Deep Learning for Language Understanding

Learning Outcomes

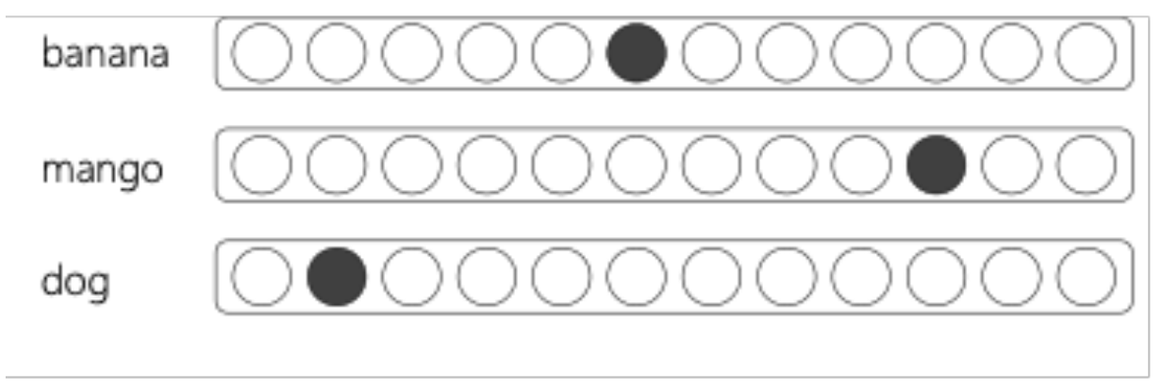
- Word2Vec distributed representation of words
- Language models using Recurrent Neural Networks
- Neural Machine Translation using Seq2Seq models
- Attention mechanisms
  - Encoder decoder attention
  - Self attention
  - Multi-head attention
  - Transformer model

How to Represent Text

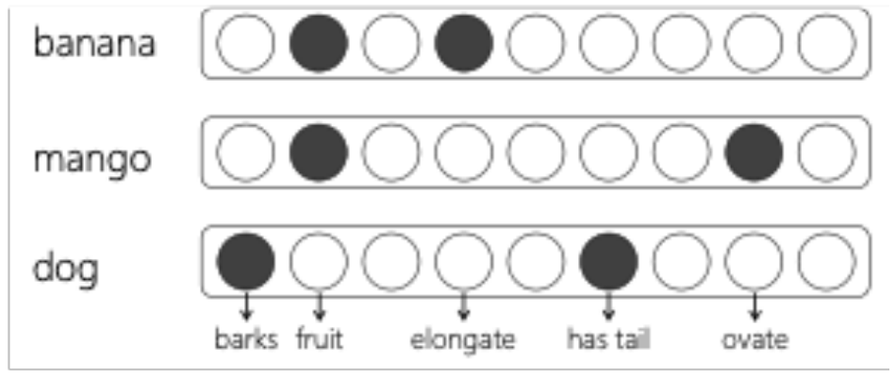
- Humans understand natural language
- Machines understand numbers
- How to represent natural language text in numbers?
- E.g, how to represent "banana is a fruit" as numbers?
  - One-hot-encoding: [0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]
  - Other solutions?

Word Representations

One-hot encoding vs distributed representation



Local representations (1-hot)



Distributed representations

Limitations?

