoder-only (generation)
- And again, I rely on Alammar (2018): The Illustrated BERT

BERT and transfer learning

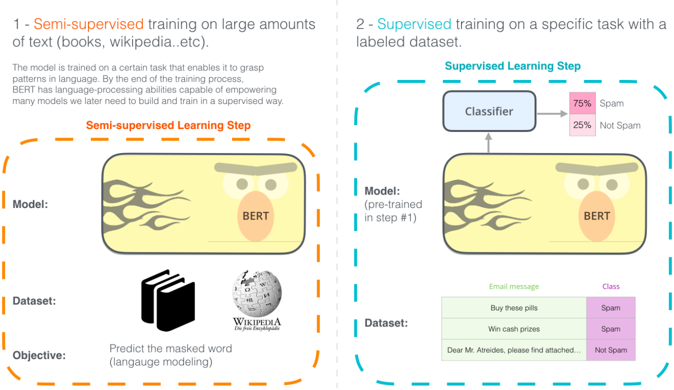

Figure: Using BERT for Transfer Learning (Alammar, 2018b)

UPPSALA UNIVERSITET

The BERT model

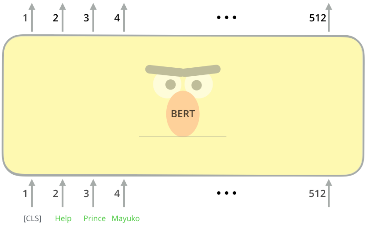

Figure: The BERT model (Alammar, 2018b)

UPPSALA UNIVERSITET

BERT Architecture

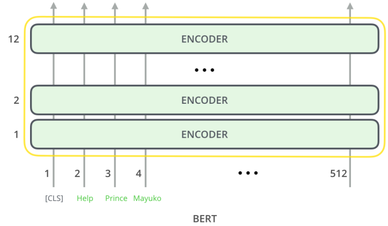

Figure: Opening up BERT (Alammar, 2018b)

Training Task 1: Masked Language Model

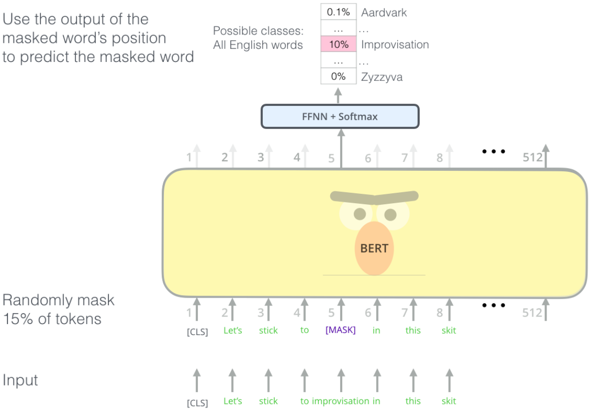

Figure: Masked Language Modeling (Alammar, 2018c)

UPPSALA UNIVERSITET

- Practicalities
- Word embeddings
- Recurrent Neural Networks
- Transformers
  - Attention
  - Multi-Head Attention
  - Tokenization
  - Positional encoding
  - Add and Normalize
- Transformer-Encoder Models
  - BERT

Training task 2: Next Sentence Prediction

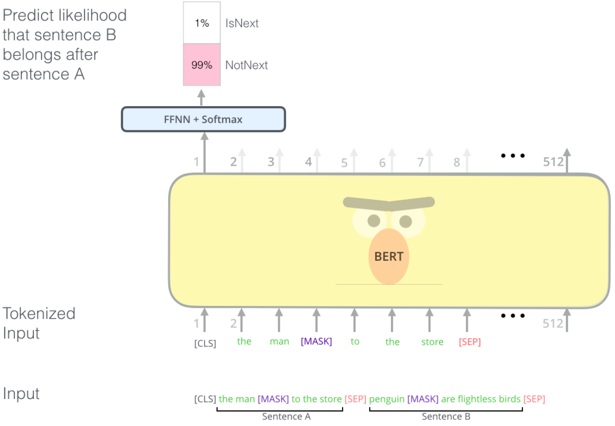

Figure: Next Sentence Prediction (Alammar, 2018c)

Using BERT for Classification

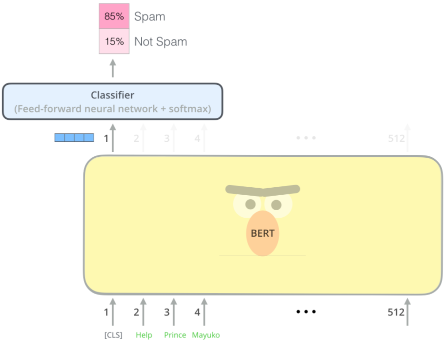

Figure: Using BERT for classification (Alammar, 2018c)

UPPSALA UNIVERSITET

BERT and Contextualized embeddings

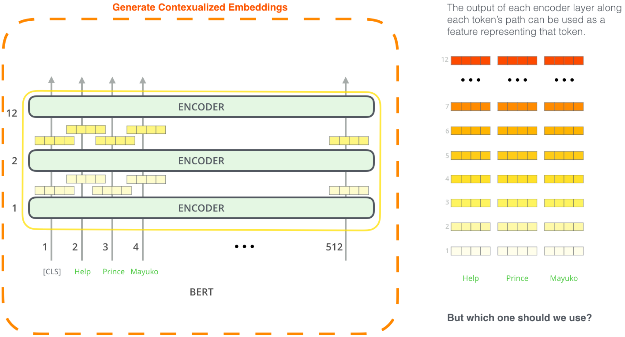

Figure: Contextualized Embeddings (Alammar, 2018c)

Using Contextualized Embeddings

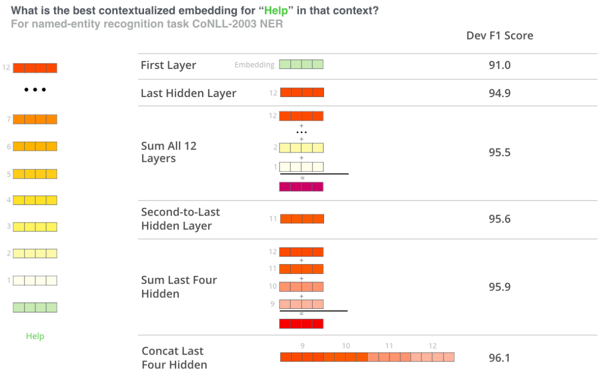

Figure: Using Contextualized Embeddings (Alammar, 2018c)