Introduction to Natural Language Processing (NLP)

Explainable and Trustworthy AI

Salvatore Greco

What is Natural Language Processing (NLP)?

*Natural Language Processing (NLP) is an **interdisciplinary field** of study that combines linguistics, computer science, and artificial intelligence, aiming to enable computers to **understand, interpret, and generate human language** in a manner that is both meaningful and contextually relevant.*

Examples of NLP tasks

1. Classifying Whole Sentences
   - *Sentiment Analysis*: Determining the sentiment of a product review (positive, negative, neutral).
   - *Email Spam Detection*: Identifying whether an email is spam or not.
   - *Toxic Language Detection*: Identifying offensive or harmful language, such as hate speech, harassment, or threats.
2. Classifying Each Word in a Sentence
   - *Part-of-Speech Tagging*: Identifying the grammatical role of each word (noun, verb, adjective, etc.).
   - *Named Entity Recognition (NER)*: Identifying entities such as persons, locations, and organizations in a text.
3. Generating Text Content
   - *Text Generation*: Generating text based on a given prompt or context.
   - *Text Masking*: Filling in masked portions of text to complete sentences.
4. Extracting Answers from Text
   - *Question Answering*: Given a question and a context, extracting the answer from the text.
5. Generating New Sentences from Input Text
   - *Machine Translation*: Translating text from one language to another.
   - *Text Summarization*: Condensing a text while retaining its key points….

And many more…

How to represent the word meaning?

Meaning = *the idea that is represented by a word, phrase, or a symbol*

Linguistic definition of meaning:

- **Signifier**: a symbol (e.g., a word)

- **Signified**: the idea or thing that the symbol signifies

How to represent the word meaning?

Meaning *= the idea that is represented by a word, phrase, or a symbol*

Linguistic definition of meaning:

- **Signifier**: a symbol (e.g., a word)

Chair, Sedia, 担任主席, الكرسي

- **Signified**: the idea or thing that the symbol signifies

Representing words as discrete symbols (traditional NLP)

- **Localist representation**
  - Represent each **word** as a separate **discrete symbol**
  - Words are represented as separate **categorical features** like in statistical ML
  - Words can be represented as **one-hot vectors**
    - Encoded vector dimensionality = vocabulary size

*Like = [0 0 0 0 … **1** 0 0 0 0 0]*
*Love = [0 0 **1** 0 … 0 0 0 0 0 0]*

Representing words as discrete symbols (traditional NLP)

- **Limitations**

Representing words as discrete symbols (traditional NLP)

- **Limitations**
  1. Huge vector dimensionality
     - **Sparse** vector representation with **huge** dimensionality equal to the number of words in the vocabulary (e.g., 500k)

Representing words as discrete symbols (traditional NLP)

- **Limitations**
  1. Huge vector dimensionality
     - **Sparse** vector representation with **huge** dimensionality equal to the number of words in the vocabulary (e.g., 500k)

  2. No intrinsic/natural notion of similarity
     - All one-hot encoded vectors are **orthogonal**
     - The dot product of these vectors is always zero
     - There is **no** notion of **words' relationship and similarity**

*"I **like** that movie!"*

*"I **love** that movie!"*

Representing words by context

- **Solution**
  - Learn word **relationships** and **similarity** in the vectors' representations
  - IDEA: The **meaning of words** is given by the words that **frequently appear** close to it (distributional semantics)
    - *"You shall know a word by the company it keeps"* by John Rupert Firth
    - We can learn the **representation** of the **meaning** of a word (e.g., *"love"*) *from the set of words that appear nearby (the **context** in which the word appear)*

*"She gazed at her newborn with overwhelming **love** in her eyes."*
*"He expressed his **love** for music through his soulful melodies."*
*"Their relationship blossomed into a deep and enduring **love** over the years."*
*"Volunteering at the animal shelter brought her immense **love** and joy."*

Representing words by context

- **Objective**
  - Learn a **small** and **dense** vector for each word representing its **meaning**
    - Containing **real-values** numbers instead of 0-1, like one-hot encoding
      - **Distributed** representation instead of **localist** representation
    - The dimensionality is not the vocabulary size but a fixed-size
  - **Similar words** should have **similar vector** representations
    - Similar words are words that appear in **similar contexts**
    - Similar words have similar vector representations using similarity metrics (e.g., dot product)

Like = [0.34 -0.10 … -0.05 0.22]
Love = [0.29 -0.22 … -0.08 0.29]

  - These vectors are usually called **word embeddings**

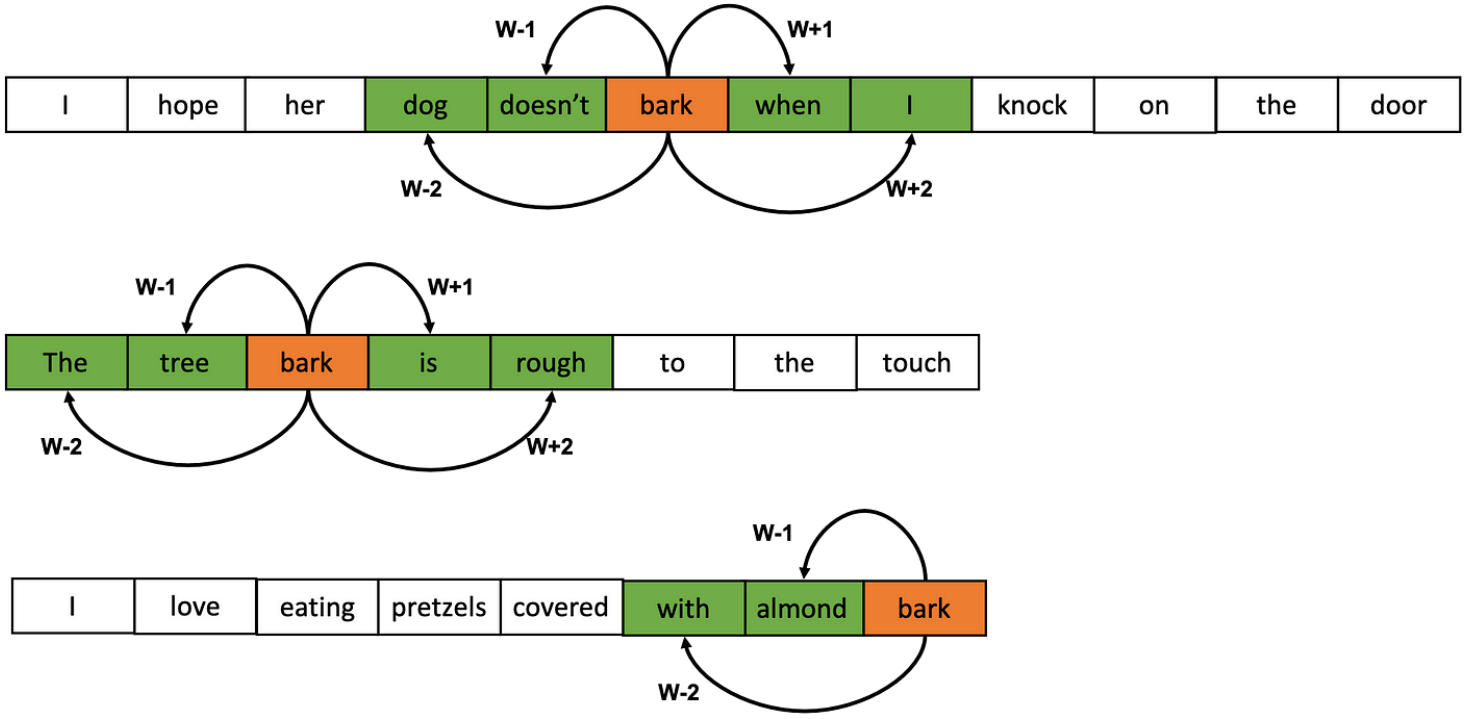
Representing words by context
Word2Vec overview

- **Word2vec [1]**
  - Starting from a large **corpus** of text
  - Select a **fixed vocabulary**, and represent each word with a vector
    - Out-of-vocabulary words are represented with a special symbol “Unknown”
  - Iterate each word in the text
    - The word will be the center word, and nearby words will be context words
  - Use **similarity measures** to compute the probability of the context words given the center word (or vice versa)
  - Compute the **gradient** on the **word representations** to adjust the word vectors to **maximize** the computed probability

[Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. https://arxiv.org/abs/1301.3781]
[Ogundepo, O. (2021, July 26). Understanding word2vec. Medium. https://medium.com/analytics-vidhya/understanding-word2vec-39fabe660705]

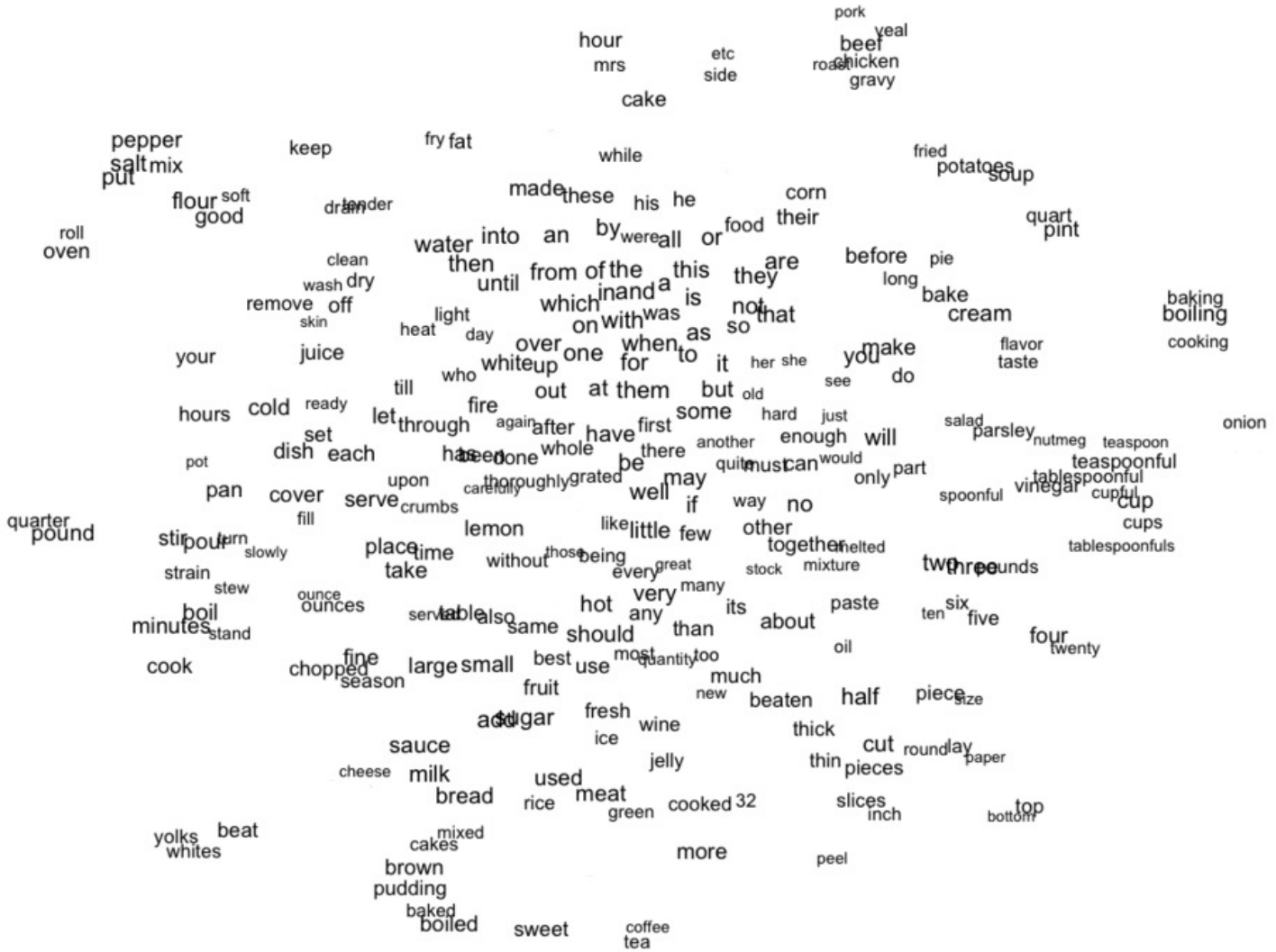
Representing words by context Word2Vec overview

- Maximize *P(dog | bark)*, *P(when | bark)*, …



[Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. https://arxiv.org/abs/1301.3781]
[Ogundepo, O. (2021, July 26). Understanding word2vec. Medium. https://medium.com/analytics-vidhya/understanding-word2vec-39fabe660705]