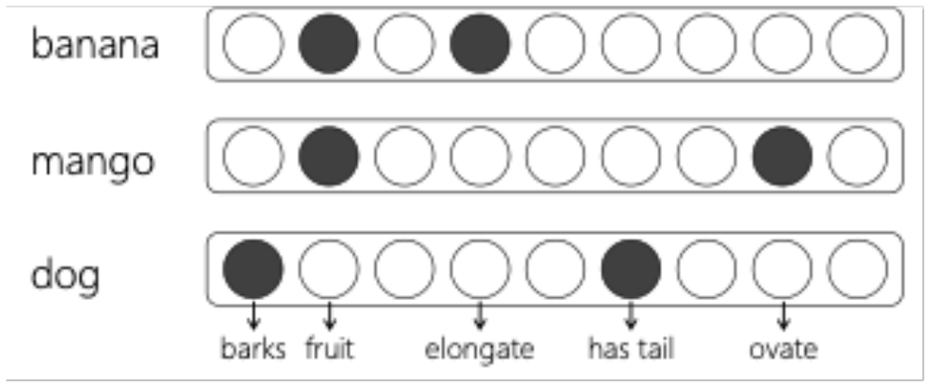
Sparseness: Lot's of 0s and few 1s

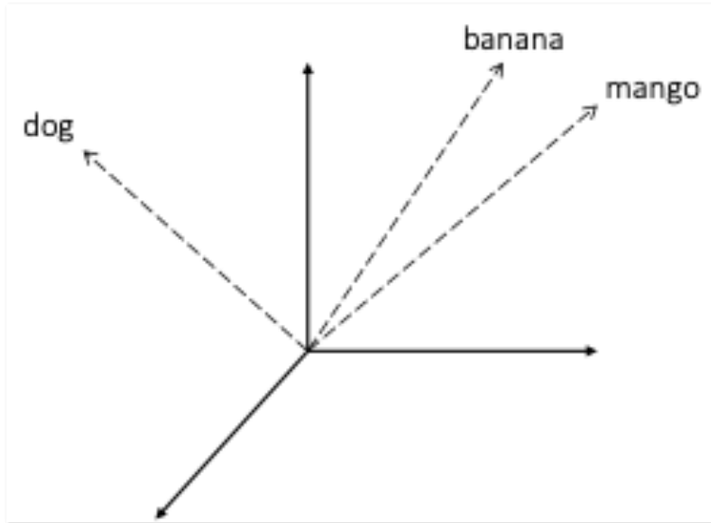
High dimensionality: 20K (speech) – 50K (PTB) – 500K (big vocab) – 13M (Google 1T)

Distributed Representation

- You can get a lot of value by representing a word by means of its neighbors "You shall know a word by the company it keeps" — (J. R. Firth 1957: 11)

Distributed Representation



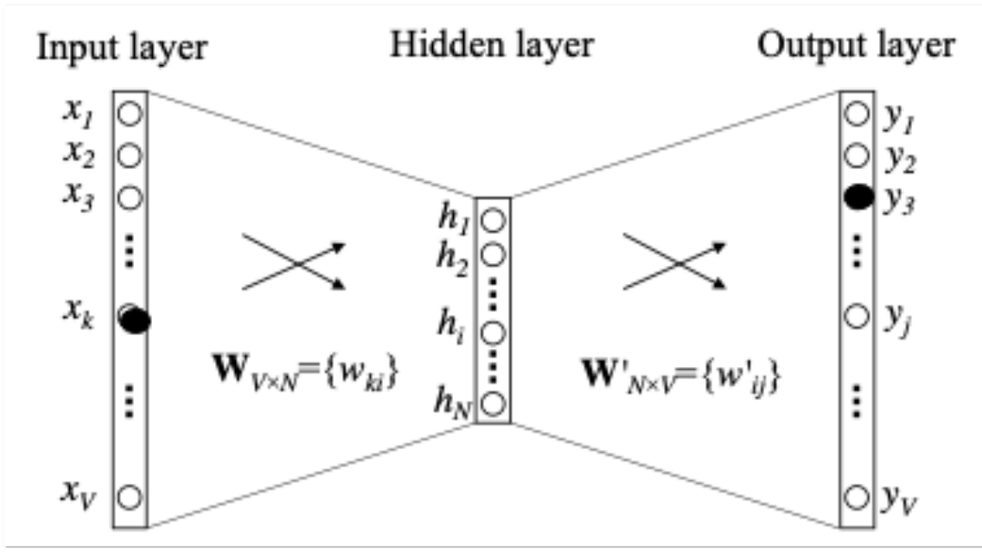


…government debt problems turning into banking crises as has happened in…
…..saying that Europe needs unified banking regulation to replace the hodgepodge…

