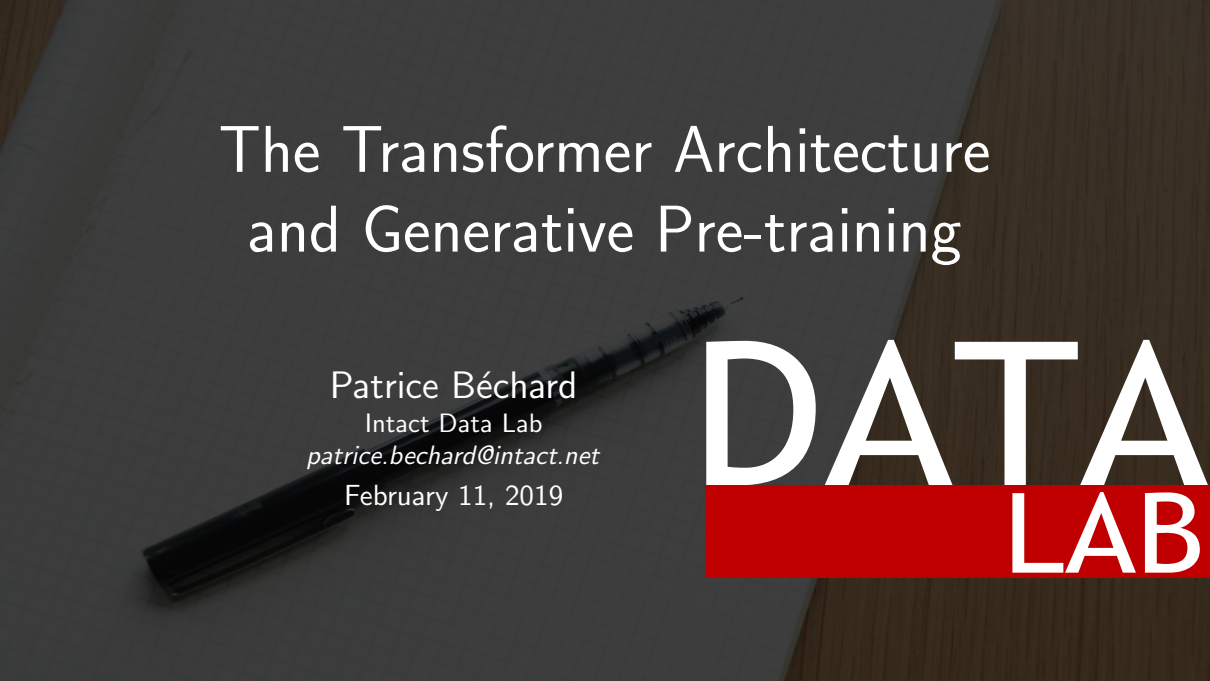
The Transformer Architecture and Generative Pre-training

Patrice Béchard
Intact Data Lab
patrice.bechard@intact.net

February 11, 2019



Why are these recent advances important?

- *Classical* Neural Machine Translation (NMT) approaches are difficult to parallelize and take a long time to train.
- Annotated datasets are expensive to build (time and resources).
- We have an *infinite* amount of raw textual data that we can now leverage.

Plan

The Task

Translating a source sentence

$$\mathbf{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}\}$$

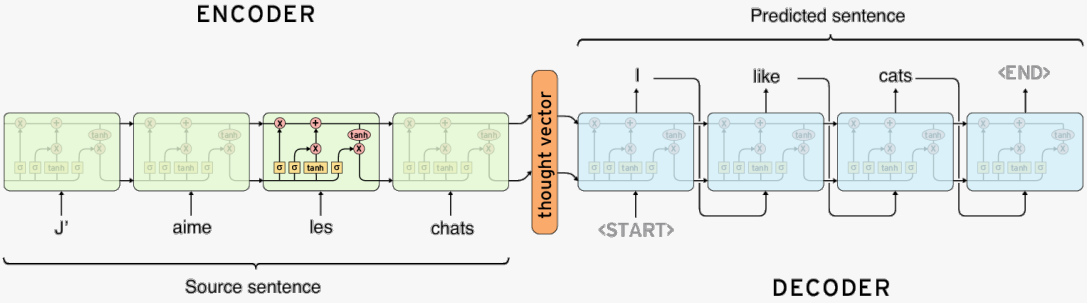
into a target sentence

$$\mathbf{Y} = \{\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(m)}\}$$

- In practice, we have huge amounts of parallel texts (e.g. parliament documents from EU, Canada, ...)

Neural Machine Translation

The Sequence to Sequence (Seq2Seq) model [5, 20]



- Difficult to learn long-term dependencies, even with a LSTM network.

Neural Machine Translation

For a sequence to sequence model (Seq2Seq), we can express the conditional probability of $\mathbf{Y}$ being outputted by the decoder given $\mathbf{X}$:

$$p(\mathbf{X}|\mathbf{Y}) = \prod_{t'=1}^{m} p(\mathbf{y}^{(t')}|\mathbf{X}, \mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(t'-1)})$$

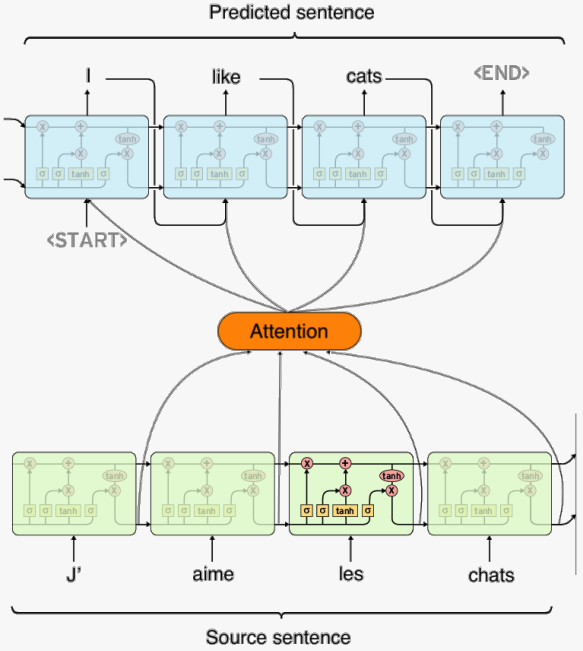
Our goal is to maximize the probability of our model producing an accurate translation. Given the *ground truth* translation $\mathbf{Y}$, the source sequence $\mathbf{X}$ and a training set $\mathcal{D}$, we train oure model to maximize the following objective :

$$\mathcal{L} = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{X},\mathbf{Y})\in\mathcal{D})} \log p(\mathbf{Y}|\mathbf{X})$$