Representing words by context



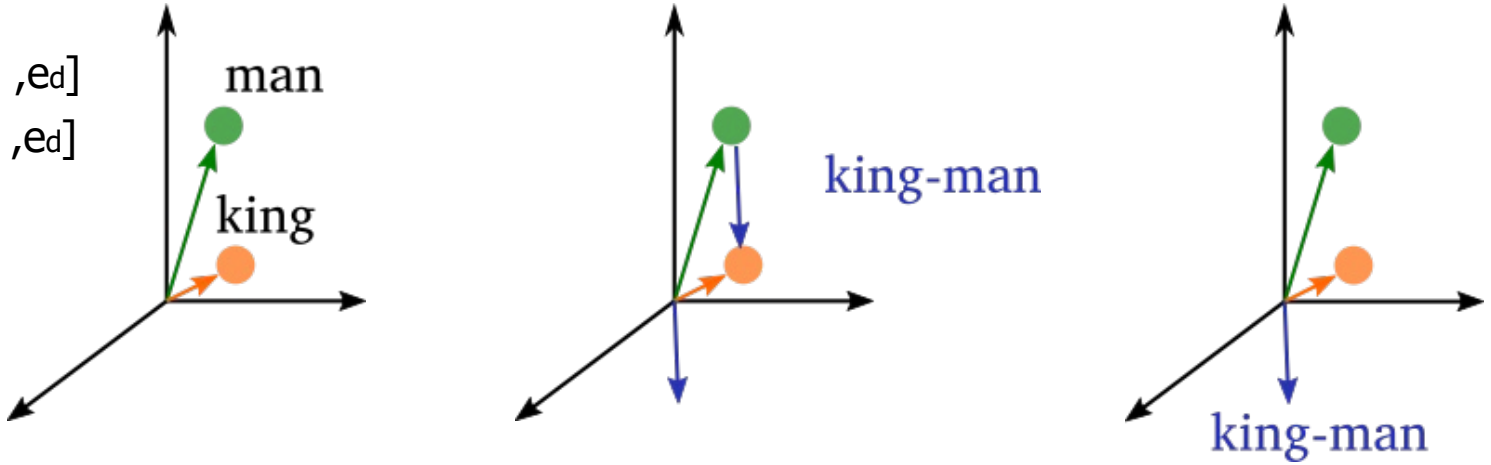
[Lopardo, A. (2019, December 23). *The basics of Language Modeling*. Medium. https://medium.com/@antonio.lopardo/the-basics-of-language-modeling-1c8832f21079]

Representing words by context

- Since each word is represented with a vector, operations among words (e.g., difference and addition) are allowed

$e(man)=[e_1,e_2, \dots ,e_d]$

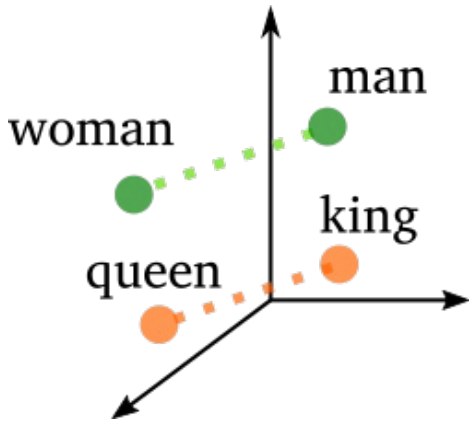
$e(king)=[e_1,e_2, \dots ,e_d]$

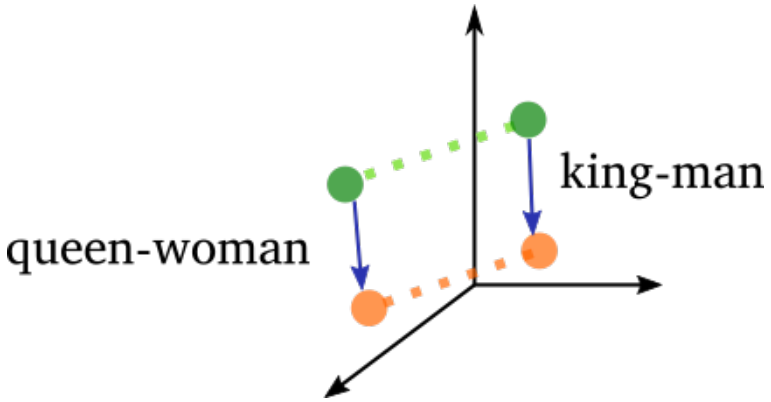


[Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. https://arxiv.org/abs/1301.3781]
[Bolukbasi, T., Chang, K. W., Zou, J. Y., Saligrama, V., & Kalai, A. T. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings. Advances in neural information processing systems, 29. https://proceedings.neurips.cc/paper_files/paper/2016/file/a486cd07e4ac3d270571622f4f316ec5-Paper.pdf]

Representing words by context

- **Semantic relationships** among words are learned and captured





king - man = queen - woman
king - man + woman = queen

[Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. https://arxiv.org/abs/1301.3781]
[Bolukbasi, T., Chang, K. W., Zou, J. Y., Saligrama, V., & Kalai, A. T. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings. Advances in neural information processing systems, 29. https://proceedings.neurips.cc/paper_files/paper/2016/file/a486cd07e4ac3d270571622f4f316ec5-Paper.pdf]

Representing words by context

- However, these representations can inherit **bias**

Man is to computer programmer as woman is to?

Man - Woman = Computer programmer - ?

homemaker

[Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. https://arxiv.org/abs/1301.3781]
[Bolukbasi, T., Chang, K. W., Zou, J. Y., Saligrama, V., & Kalai, A. T. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings. Advances in neural information processing systems, 29. https://proceedings.neurips.cc/paper_files/paper/2016/file/a486cd07e4ac3d270571622f4f316ec5-Paper.pdf]

Representing words by context

- **Takeaways**
  - We learned how to encode words in a numerical format suitable for ML
    - Dense representations (word embedding)
    - Capturing words similarity and relationship
  - Several pre-trained word embedding representations
    - Word2Vec (Mikolov et al.)
    - GloVe (Pennington et al.)
  - They are already trained on large corpora
    - You can just download the word vectors (i.e., weights or embeddings) and use them as a starting point to train neural networks for specific tasks

[Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. https://arxiv.org/abs/1301.3781]
[Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics. https://aclanthology.org/D14-1162.pdf]

Representing words by context

- **Limitations**
  - These embedding models generate **static word representations**, meaning that each word has a **fixed vector regardless of its context**
  - **Context Agnostic**
    - **T**hey assign the **same vector to a word in all its usage contexts**, ignoring the variations in meaning based on sentence context
  - **Polysemy Limitation**
    - Words with **multiple meanings** (polysemous words) are **represented by a single embedding**, which can be misleading in complex semantic tasks
  - **Example:**

"He sat by the river **bank**."

"She deposited money in the **bank**."

How can we use word vectors in NN?

- **Language Modeling (LM) Task**
  - Language modeling is the task consisting of **predicting the next word** given a sequence of words
  - Formally, given a sequence of words $x^{(1)}$, $x^{(2)}$, … $x^{(t)}$, compute the probability of the next word $x^{(t+1)}$

$$P(\boldsymbol{x}^{(t+1)} | \ \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$$

I love this ____

[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Lopardo, A. (2019, December 23). The basics of Language Modeling. Medium. https://medium.com/@antonio.lopardo/the-basics-of-language-modeling-1c8832f21079]

Can we use word vectors and MLP for LM?

- We can use a window-based neural network
  - Select a fixed-window size (e.g., 5 words)
  - Discard the remaining words

output distribution

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{U}\boldsymbol{h} + \boldsymbol{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

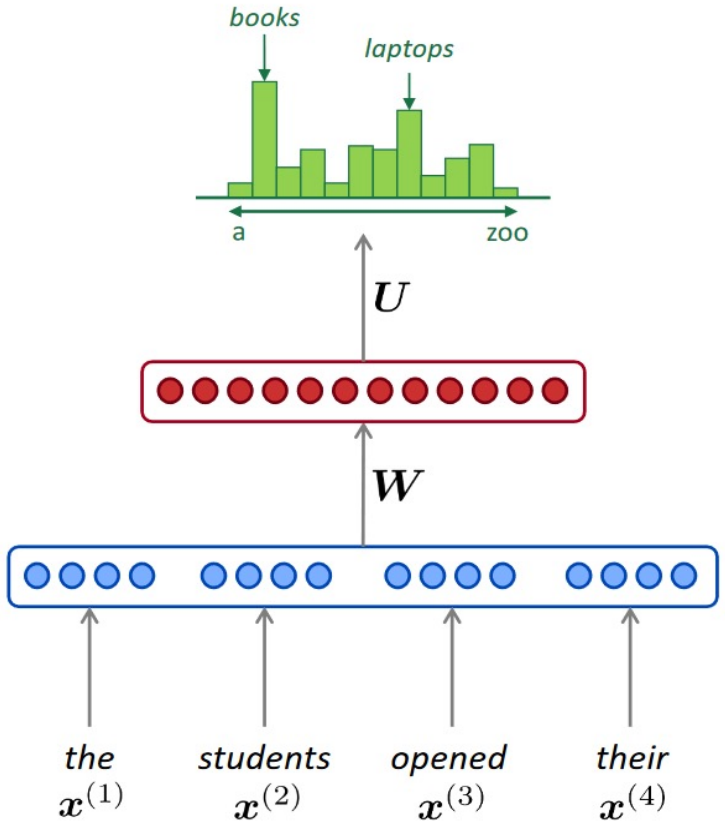
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

concatenated word embeddings

$$\boldsymbol{e} = [\boldsymbol{e}^{(1)}; \boldsymbol{e}^{(2)}; \boldsymbol{e}^{(3)}; \boldsymbol{e}^{(4)}]$$

words / one-hot vectors

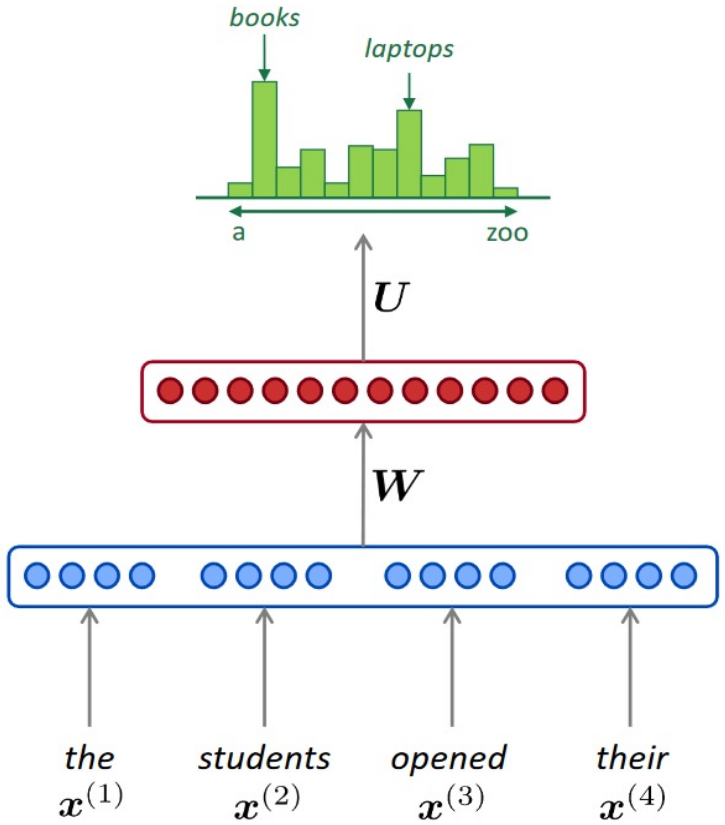
$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$$



[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Lopardo, A. (2019, December 23). The basics of Language Modeling. Medium. https://medium.com/@antonio.lopardo/the-basics-of-language-modeling-1c8832f21079]

Can we use word vectors and MLP for LM?