↖ These words will represent *banking* ↗

Word Embeddings

- Goal: Given one-hot encodings of words and their context represent them in dense lower dimension space
  - Considers learning word embeddings as a prediction task
  - Training context : "Norway wins gold in Olympics"
  - **Continuous Bag of Words Model (CBOW):**
    - Guess the central word in the context
    - "norway wins X in Olympics"
    - Training data (Norway, gold), (wins, gold), (in, gold), (olympics, gold), (olympics, gold),
  - **Skip-gram**
    - Guess the context given the central word
    - "X X gold X X"
    - Training data (gold, Norway), (gold, wins), (gold, in), (gold, Olympics)



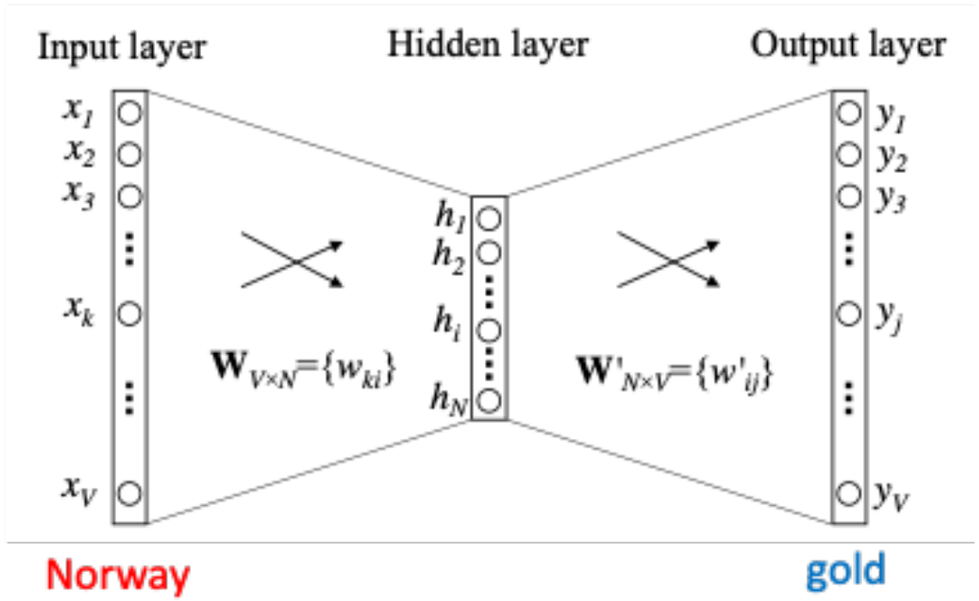
Norway gold

Is it a regression or a classification problem ?