Neural Machine Translation

The Attention Mechanism [3, 13]

- Shortcut between word from source sentence and target sentence
- Allows the model to jointly learn to :
  - **align** words from the source sentence with words from the target sentence;
  - **translate** the source sentence to the target sentence.



How to compute attention weights?

- We compute a weighted sum of all the encoder's outputs (the *context vector* $\mathbf{c}^{(t')}$) for each predicted word.

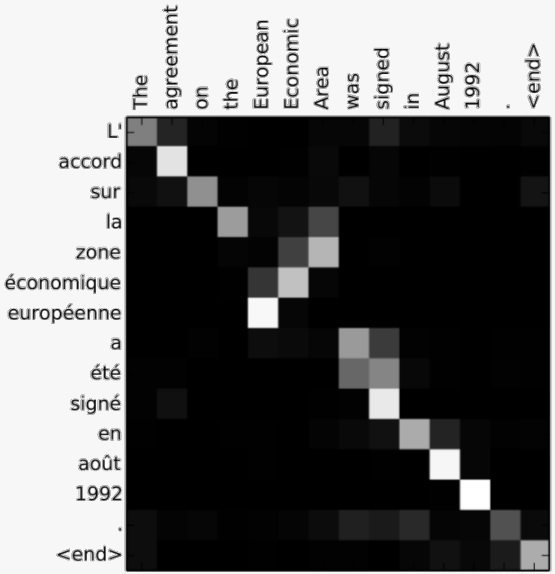
$$e^{(t',t)} = \mathbf{v}_a^T \tanh\left(\mathbf{W}_a \mathbf{s}^{(t'-1)} + \mathbf{U}_a \mathbf{h}^{(t)}\right)$$

$$\alpha^{(t',t)} = \frac{\exp e^{(t',t)}}{\sum_{t=1}^{n} \exp e^{(t',t)}}$$

$$\mathbf{c}^{(t')} = \sum_{t=1}^{n} \alpha^{(t',t)} \mathbf{h}^{(t)}$$

Visualizing Attention Weights

- Here we visualize the $\alpha^{(t',t)}$ from the previous slide.
- The model learns to align word groups together properly.



Limitations

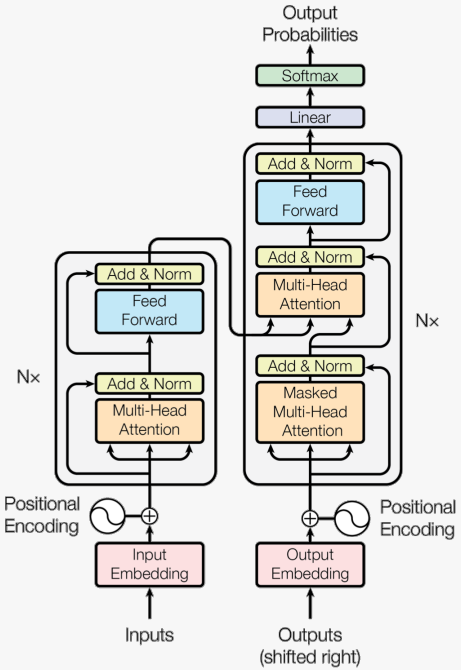
- The model is inherently sequential. The computing time depends on the length of the input sequence as well as the output sequence.
- In most cases, we know the entirety of the input sequence before beginning the translation. We can try to leverage this in order to process the sequences faster.
- Many people have tried to tackle this problem [9, 11, 8]

The Transformer Architecture

The Transformer Architecture

The Transformer Architecture [21]

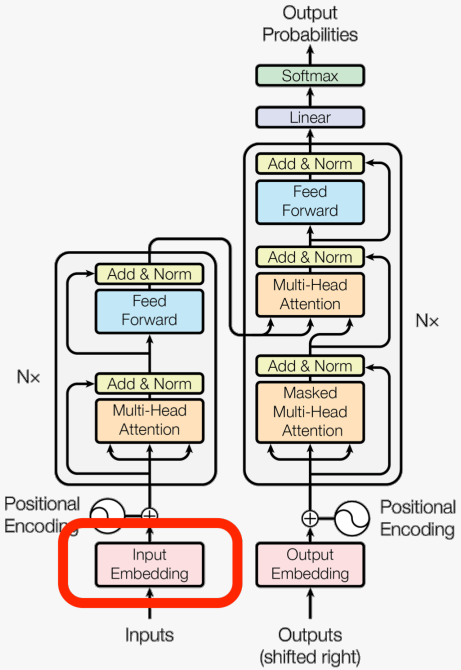
- No use of recurrent neural networks
- We still have the encoder-decoder architecture
- Composed of multiple parts :
  - Input Embedding
  - Position Embedding
  - Multi-Head Attention
  - Feed Forward Network



The Transformer Architecture

The Transformer Architecture [21]