- We can use a window-based neural network
  - Select a fixed-window size (e.g., 5 words)
  - Discard the remaining words

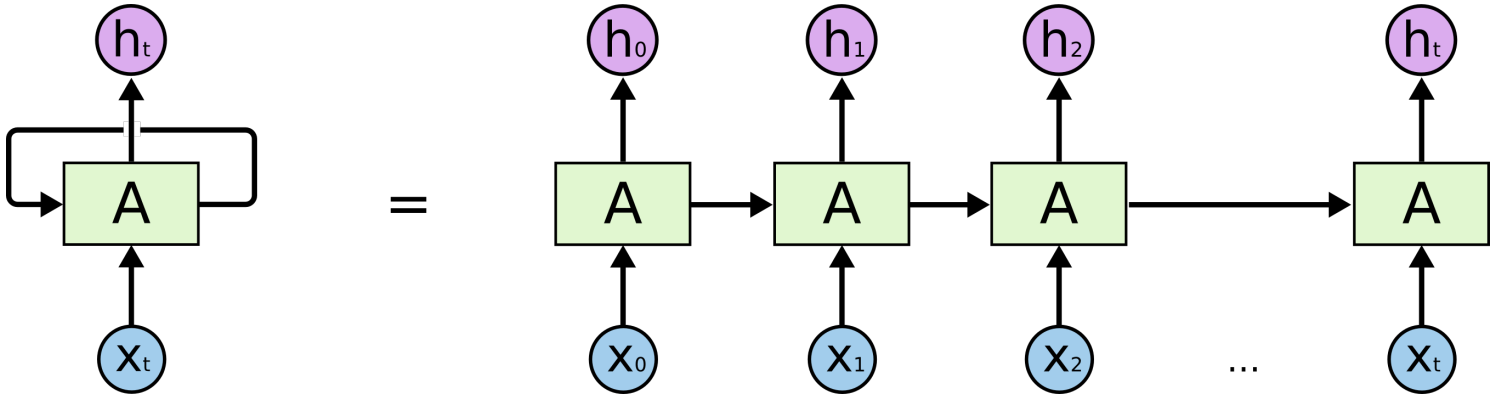


- **Limitations**
  1. Texts can have an **arbitrary size**, but this architecture only uses the last words depending on the context window size
  2. Increasing the window size, **increases the size of the weight (*W*)**
     W size = *window size* x *embedding dimensionality*
  3. Each word is multiplied by **completely different weights** based on the position
     - There is no symmetry in how input words are processed

[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Lopardo, A. (2019, December 23). The basics of Language Modeling. Medium. https://medium.com/@antonio.lopardo/the-basics-of-language-modeling-1c8832f21079]

Recurrent Neural Networks RNNs

- **Idea:** apply the **same weights** *W* repeatedly over time (for each word)
- **Recurrent:** we have and hidden layer (hidden state) what we compute and maintain over time, and we feed it back into the layer itself
  - Forward and backward propagation over time



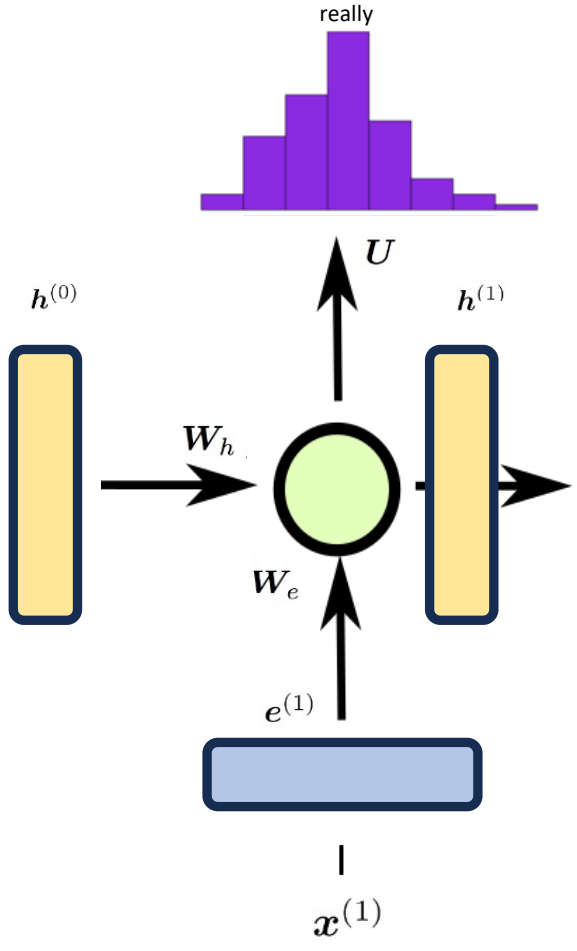
[Olah, C. *Understanding LSTM networks*. Understanding LSTM Networks -- colah's blog. https://colah.github.io/posts/2015-08-Understanding-LSTMs]

Recurrent Neural Networks RNNs (LM)

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

$\boldsymbol{h}^{(0)}$ is the initial hidden state



$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

words / one-hot vectors
$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$

[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Lopardo, A. (2019, December 23). The basics of Language Modeling. Medium. https://medium.com/@antonio.lopardo/the-basics-of-language-modeling-1c8832f21079]

Recurrent Neural Networks RNNs (LM)

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

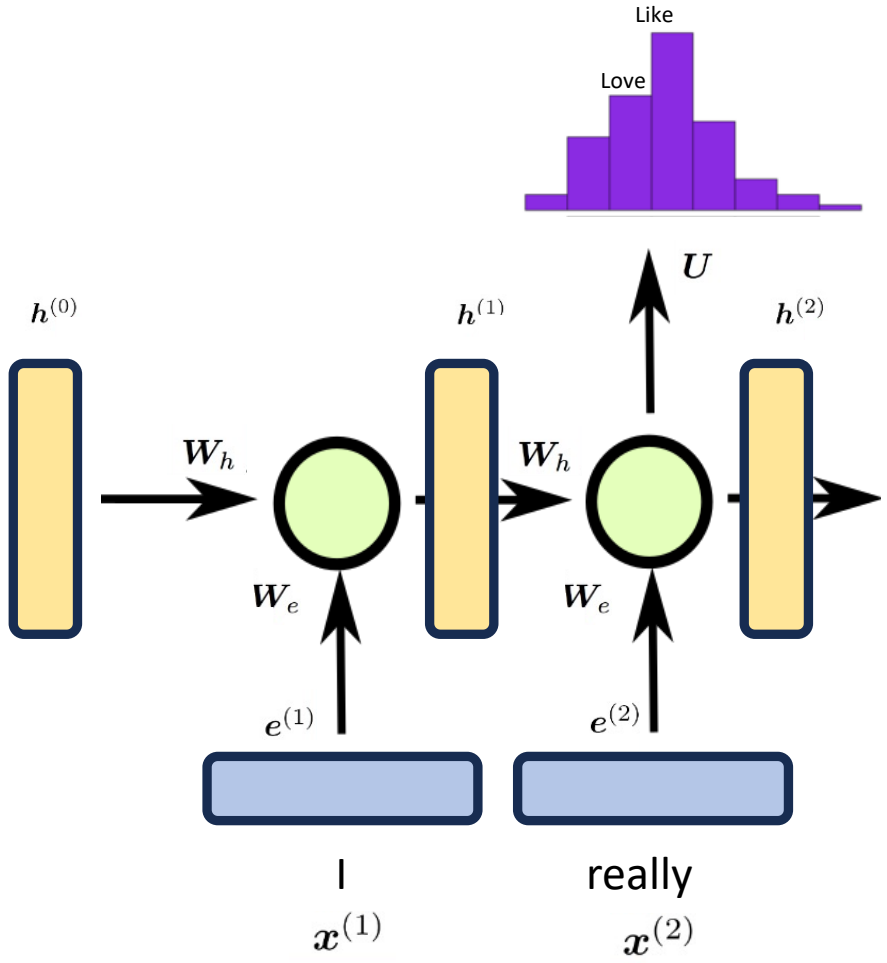
$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

$\boldsymbol{h}^{(0)}$ is the initial hidden state

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

words / one-hot vectors

$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$



[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Lopardo, A. (2019, December 23). The basics of Language Modeling. Medium. https://medium.com/@antonio.lopardo/the-basics-of-language-modeling-1c8832f21079]

Recurrent Neural Networks RNNs (LM)

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

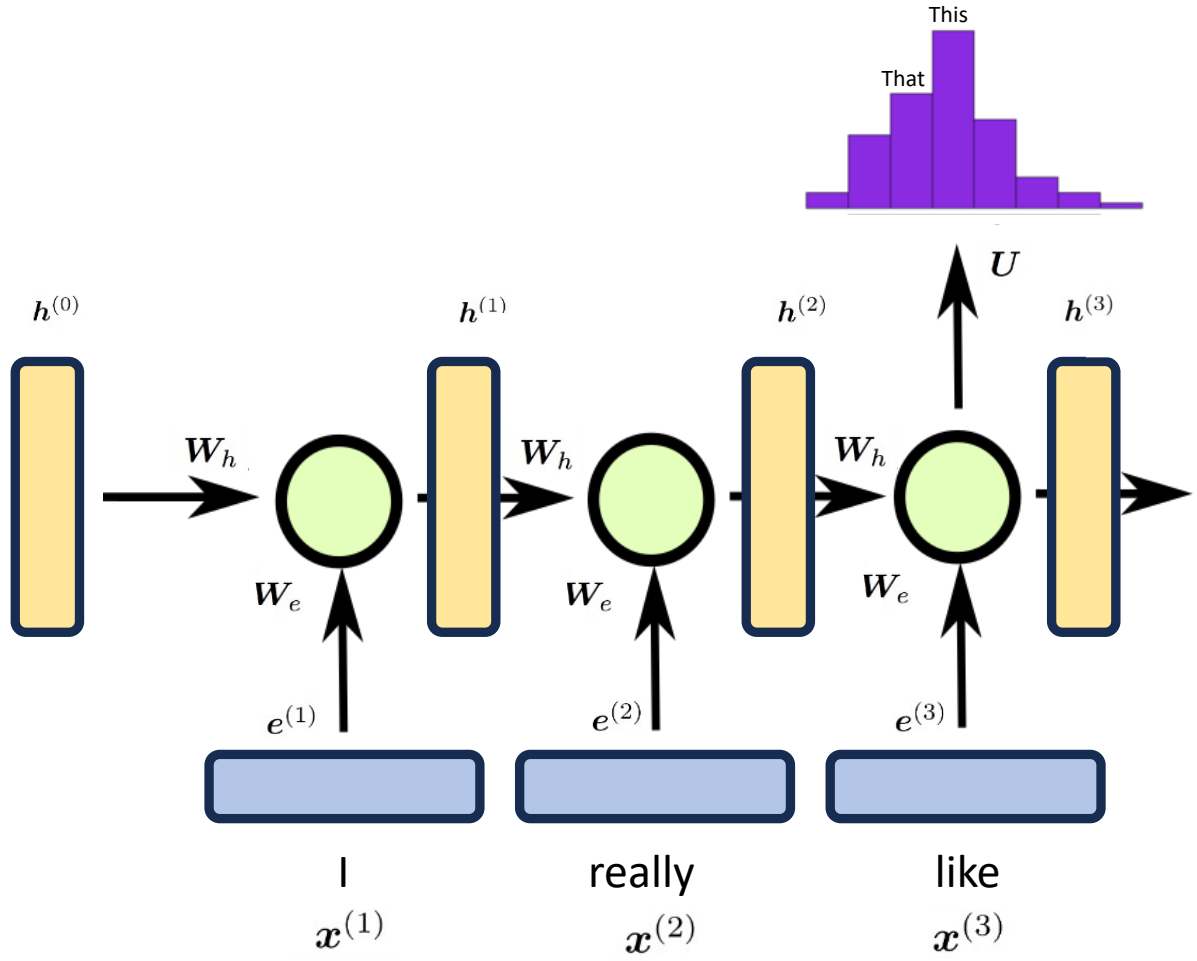
$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

$\boldsymbol{h}^{(0)}$ is the initial hidden state

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

words / one-hot vectors

$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$



[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Lopardo, A. (2019, December 23). The basics of Language Modeling. Medium. https://medium.com/@antonio.lopardo/the-basics-of-language-modeling-1c8832f21079]

Recurrent Neural Networks RNNs (LM)