Word2Vec

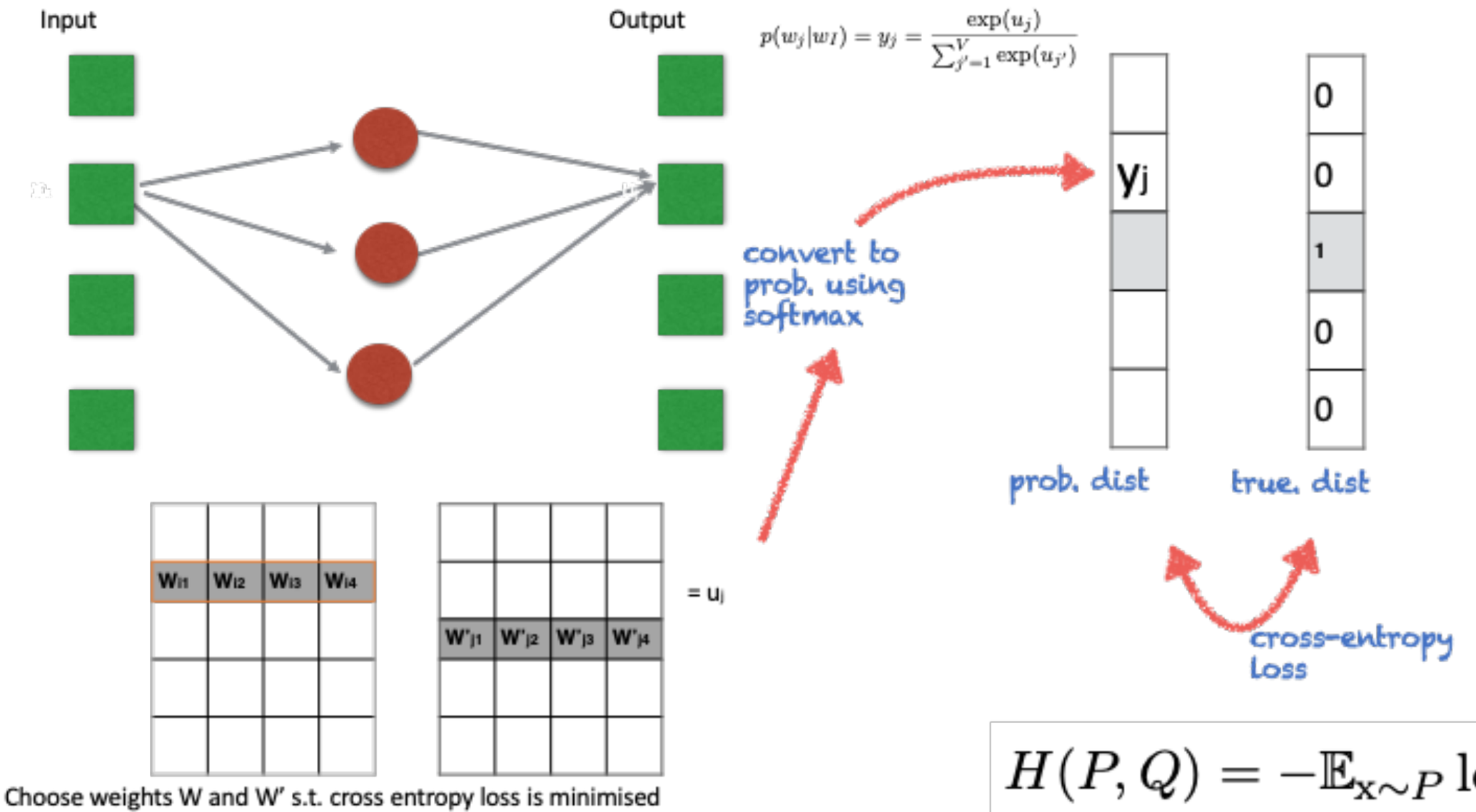
- Input and Output dimensions are size V (vocabulary), hidden layer N << V

- Input representation : **h** in the hidden space
- Output representation : **h.W'**



- Central question: how do we learn the parameters of this model (weights) given training examples from text
- What is cosine distance between output and input representation ?

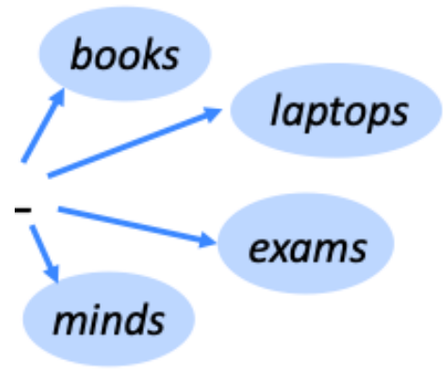
Word2Vec Forward Pass



$$H(P, Q) = -\mathbb{E}_{x \sim P} \log Q(x)$$

Language Modeling Task

- Language Modeling is the task of predicting what word comes next.
- the students opened their _____