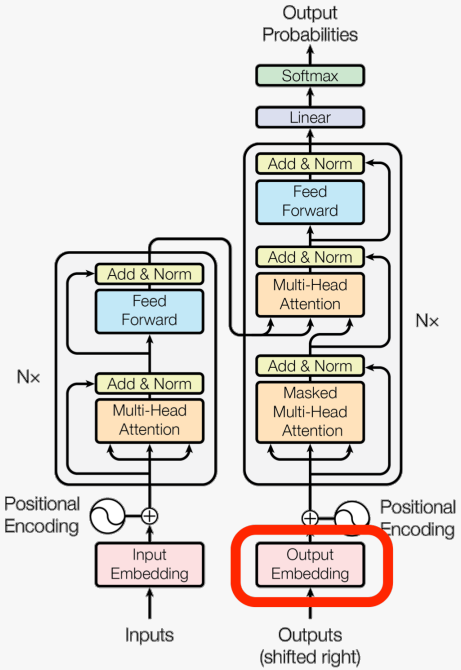
- No use of recurrent neural networks
- We still have the encoder-decoder architecture
- Composed of multiple parts :
  - Input Embedding
  - Position Embedding
  - Multi-Head Attention
  - Feed Forward Network



The Transformer Architecture

The Transformer Architecture [21]

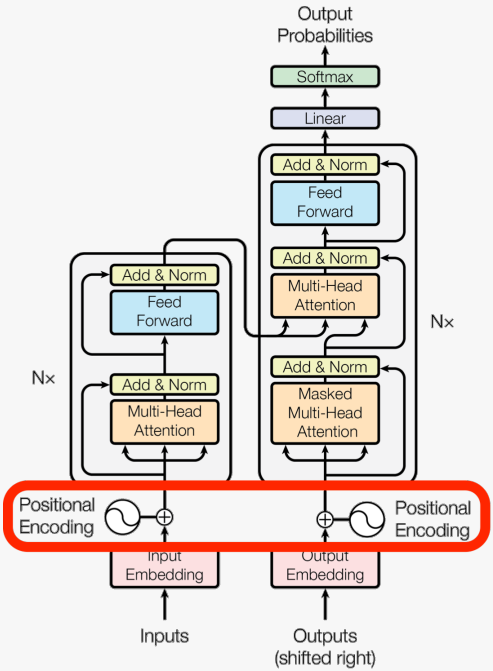
- No use of recurrent neural networks
- We still have the encoder-decoder architecture
- Composed of multiple parts :
  - Input Embedding
  - Position Embedding
  - Multi-Head Attention
  - Feed Forward Network



The Transformer Architecture

The Transformer Architecture [21]

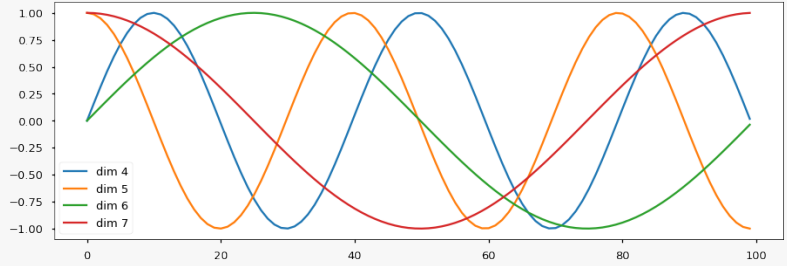
- No use of recurrent neural networks
- We still have the encoder-decoder architecture
- Composed of multiple parts :
  - Input Embedding
  - Position Embedding
  - Multi-Head Attention
  - Feed Forward Network



Position Embeddings

- Can be either learned (via backpropagation) or defined as a bunch of sines and cosines.

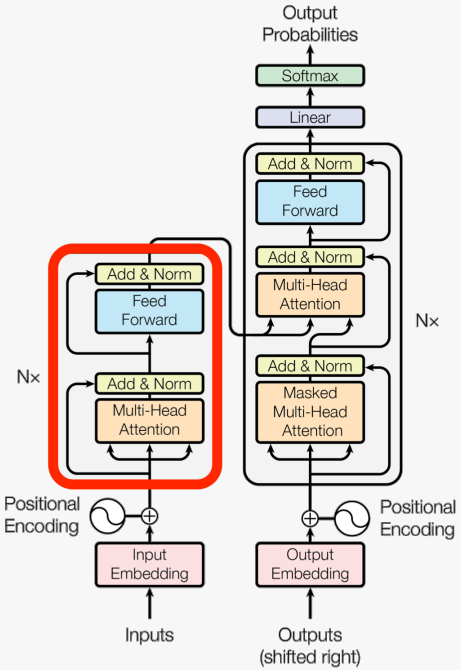
$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \qquad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$



The Transformer Architecture

The Transformer Architecture [21]

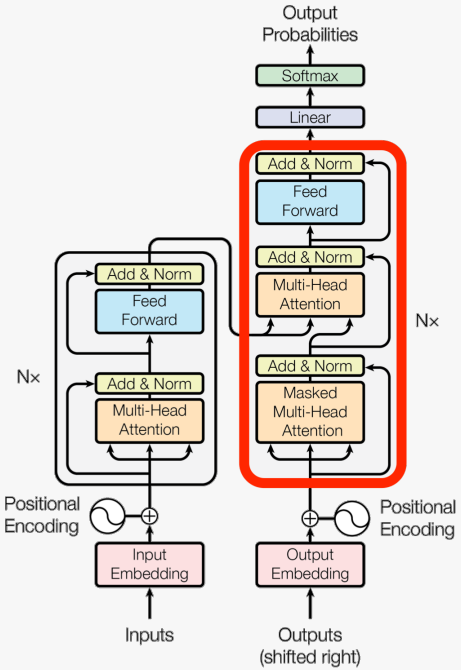
- No use of recurrent neural networks
- We still have the encoder-decoder architecture
- Composed of multiple parts :
  - Input Embedding
  - Position Embedding
  - Multi-Head Attention
  - Feed Forward Network



The Transformer Architecture

The Transformer Architecture [21]