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

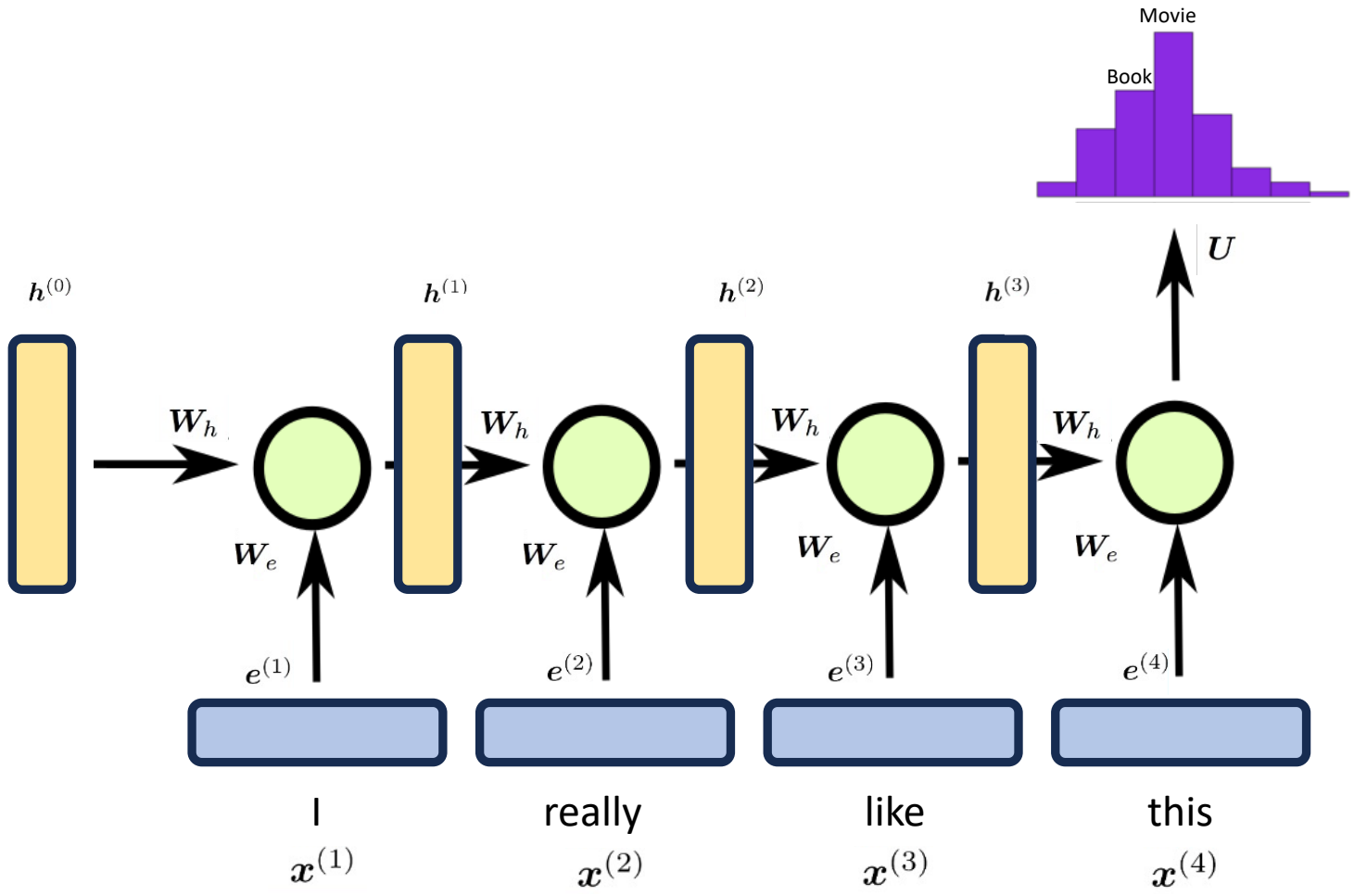
$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

$\boldsymbol{h}^{(0)}$ is the initial hidden state

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

words / one-hot vectors

$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$



[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Lopardo, A. (2019, December 23). The basics of Language Modeling. Medium. https://medium.com/@antonio.lopardo/the-basics-of-language-modeling-1c8832f21079]

Recurrent Neural Networks RNNs (LM)

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

Decoding

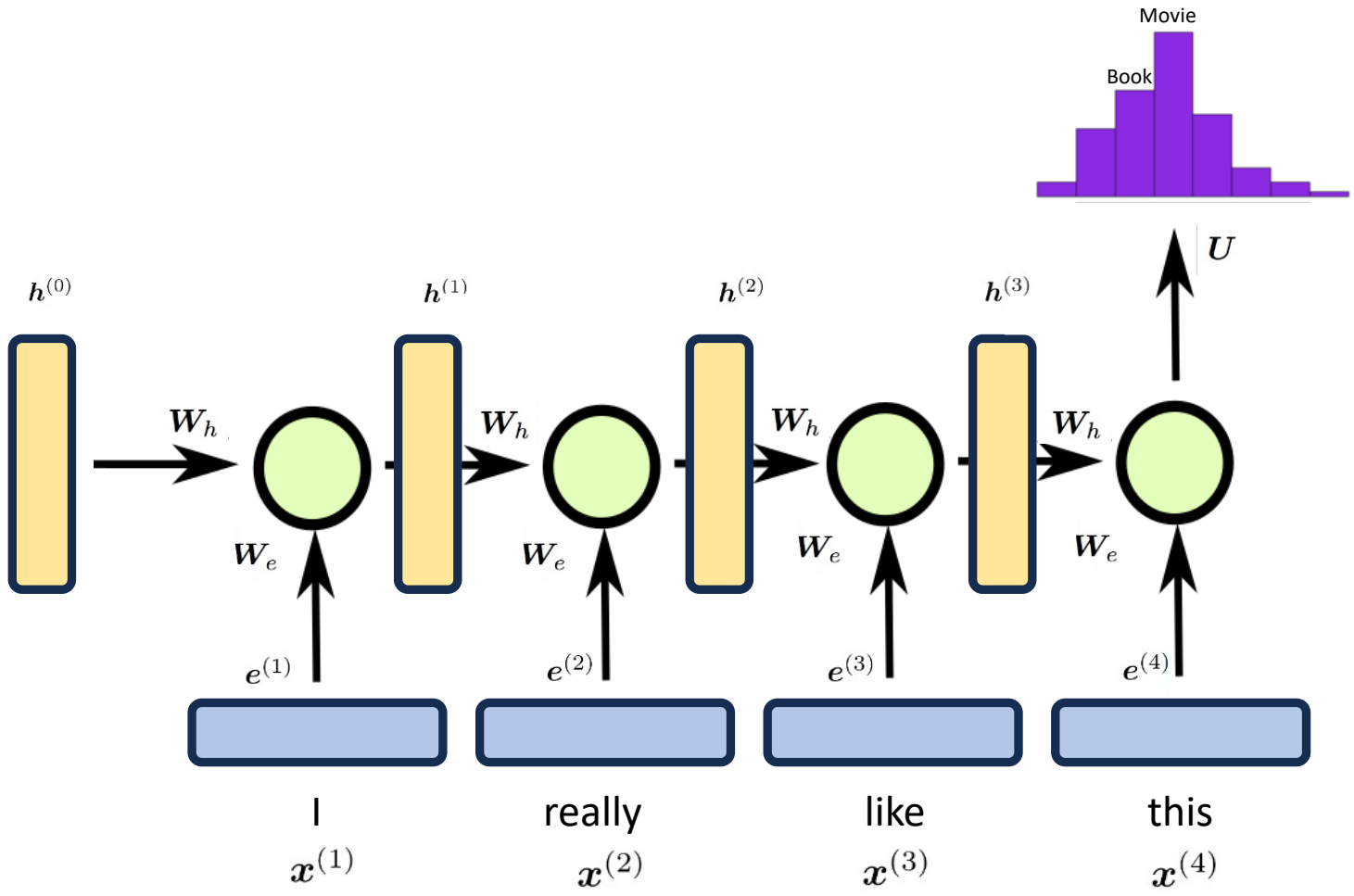
$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

$\boldsymbol{h}^{(0)}$ is the initial hidden state

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

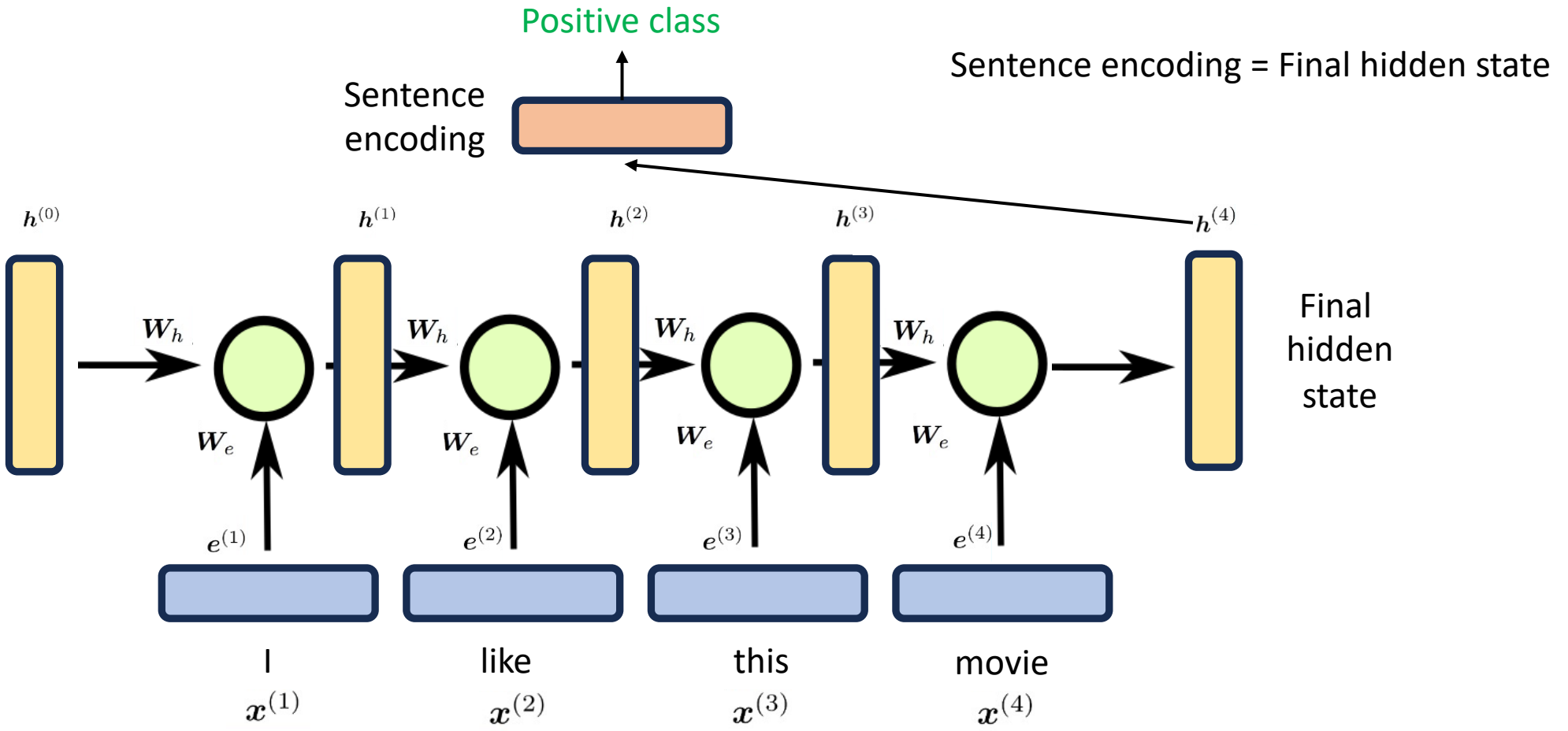
words / one-hot vectors

$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$



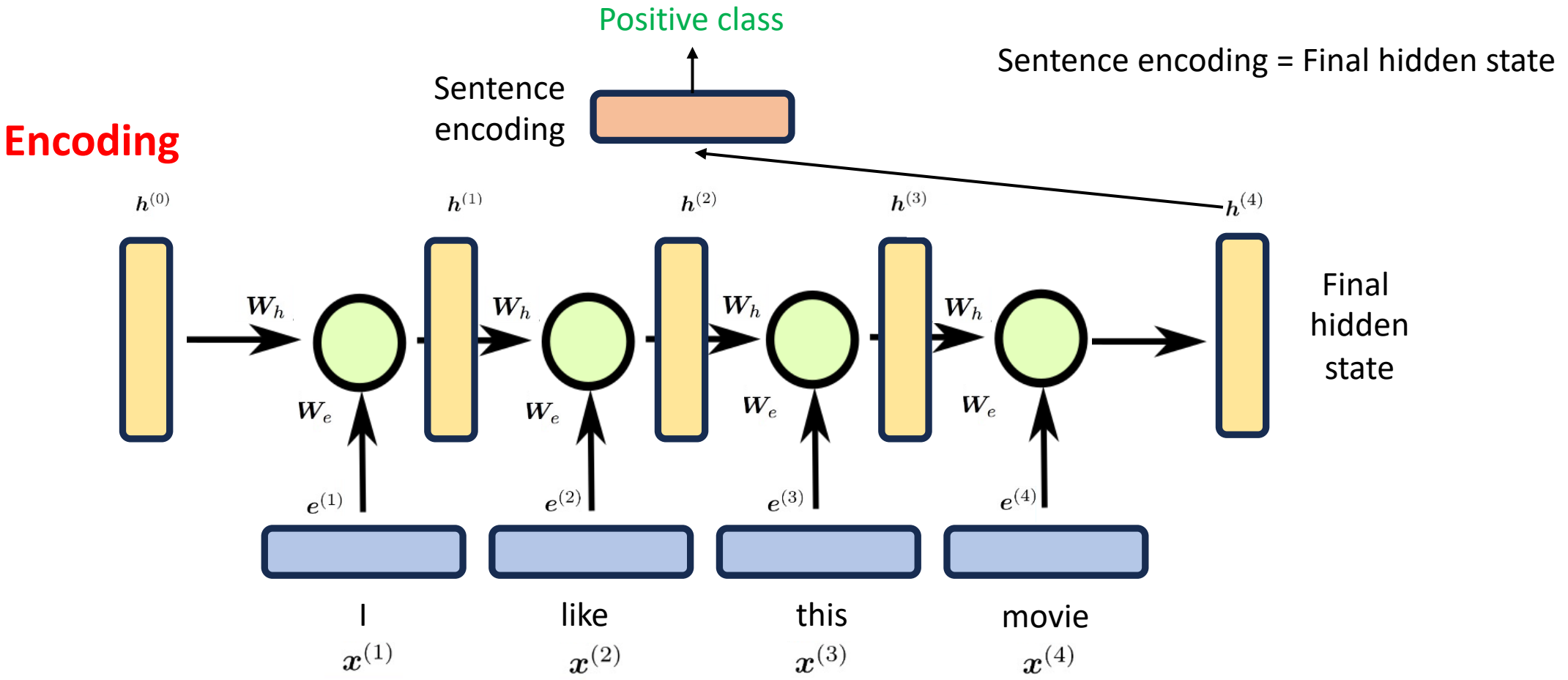
[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Lopardo, A. (2019, December 23). The basics of Language Modeling. Medium. https://medium.com/@antonio.lopardo/the-basics-of-language-modeling-1c8832f21079]

Recurrent Neural Networks RNNs
How can we do classification? (1)



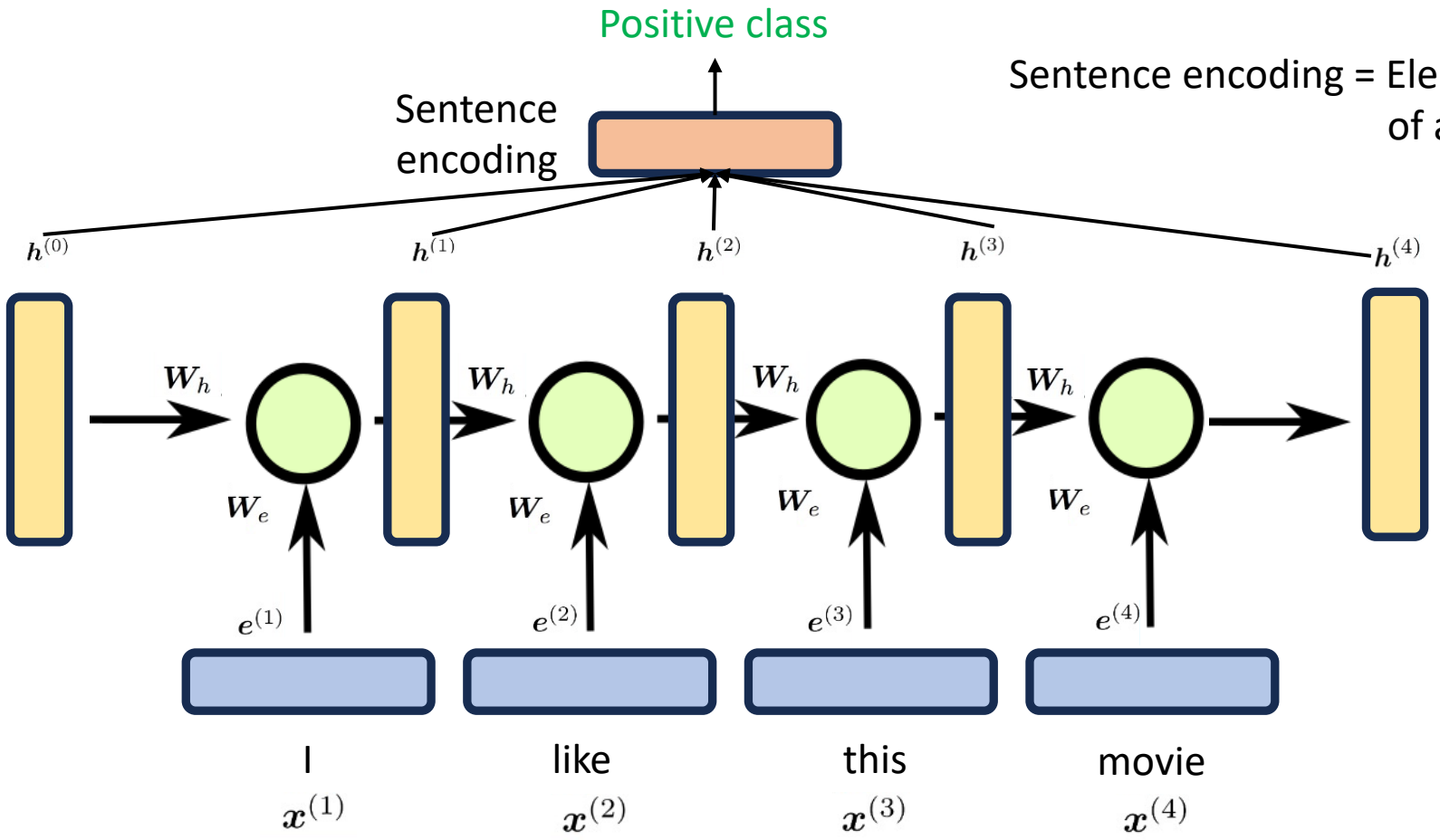
[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Lopardo, A. (2019, December 23). The basics of Language Modeling. Medium. https://medium.com/@antonio.lopardo/the-basics-of-language-modeling-1c8832f21079]

Recurrent Neural Networks RNNs
How can we do classification? (1)



[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Lopardo, A. (2019, December 23). The basics of Language Modeling. Medium. https://medium.com/@antonio.lopardo/the-basics-of-language-modeling-1c8832f21079]

Recurrent Neural Networks RNNs
How can we do classification? (2)



Sentence encoding = Element-wise mean or max of all hidden states

[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]

[Lopardo, A. (2019, December 23). The basics of Language Modeling. Medium. https://medium.com/@antonio.lopardo/the-basics-of-language-modeling-1c8832f21079]

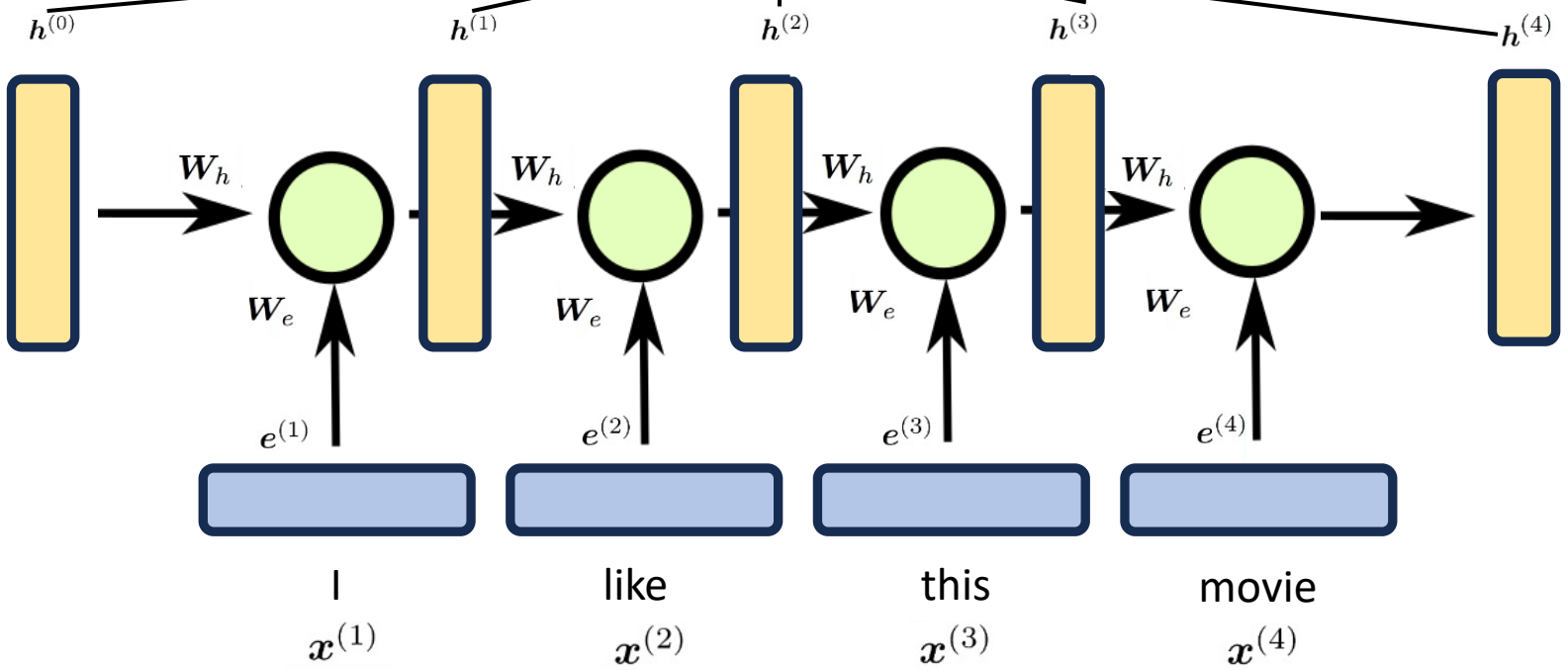
Recurrent Neural Networks RNNs
How can we do classification? (2)

Positive class

Sentence encoding

Sentence encoding = Element-wise mean or max of all hidden states

Usually Better!



[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]

[Lopardo, A. (2019, December 23). The basics of Language Modeling. Medium. https://medium.com/@antonio.lopardo/the-basics-of-language-modeling-1c8832f21079]

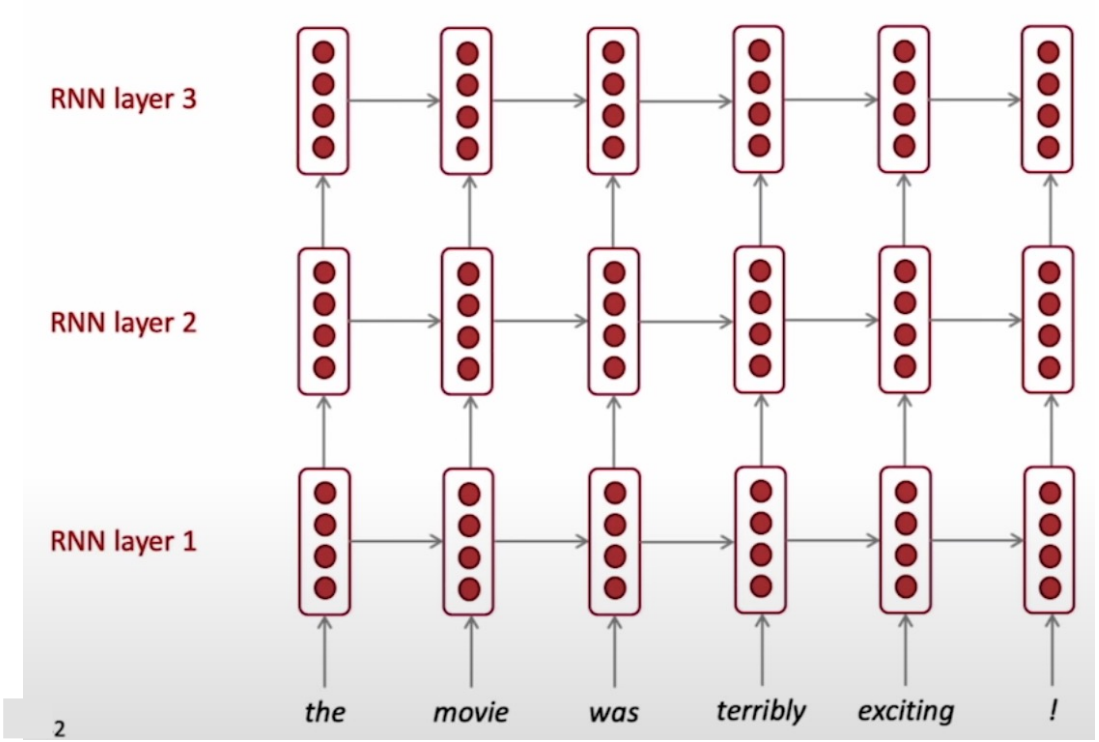
Recurrent Neural Networks RNNs

- **Still two possible improvements**
  1. Deep learning usually exploits **“deep” representations** by stacking multiple layers
     - RNNs are deep in the time dimension but not in the hidden dimension
  2. To create the hidden states, for each word only the **left context** (previous words are used)
     - Useful information can be present in “future” words
     - This applies only to tasks where you have access to the entire sentence
       - Classification
       - Translations
       - …
       - But not Language Modeling!