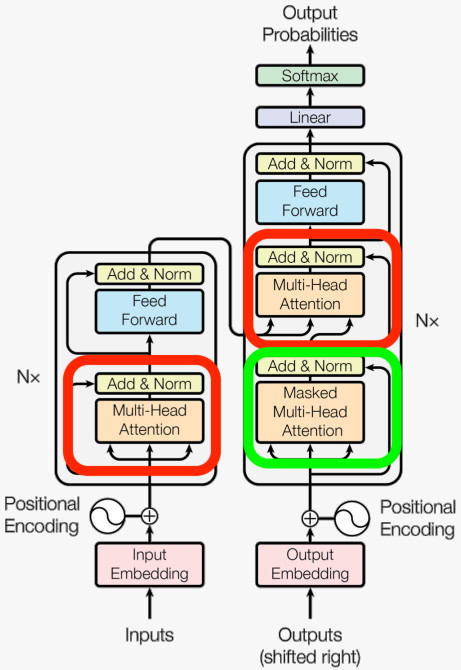
- No use of recurrent neural networks
- We still have the encoder-decoder architecture
- Composed of multiple parts :
  - Input Embedding
  - Position Embedding
  - Multi-Head Attention
  - Feed Forward Network



The Transformer Architecture

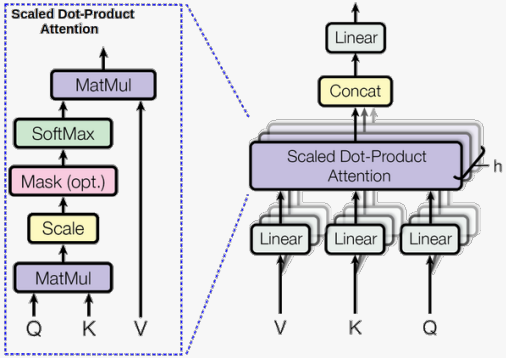
The Transformer Architecture [21]

- No use of recurrent neural networks
- We still have the encoder-decoder architecture
- Composed of multiple parts :
  - Input Embedding
  - Position Embedding
  - Multi-Head Attention
  - Feed Forward Network



The Transformer Architecture

Multi-Head Attention



Multi-Head Attention

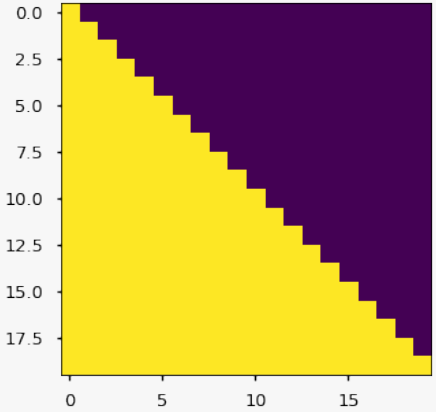
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\text{head}_i = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right)$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O$$

- where $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W_O \in \mathbb{R}^{hd_v \times d_{model}}$, with $d_k = d_v = d_{model}/h$.
- **Dot product attention** is much faster and space efficient than Bahdanau's attention.
- The scaling allows the variance of the dot product to be ≈ 1.

Masked Multi-Head Attention

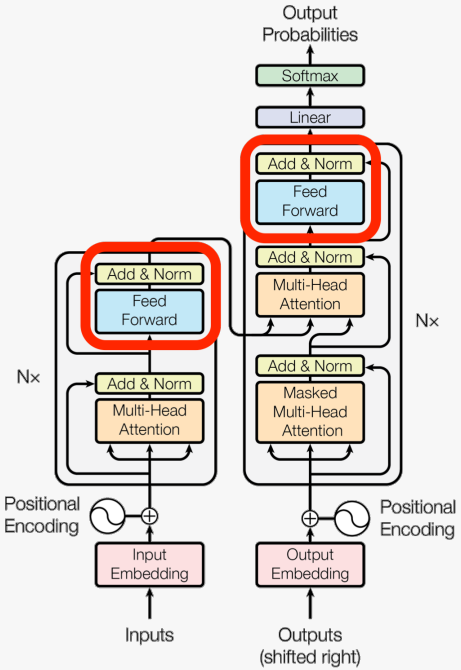


- Used so that the model cannot attend to words in the future.

The Transformer Architecture

The Transformer Architecture [21]

- No use of recurrent neural networks
- We still have the encoder-decoder architecture
- Composed of multiple parts :
  - Input Embedding
  - Position Embedding
  - Multi-Head Attention
  - Feed Forward Network



Position-wise feedforward layer

$$\mathrm{FFN}(x) = \mathrm{ReLU}(xW_1 + b_1)W_2 + b_2$$

- where $W_1 \in \mathbb{R}^{d_{model} \times d_{ff}}$ and $W_2 \in \mathbb{R}^{d_{ff} \times d_{model}}$
- Can also be seen as two convolutions with kernel of size 1.

Residual Connections and Layer Normalization

- Helps with the gradient propagation for very deep neural networks [10, 2]

Residual layer

$$\text{Residual}(x) \approx f(x) + x$$

Layer Normalization

$$\text{LayerNorm}(x) \approx \frac{x - \mu}{\sigma + \epsilon}$$

Limitations

- Algorithm is $\mathcal{O}(n^2)$
- The maximum size of the input is limited (Solved by Transformer-XL [6])

Two Approaches to Transfer Learning

Feature-based Approach

- Using pre-trained representations as features for the model
- The architecture of the model is task-specific
- ex. *Word embeddings, ELMo*

Fine-tuning Approach

- Introduces minimal task-specific parameters
- The base model is a pre-trained language model
- The final model is trained by fine-tuning the pre-trained model.
- ex. *OpenAI-GPT, BERT*

How to represent words in memory ?

Trivial way is by using a *onehot* representation :

- The word is represented by a vector of length V
- All of the elements of the vector are 0 except for one element which is 1.
- V : Size of vocabulary

Example :

$$\text{onehot}_V(\text{cat}) = [0, 0, \ldots, 0, 1, 0, \ldots, 0, 0]$$

How to represent words in memory ?

Limitations with the *onehot* representation :

- Every word vector is extremely sparse.
- Every word is independent from each other.

$$\text{onehot}_V(\text{cat}) \perp \text{onehot}_V(\text{dog})$$

How to represent words in memory ?

Possible solution : **Word Embeddings**

We project the *onehot* representation of the words in a n dimension space. We **learn** this new representation.

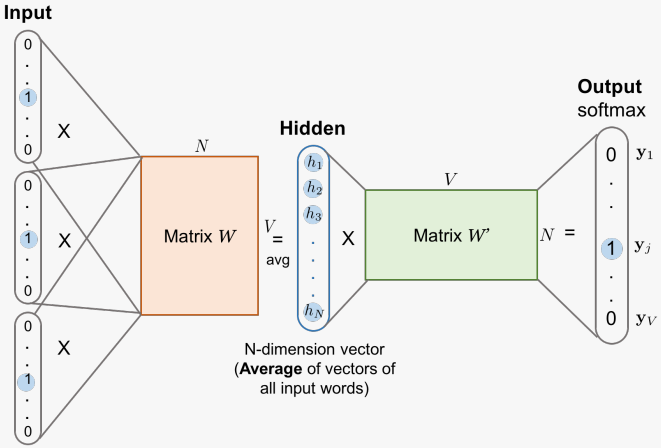
2 simple approaches (*Word2Vec* [15, 14]):

- *Continuous Bag of Words (CBOW)*
- *Skip-Gram*

Other popular approach : *GloVe* [16]

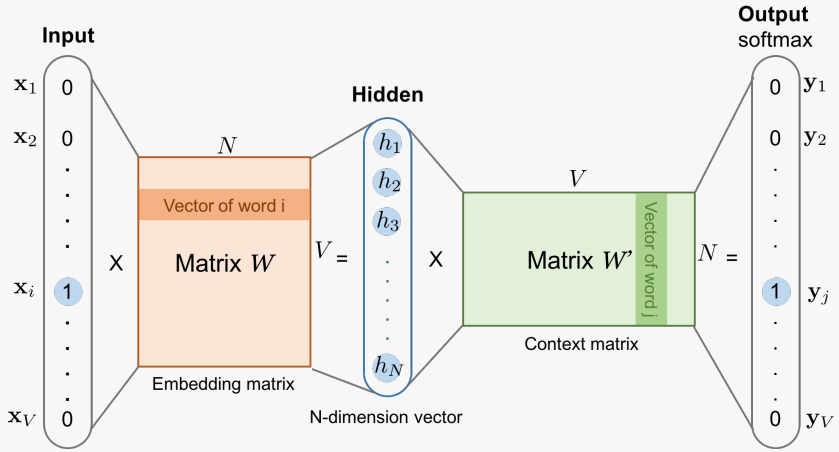
Continuous Bag of Words (CBOW)

- We use the context to predict the word.



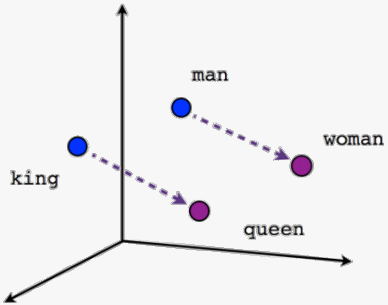
Skip-Gram

- We use the word to predict its context.

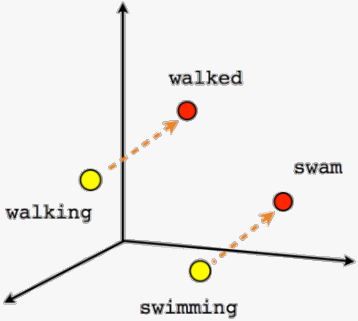


Word Embeddings

$$\text{Repr}(\text{man}) - \text{Repr}(\text{woman}) + \text{Repr}(\text{King}) \approx \text{Repr}(\text{Queen})$$

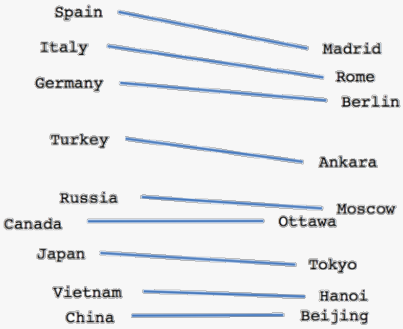


Male-Female



Verb tense

Word Embeddings



Country-Capital

Limitations with *word embeddings*

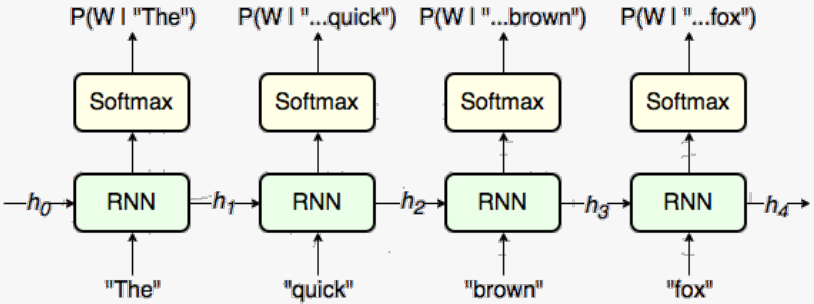
- Some words have multiple possible meaning. Ex :
  - I **play** the piano.
  - I went to a **play** yesterday.

Possible solution : **Contextualized Word Embeddings** (More on that later...)