[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]

Recurrent Neural Networks RNNs
Multi-layer RNNs

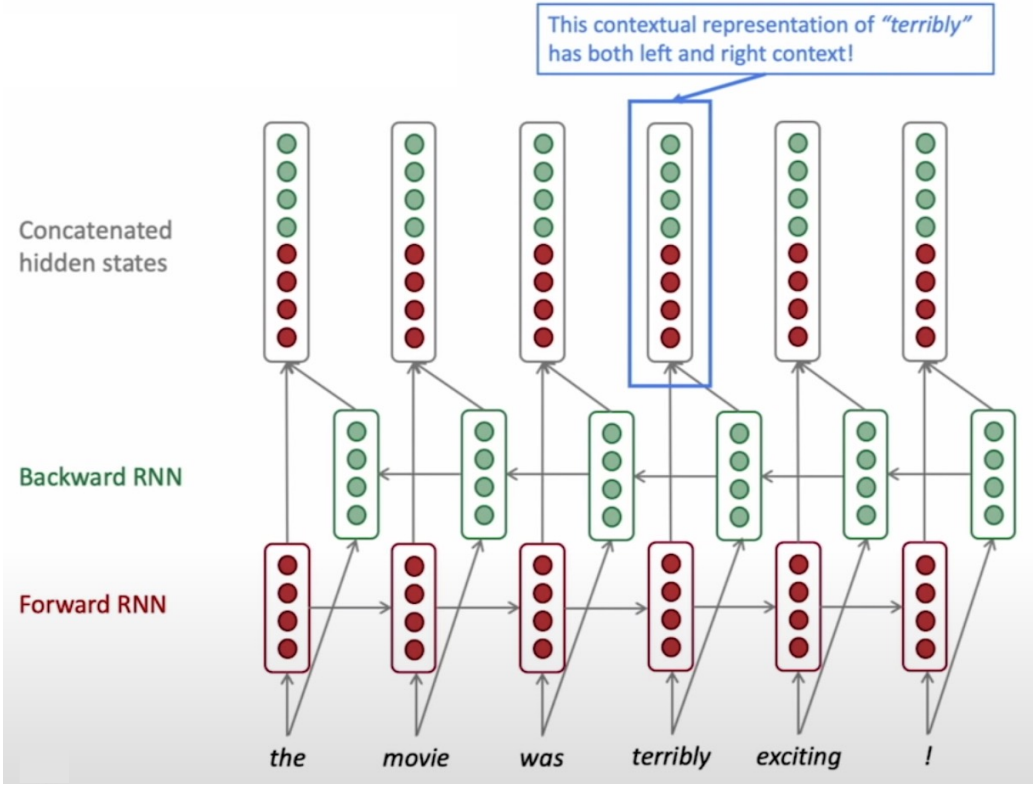
- We can **stack** multiple RNNs to create *"deeper"* representations



[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Lopardo, A. (2019, December 23). The basics of Language Modeling. Medium. https://medium.com/@antonio.lopardo/the-basics-of-language-modeling-1c8832f21079]

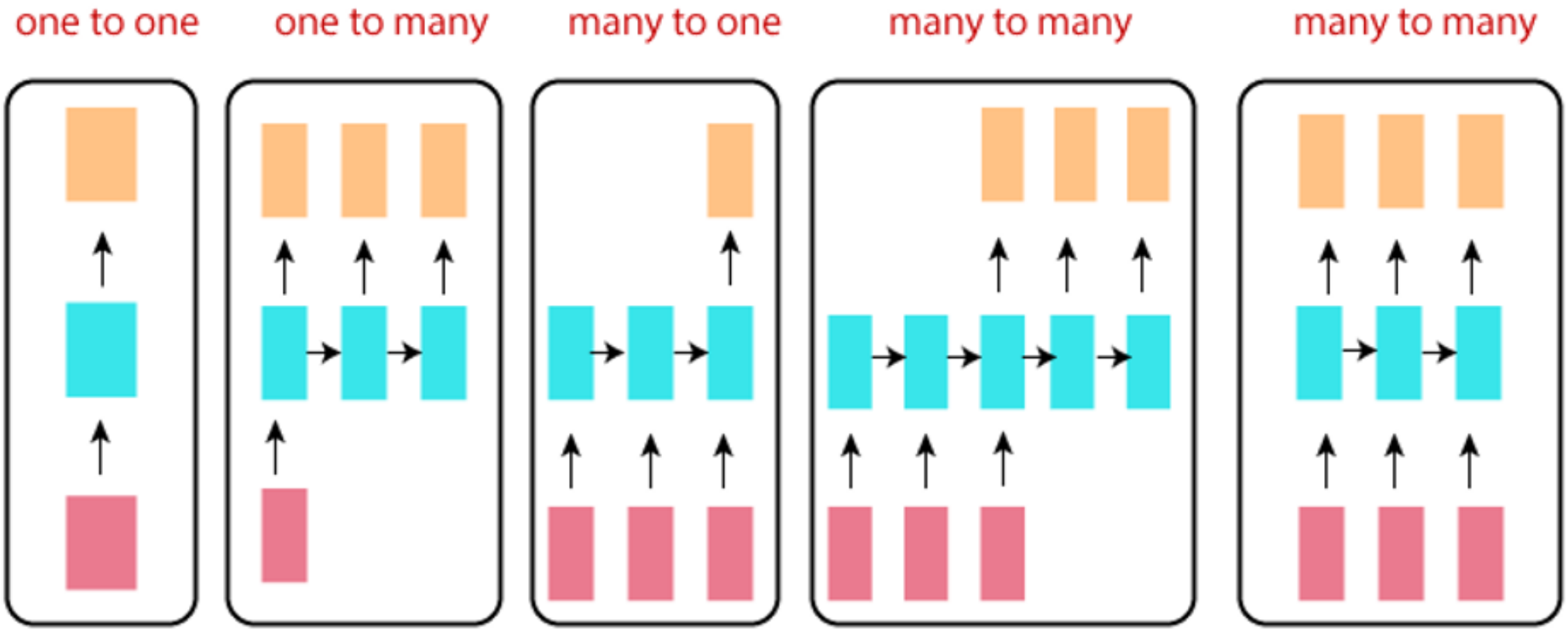
Recurrent Neural Networks RNNs Bidirectional RNNs

- We can provide both **left and right context**
  - Only applicable if you can access the entire input sequence (e.g., encoding for classification or translations, but not for language modeling)



[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Lopardo, A. (2019, December 23). The basics of Language Modeling. Medium. https://medium.com/@antonio.lopardo/the-basics-of-language-modeling-1c8832f21079]

Recurrent Neural Networks RNNs
Possible architectures



[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[https://dotnettutorials.net/lesson/recurrent-neural-network/]

Recurrent Neural Networks RNNs

- **Advantages**
  - Can process input texts of **any length** (i.e., number of words)
  - The **same weights** *W* are applied to each word (i.e., timestamp)
  - At each timestamp, the model can **use information of previous and future** (in bidirectional RNNs**) timestamps**
  - **Fixed model size**
    - Does not increase with the size of the input text (or the context window)

[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]

Recurrent Neural Networks RNNs

- **Limitations**
  1. **Slow** forward and backward propagation
     - They should be performed **sequentially**
     - Cannot be parallelized
     - Difficult to train on huge corpora

  2. Fail to learn **long-distance dependencies**
     - **Vanishing** and **exploding gradients**
     - RNNs take *O(sequence length)* steps for distant words to affect each other (interaction distance between words)

[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]

Attention

Attention uses each word's representation as a **query** to access and incorporate information from a **set of values** to create new representations

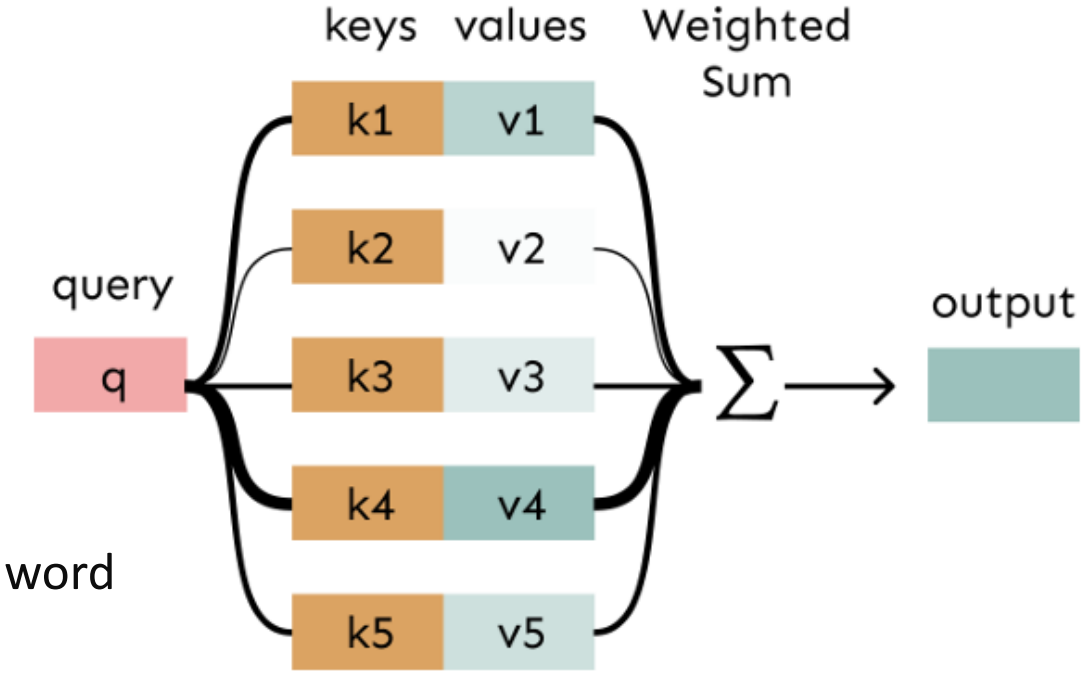
Advantages:

- **Better parallelization** (unparallelizable computations does not increase with sequence length)
- **Faster** forward and backward propagation
- **Interaction distance** *O(1)* since all words "attend" each other at every layer
  - Don't need "recurrency" to interact

[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). https://papers.nips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf]
[Fontana, U. (2023, October 10). *NLP part 7 - self-attention and Transformers*. Medium. https://medium.com/@umbertofontana/nlp-part-7-self-attention-and-transformers-c770dea1283a]

Self-attention

- The **attention** mechanism functions like a fuzzy, non-discrete **lookup in a key-value** store
- The match is more **soft**
  - Weights are not binary, but scores between 0 and 1
- The output value will be a **weighted sum** of all the possible values
- This creates a **contextualized representation** of each word
- It is called **self-attention** because each word can attend each word in the **same input sequence** (both past and future)



[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Jay Alammar, The Illustrated Transformer. https://jalammar.github.io/illustrated-transformer]
[Fontana, U. (2023, October 10). *NLP part 7 - self-attention and Transformers*. Medium. https://medium.com/@umbertofontana/nlp-part-7-self-attention-and-transformers-c770dea1283a]

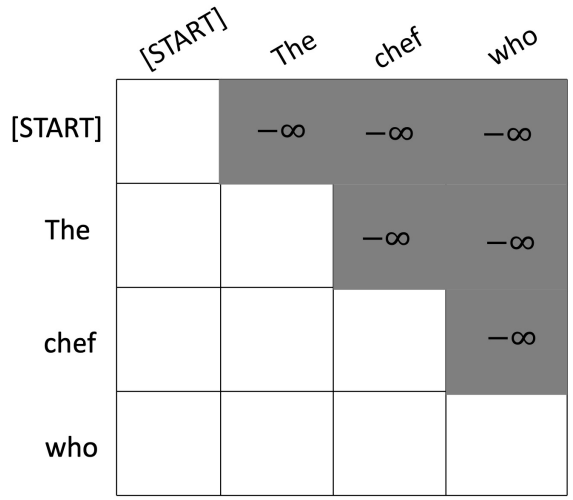
Self-attention
Problem 1: Lack of inherent notion of order

- Self-attention does **not have an inherent notion of order**
- We need to **encode the order of the sentence**
- **Solution**: We can learn an **embedding of the position** and **add** it to the non-contextualized embedding representation of each word
  - Usually we learn **absolute position representations**
  - This is usually done by just **adding** to the embedding vector the position embedding at the **first layer**

[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Jay Alammar, The Illustrated Transformer. https://jalammar.github.io/illustrated-transformer]
[Fontana, U. (2023, October 10). *NLP part 7 - self-attention and Transformers*. Medium. https://medium.com/@umbertofontana/nlp-part-7-self-attention-and-transformers-c770dea1283a]

Masking future words

- To use self-attention for decoders you need to ensure it does not use future words
- You can **mask out attention** to future words by setting attention scores to $-\infty$