The Task

We want to predict the next word based on a history of previous words.

$$p(\mathbf{x}_n|\mathbf{x_1}, \mathbf{x}_2, \ldots, \mathbf{x}_{n-1})$$



Exemple Language model imitating **Shakespeare** [12]

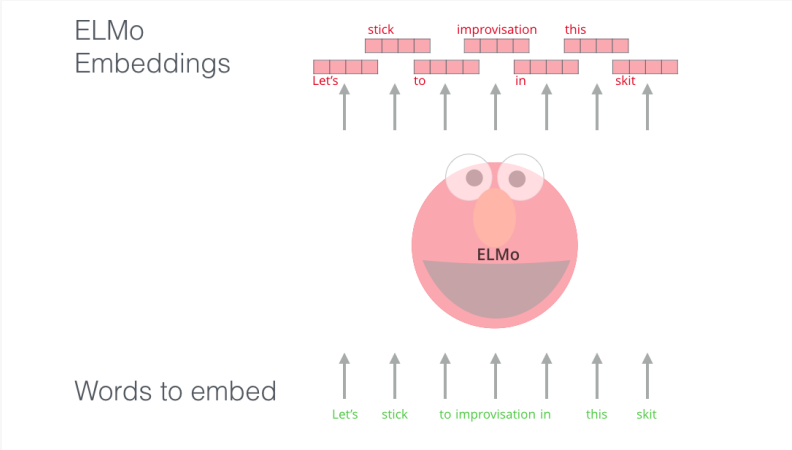
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

ELMo : Embeddings from Language Models [17]

ELMo : Embeddings from Language Models [17]



ELMo : Embeddings from Language Models [17]

- Uses Bidirectional LSTM (one language model for each direction)
- **Forward LM** :

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1})$$

- **Backward LM**

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i | x_{i+1}, \ldots, x_n)$$

ELMo : Embeddings from Language Models [17]

- Instead of just choosing the output of both direction, we learn a linear combination of the representations at each layer for a given task.

$$R_i = \{h_{i,l} | l = 0, \ldots, L\}$$

$$v_i = f(R_i, \Theta^{\text{task}}) = \gamma^{\text{task}} \sum_{l=0}^{L} s_i^{\text{task}} \mathbf{h}_{i,l}$$

ELMo : Embeddings from Language Models [17]

- **Lower layers** of ELMo are related to **syntactic tasks** (e.g. POS tagging)
- **Upper layers** of ELMo are related to more **semantic tasks** (e.g. Word Sense Disambiguation (WSG) where we try to understand the meaning of a given word based on the context)

Usage

- We use them like we would use normal word embeddings.

Limitations

- We still need to train a task-specific architecture for the various problems.

Two Approaches to Transfer Learning

Feature-based Approach

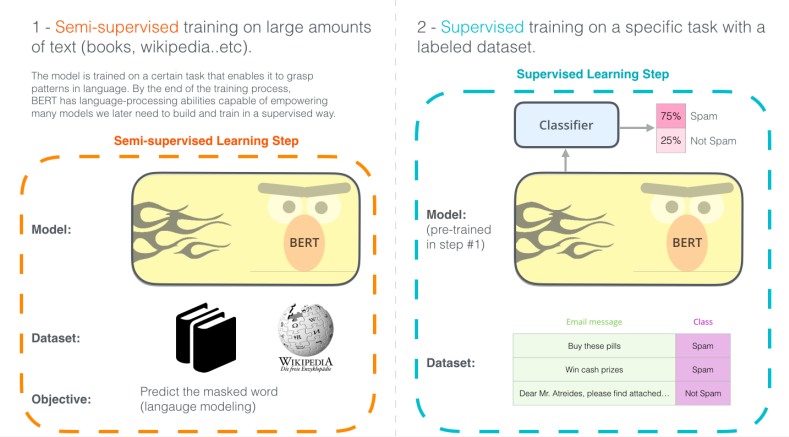
- Using pre-trained representations as features for the model
- The architecture of the model is task-specific
- ex. *Word embeddings, ELMo*

Fine-tuning Approach

- Introduces minimal task-specific parameters
- The base model is a pre-trained language model
- The final model is trained by fine-tuning the pre-trained model.
- ex. *OpenAI-GPT, BERT*

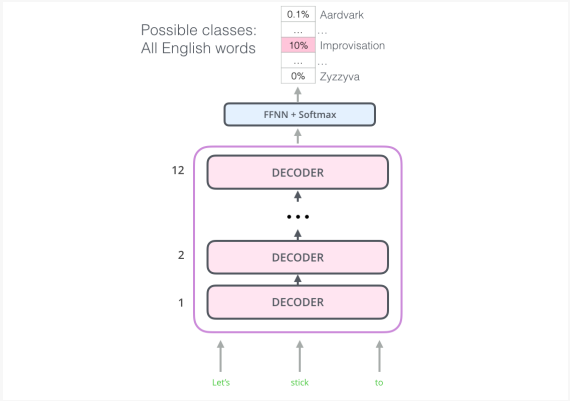
Generative Pre-training

- Similar to pretraining a computer vision model on Imagenet and learning other tasks afterwards
- Training is done in two steps :



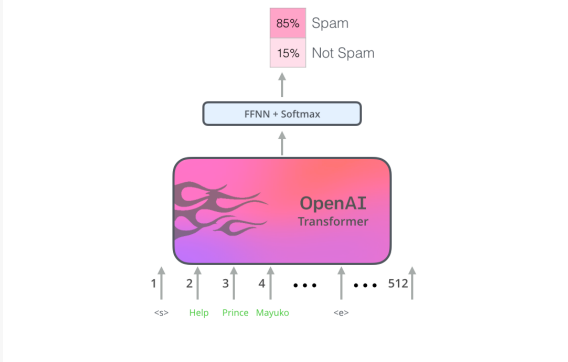
OpenAI-GPT [18]

- We train a language model using Transformer Decoders.
- We fine-tune the model on supervised tasks at the end.

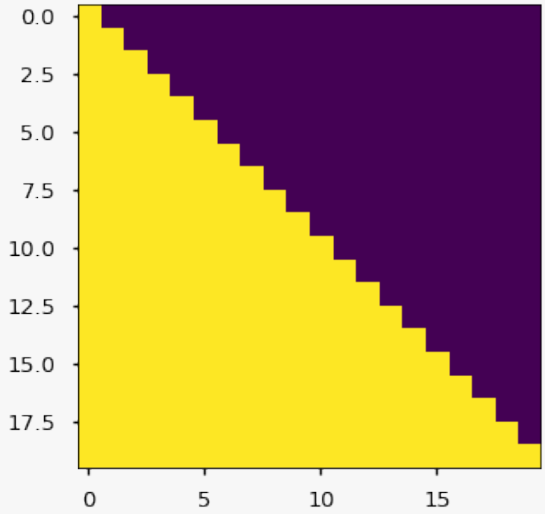


OpenAI-GPT [18]

- We train a language model using Transformer Decoders.
- We fine-tune the model on supervised tasks at the end.



OpenAI-GPT [18]



Limitations

- The language model we have is unidirectional. We want to take into account the full context to predict a word.

BERT : Bidirectional Encoder Representations from Transformers [7]

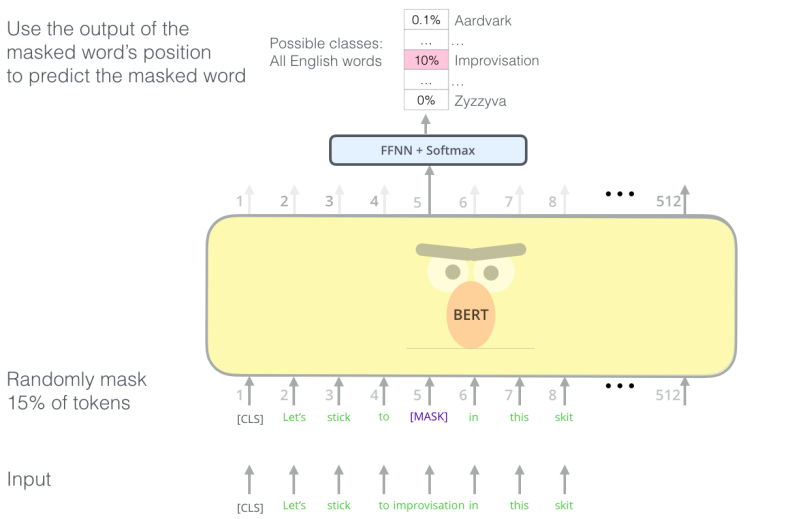
BERT : Bidirectional Encoder Representations from Transformers [7]

- We pre-train the model using Transformer Encoders on a **masked language model problem (MLM)**.

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$$

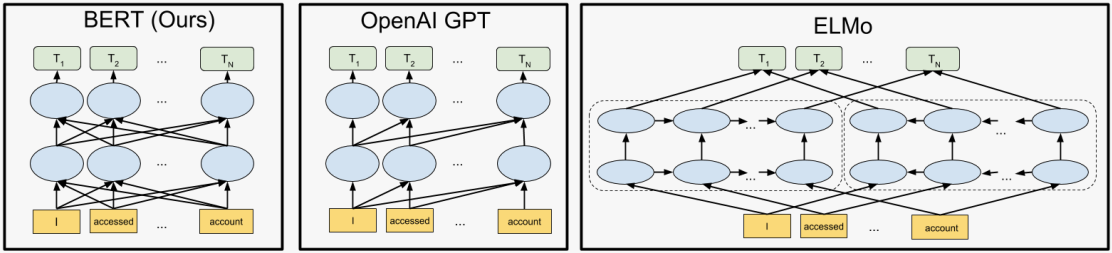
BERT : Bidirectional Encoder Representations from Transformers [7]

- We pre-train the model using Transformer Encoders on a **masked language model problem (MLM)**.



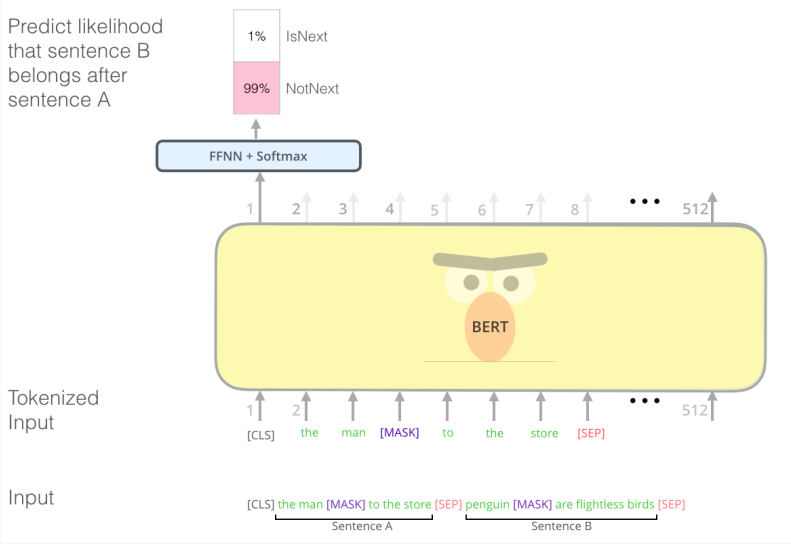
BERT : Bidirectional Encoder Representations from Transformers [7]

- Difference between BERT, OpenAI-GPT and ELMo :.



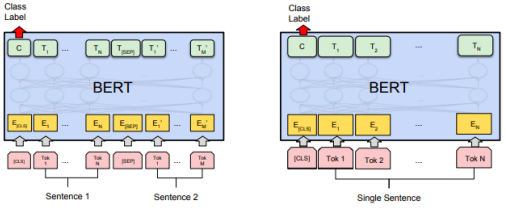
BERT : Bidirectional Encoder Representations from Transformers [7]

- For tasks taking as input multiple sentences, we want the model to be able to understand the relationship between the various sentences.
- To do so, we feed 2 sentences in the model, and ask the model to predict if the second sentence follows the first one.



BERT : Bidirectional Encoder Representations from Transformers [7]

- After the pre-training, we can train BERT on downstream tasks easily.