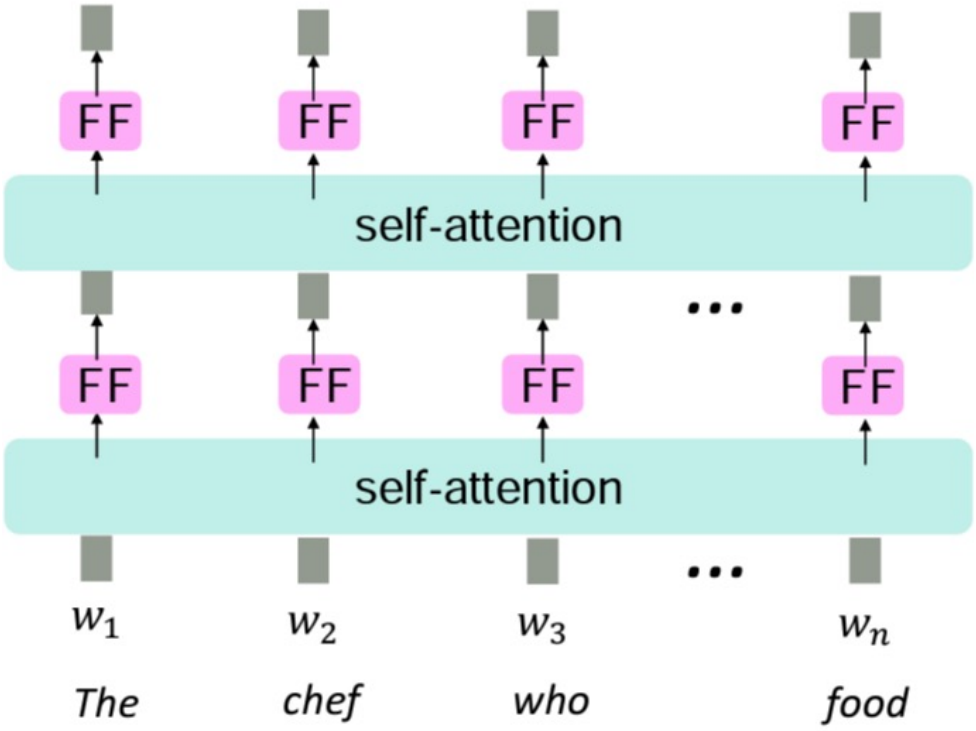


[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Jay Alammar, The Illustrated Transformer. https://jalammar.github.io/illustrated-transformer]
[Fontana, U. (2023, October 10). *NLP part 7 - self-attention and Transformers*. Medium. https://medium.com/@umbertofontana/nlp-part-7-self-attention-and-transformers-c770dea1283a]

Self-attention
Problem 2: Lack of non-linearities

- There are **no nonlinearities** in self-attention
- Stacking more self-attention layers just re-averages the value vectors
  - Like stacking more linear layers in MLP without adding non-linear functions

- **Solution**: add **feed-forward networks** to post-process each output value vector



[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Jay Alammar, The Illustrated Transformer. https://jalammar.github.io/illustrated-transformer]
[Fontana, U. (2023, October 10). *NLP part 7 - self-attention and Transformers*. Medium. https://medium.com/@umbertofontana/nlp-part-7-self-attention-and-transformers-c770dea1283a]

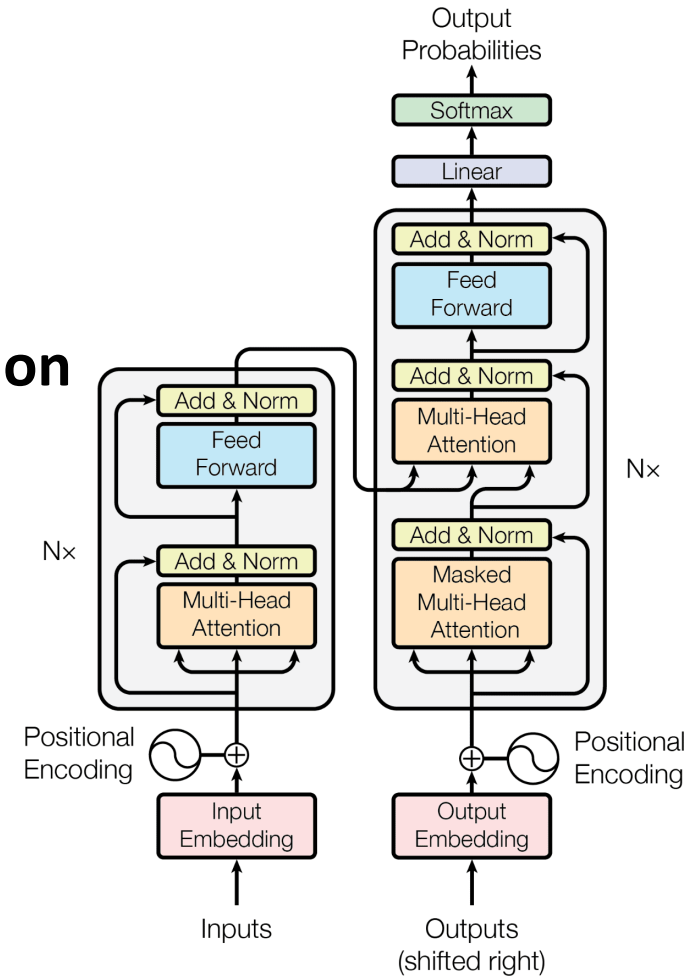
Self-attention
Problem 3: Look at the future

- Self-attention ”look” at all the **words within the same sentence**
- It can **“look at the future”** by attending to the next words
- This is **not a problem** when you want to encode a full sentence for tasks such as **classification**
- This **is a problem** for tasks such as **language modeling** or **machine translation**
  - The model can “cheat” by looking at the future words to predict the next word

[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Jay Alammar, The Illustrated Transformer. https://jalammar.github.io/illustrated-transformer]
[Fontana, U. (2023, October 10). *NLP part 7 - self-attention and Transformers*. Medium. https://medium.com/@umbertofontana/nlp-part-7-self-attention-and-transformers-c770dea1283a]

Transformers

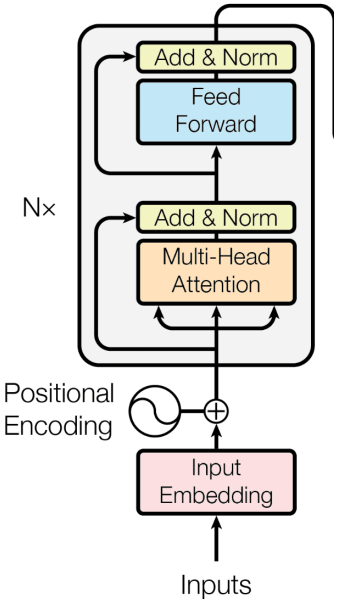
- The transformer model has an **encoder** and a **decoder** and is based on multiple stacked attention blocks
- Instead of using a single attention it uses **a multi-head attention**
  - Multiple attention head mechanisms
  - Each attention head can learn different aspects
- The main difference between the encoder and the decoder is that the **encoder uses self-attention** while the **decoder masked attention**



[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Jay Alammar, The Illustrated Transformer. https://jalammar.github.io/illustrated-transformer]
[Fontana, U. (2023, October 10). *NLP part 7 - self-attention and Transformers*. Medium. https://medium.com/@umbertofontana/nlp-part-7-self-attention-and-transformers-c770dea1283a]

Transformers - Encoder

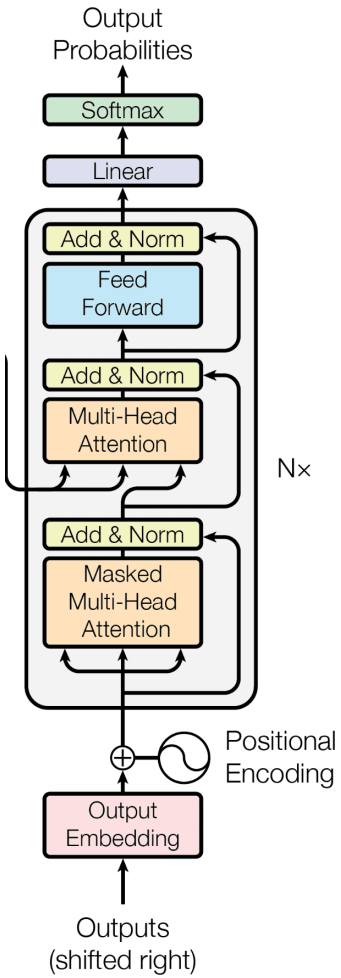
- The **encoder** uses **self-attention**
- It can be used to create a **contextualized bidirectional representat** sentence
- It can be used for tasks such as **classification**
- The main difference between the encoder and the decoder is that the **encoder uses the self-attention** while the **decoder the masked attention**



[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Jay Alammar, The Illustrated Transformer. https://jalammar.github.io/illustrated-transformer]
[Fontana, U. (2023, October 10). *NLP part 7 - self-attention and Transformers*. Medium. https://medium.com/@umbertofontana/nlp-part-7-self-attention-and-transformers-c770dea1283a]

Transformers - Decoder

- The Transformer Decoder constraints to **contextualized unidirectional context**
  - E.g., language modeling
- Add **masking** to self-attention
- It can be used for tasks such as **language modeling**



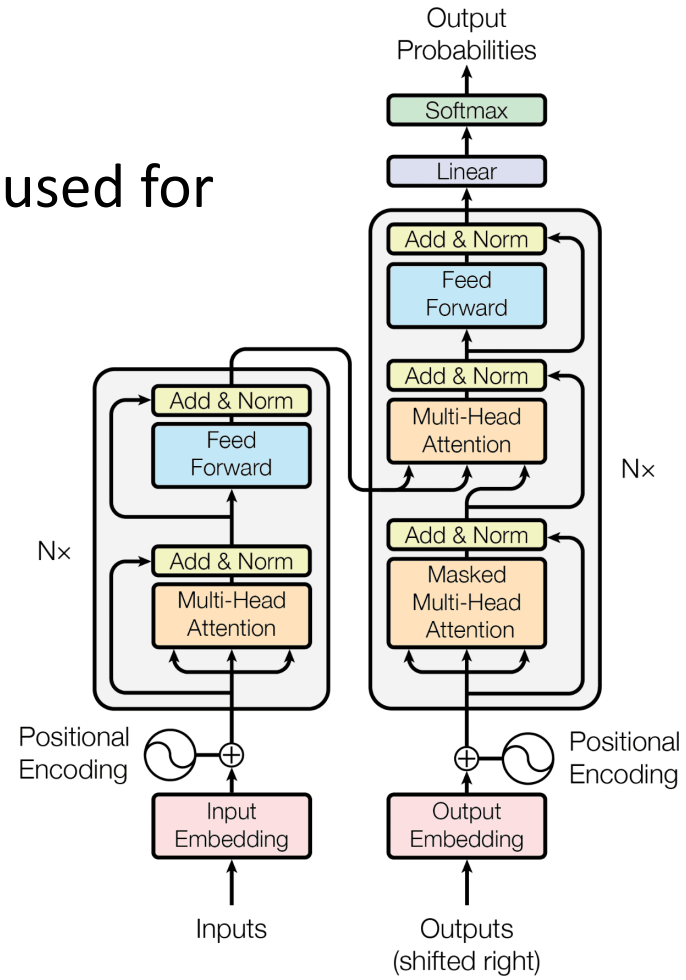
[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Jay Alammar, The Illustrated Transformer. https://jalammar.github.io/illustrated-transformer]
[Fontana, U. (2023, October 10). *NLP part 7 - self-attention and Transformers*. Medium. https://medium.com/@umbertofontana/nlp-part-7-self-attention-and-transformers-c770dea1283a]

Transformers Encoder-decoder

- The transformer model as an **encoder-decoder** can be used for **sequence to sequence** (sq2seq) tasks
  - E.g., machine translation

- In these tasks, we process the **source sentence with a bidirectional model** (encoding of the sentence) and generate the **target sentence with a unidirectional model**

- Transformer Decoder is modified to perform **cross-attention** to the output of the Encoder