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

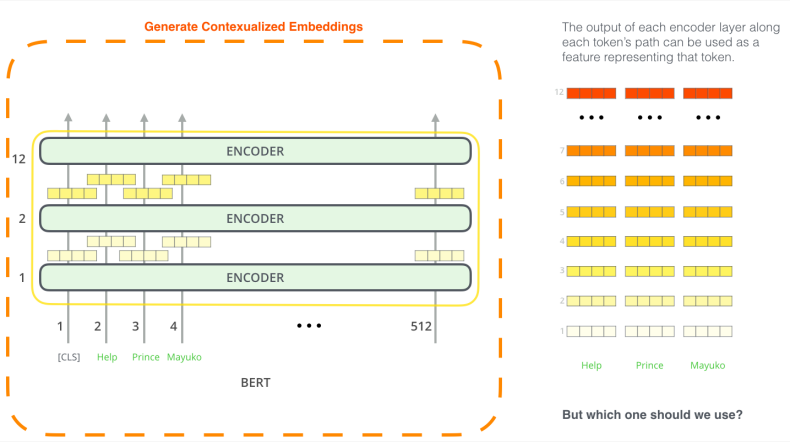
(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

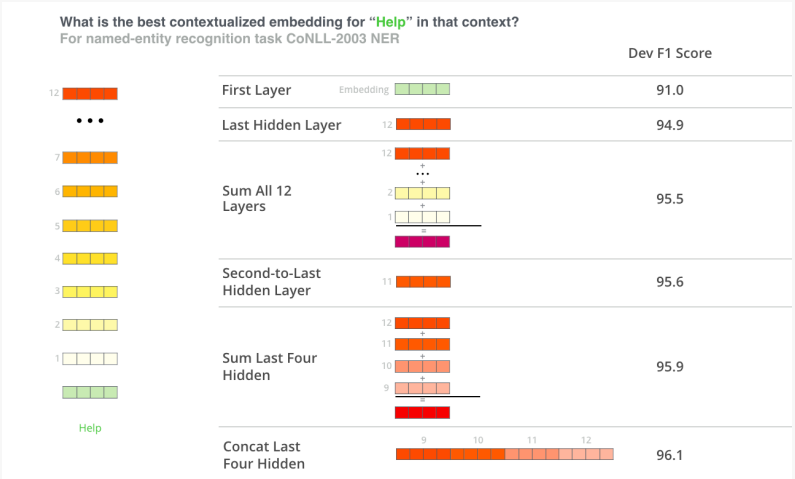
BERT : Bidirectional Encoder Representations from Transformers [7]

- BERT can also be used directly to create contextualized word embeddings just like ELMo !



BERT : Bidirectional Encoder Representations from Transformers [7]

- Which layers should you use?



BERT : Bidirectional Encoder Representations from Transformers [7]

- BERT is available for anyone to use it (Base and Large version)
- Multiple languages are available (English, Chinese, Multi-lingual version trained on Wikipedia)

Using These Recent Advances in the Data Lab

- For the language team, the benefits are evident. Using BERT let's us train supervised models that perform better with less data.
- For the UBI team, we could think of training a "driving model" in a similar way that we pretrain BERT. We could then fine-tune this model on various downstream tasks (but which ones?).

Going Further

- **OpenNMT :** Open-source implementation of the transformer architecture and multiple other models used in NMT and NLP.
- **The Annotated Transformer :** Step by step explanation of the transformer architecture.
- **The Illustrated BERT, ELMo and co. :** Step by step explanation of how to use BERT and others.
- **Generalized Language Models :** In-depth explanation of how we went from word embeddings to contextualized word vectors.
- **Multi-label Text Classification using BERT - The Mighty Transformer :** How to use the pre-trained BERT

References I

[1] J. Alammar, *The illustrated bert, elmo and co. (how nlp cracked transfer learning)*, jalammar.github.io, (2018).

[2] J. L. Ba, J. R. Kiros, and G. E. Hinton, *Layer normalization*, arXiv preprint arXiv:1607.06450, (2016).

[3] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, arXiv preprint arXiv:1409.0473, (2014).

[4] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, *A neural probabilistic language model*, Journal of machine learning research, 3 (2003), pp. 1137–1155.

References II

[5] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, *Learning phrase representations using rnn encoder-decoder for statistical machine translation*, arXiv preprint arXiv:1406.1078, (2014).

[6] Z. Dai, Z. Yang, Y. Yang, W. W. Cohen, J. Carbonell, Q. V. Le, and R. Salakhutdinov, *Transformer-xl: Attentive language models beyond a fixed-length context*, arXiv preprint arXiv:1901.02860, (2019).

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, arXiv preprint arXiv:1810.04805, (2018).

[8] J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin, *A convolutional encoder model for neural machine translation*, arXiv preprint arXiv:1611.02344, (2016).

References III

[9] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, *Convolutional sequence to sequence learning*, arXiv preprint arXiv:1705.03122, (2017).

[10] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[11] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. v. d. Oord, A. Graves, and K. Kavukcuoglu, *Neural machine translation in linear time*, arXiv preprint arXiv:1610.10099, (2016).

[12] A. Karpathy, *The unreasonable effectiveness of recurrent neural networks*, Andrej Karpathy blog, (2015).

References IV

[13] M.-T. Luong, H. Pham, and C. D. Manning, *Effective approaches to attention-based neural machine translation*, arXiv preprint arXiv:1508.04025, (2015).

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, arXiv preprint arXiv:1301.3781, (2013).

[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, *Distributed representations of words and phrases and their compositionality*, in Advances in neural information processing systems, 2013, pp. 3111–3119.

[16] J. Pennington, R. Socher, and C. Manning, *Glove: Global vectors for word representation*, in Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.

[17] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, *Deep contextualized word representations*, arXiv preprint arXiv:1802.05365, (2018).

[18] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, *Improving language understanding by generative pre-training*, URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/language-unsupervised/language_ understanding_paper. pdf, (2018).

[19] A. Rush, V. Nguyen, and G. Klein, *The annotated transformer*, harvardnlp, (2018).

[20] I. Sutskever, O. Vinyals, and Q. V. Le, *Sequence to sequence learning with neural networks*, in Advances in neural information processing systems, 2014, pp. 3104–3112.

References VI

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, *Attention is all you need*, in Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.

[22] L. Weng, *Generalized language models*, lilianweng.github.io, (2019).