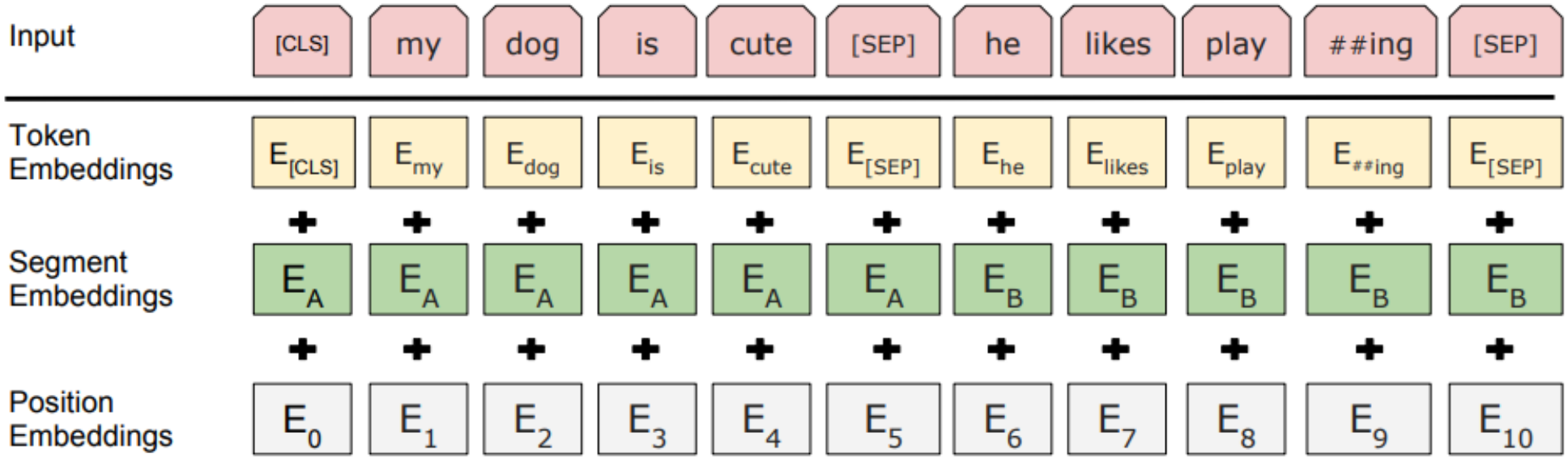
[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Jay Alammar, The Illustrated Transformer. https://jalammar.github.io/illustrated-transformer]
[Fontana, U. (2023, October 10). *NLP part 7 - self-attention and Transformers*. Medium. https://medium.com/@umbertofontana/nlp-part-7-self-attention-and-transformers-c770dea1283a]

BERT: Bidirectional Encoder Representations from Transformers

- BERT stands for ***B**idirectional **E**ncoder **R**epresentations from **T**ransformers*
- It uses **only the encoder** of the transformer architecture to learn bidirectional representations (12 layers in the base version)
- It is a **Language Model**
- Since it is an encoder, it cannot be used for Language Modeling. Thus, it introduces two tasks for the language modeling training:
  - (MLM) **Masked Language Modeling:** mask (remove) some fraction of words in the input with a special [MASK] token and predict these words
  - (NSP) **Next Sentence Prediction**

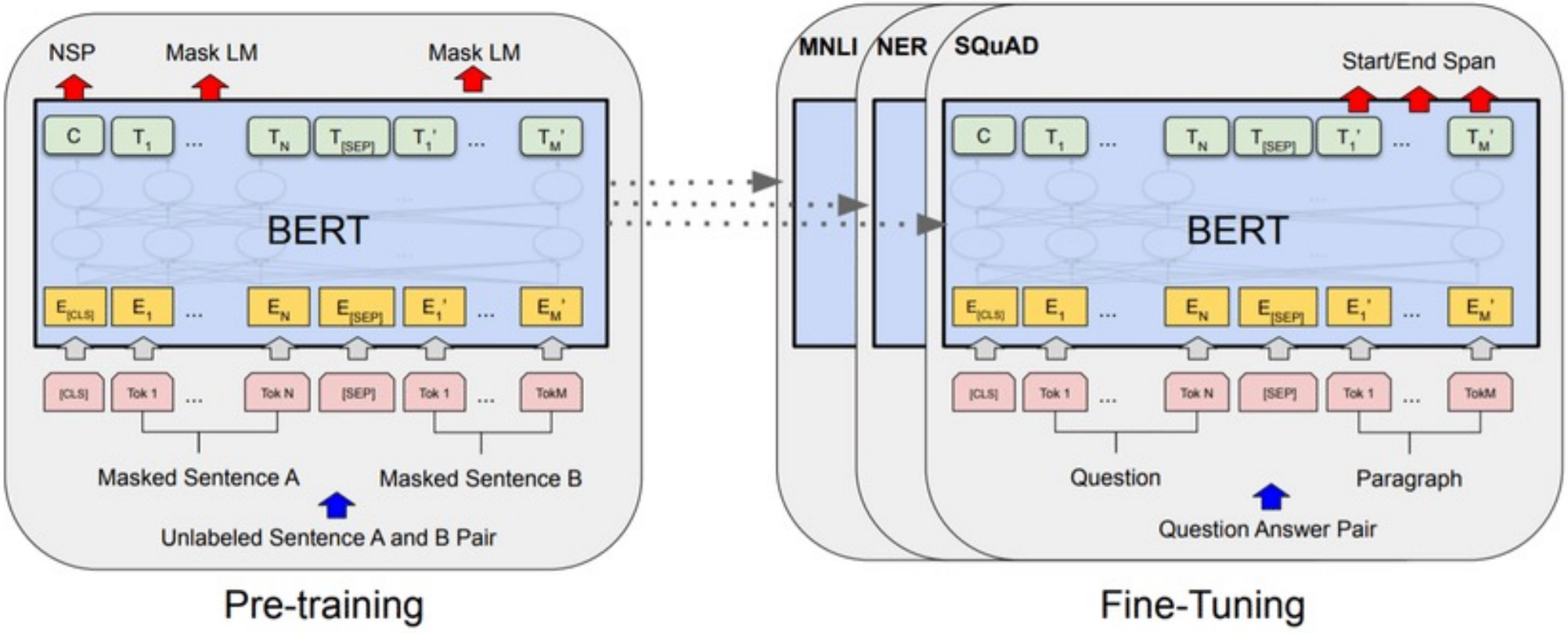
[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Devlin et al., BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL 2019 https://aclanthology.org/N19-1423]
[Jay Alammar, The Illustrated Transformer. https://jalammar.github.io/illustrated-transformer]

BERT: Bidirectional Encoder Representations from Transformers Input representation



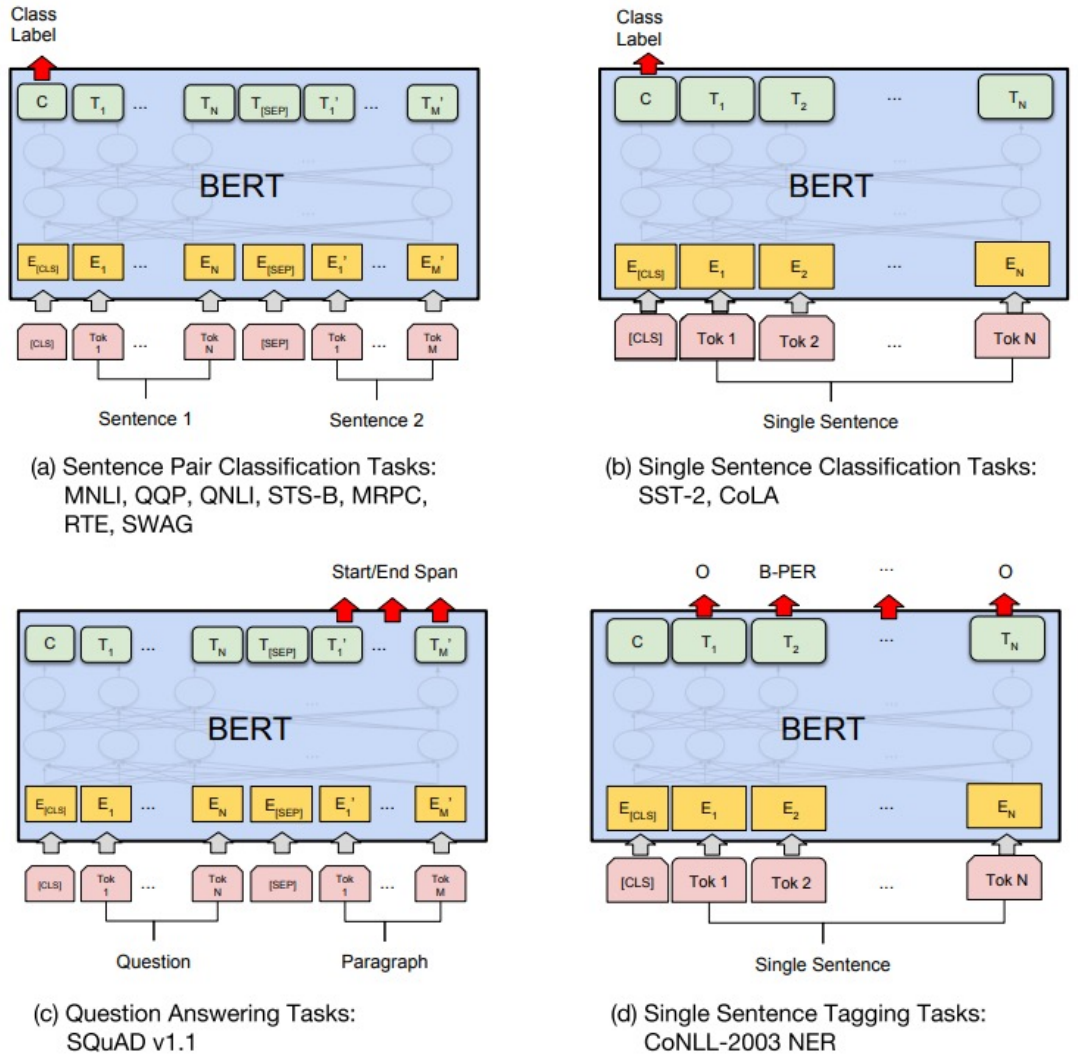
[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Devlin et al., BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL 2019 https://aclanthology.org/N19-1423]
[Jay Alammar, The Illustrated Transformer. https://jalammar.github.io/illustrated-transformer]

BERT: Bidirectional Encoder Representations from Transformers
Pre-training and Fine-tuning



[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Devlin et al., BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL 2019 https://aclanthology.org/N19-1423]
[Jay Alammar, The Illustrated Transformer. https://jalammar.github.io/illustrated-transformer]

BERT: Bidirectional Encoder Representations from Transformers



[Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]
[Devlin et al., BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL 2019 https://aclanthology.org/N19-1423]
[Tirana Fatyanosa Fine-Tuning Pre-Trained Transformer-based Language Model. https://fatyanosa.medium.com/fine-tuning-pre-trained-transformer-based-language-model-c542af0e7fc1]

BERT: Bidirectional Encoder Representations from Transformers

- We will train, use, and explain BERT in the next lab

- Additional consideration
  - Transformer models usually use sub-tokens tokenization
    - E.g., Sub ##token
  - BERT contains an additional token [CLS] to perform classification

HuggingFace Tutorial

References (1)

These slides (and many images) are inspired and taken by [Manning, C. CS224N: Natural Language Processing with deep learning. Stanford CS 224N | Natural Language Processing with Deep Learning. https://web.stanford.edu/class/cs224n/index.html]

[Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. https://arxiv.org/abs/1301.3781]

[Ogundepo, O. (2021, July 26). Understanding word2vec. Medium. https://medium.com/analytics-vidhya/understanding-word2vec-39fabe660705]

[Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics. https://aclanthology.org/D14-1162.pdf]

[Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, & Jeffrey Dean. (2013). Distributed Representations of Words and Phrases and their Compositionality. https://arxiv.org/pdf/1310.4546]

[Bolukbasi, T., Chang, K. W., Zou, J. Y., Saligrama, V., & Kalai, A. T. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings. Advances in neural information processing systems, 29. https://proceedings.neurips.cc/paper_files/paper/2016/file/a486cd07e4ac3d270571622f4f316ec5-Paper.pdf]

[Lopardo, A. (2019, December 23). The basics of Language Modeling. Medium. https://medium.com/@antonio.lopardo/the-basics-of-language-modeling-1c8832f21079]

[Voita, L. Language modeling. https://lena-voita.github.io/nlp_course/language_modeling.html#main_content]

References (2)

[Olah, C. *Understanding LSTM networks*. Understanding LSTM Networks -- colah's blog. https://colah.github.io/posts/2015-08-Understanding-LSTMs]

[Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). https://papers.nips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf]

[Fontana, U. (2023, October 10). *NLP part 7 - self-attention and Transformers*. Medium. https://medium.com/@umbertofontana/nlp-part-7-self-attention-and-transformers-c770dea1283a]

[Devlin et al., BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL 2019 https://aclanthology.org/N19-1423]

[Jay Alammar, The Illustrated Transformer. https://jalammar.github.io/illustrated-transformer]

[Tirana Fatyanosa Fine-Tuning Pre-Trained Transformer-based Language Model. https://fatyanosa.medium.com/fine-tuning-pre-trained-transformer-based-language-model-c542af0e7fc1