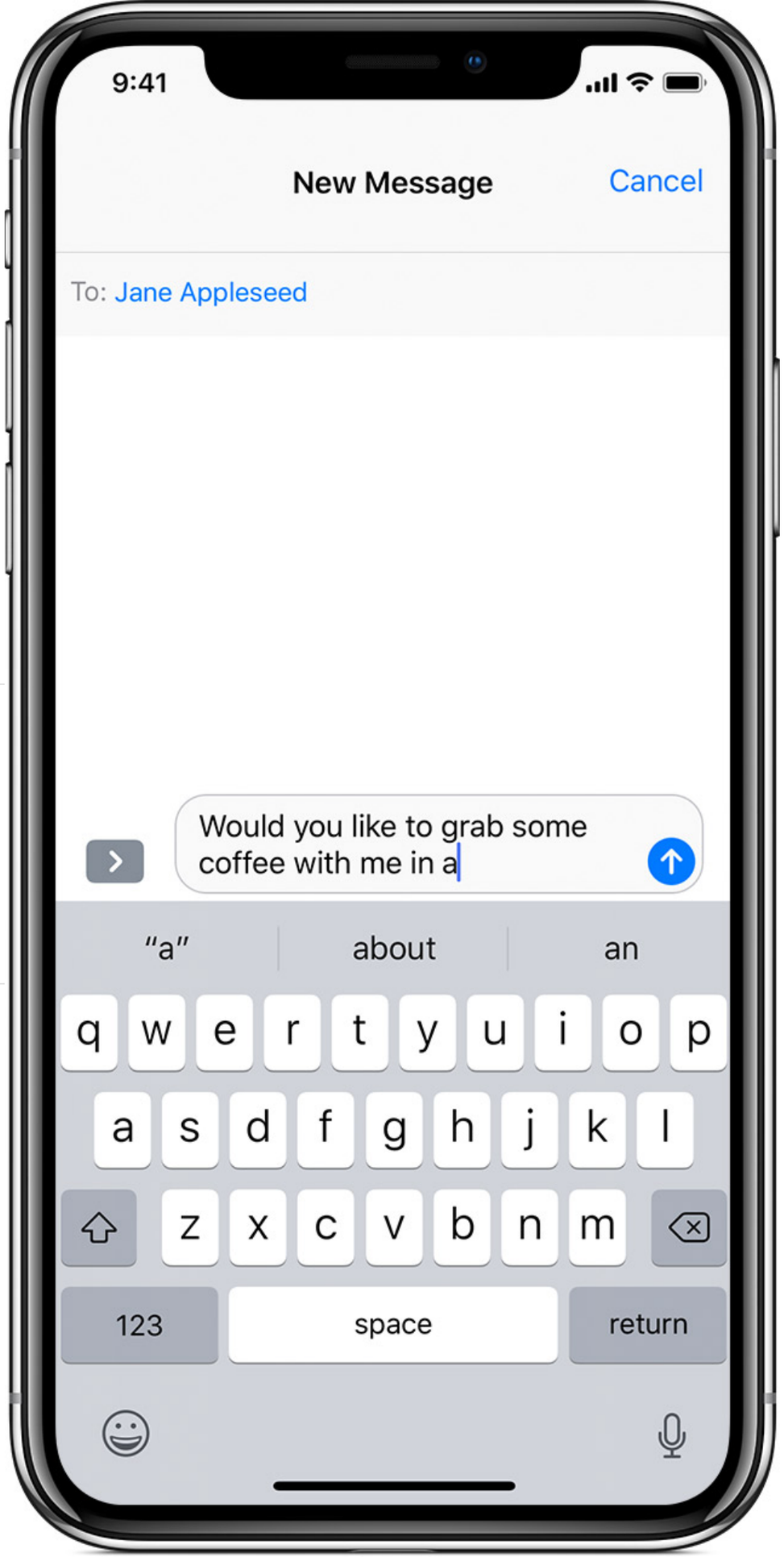
- More formally: given a sequence of words $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$ compute the probability distribution of the next word $\boldsymbol{x}^{(t+1)}$

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

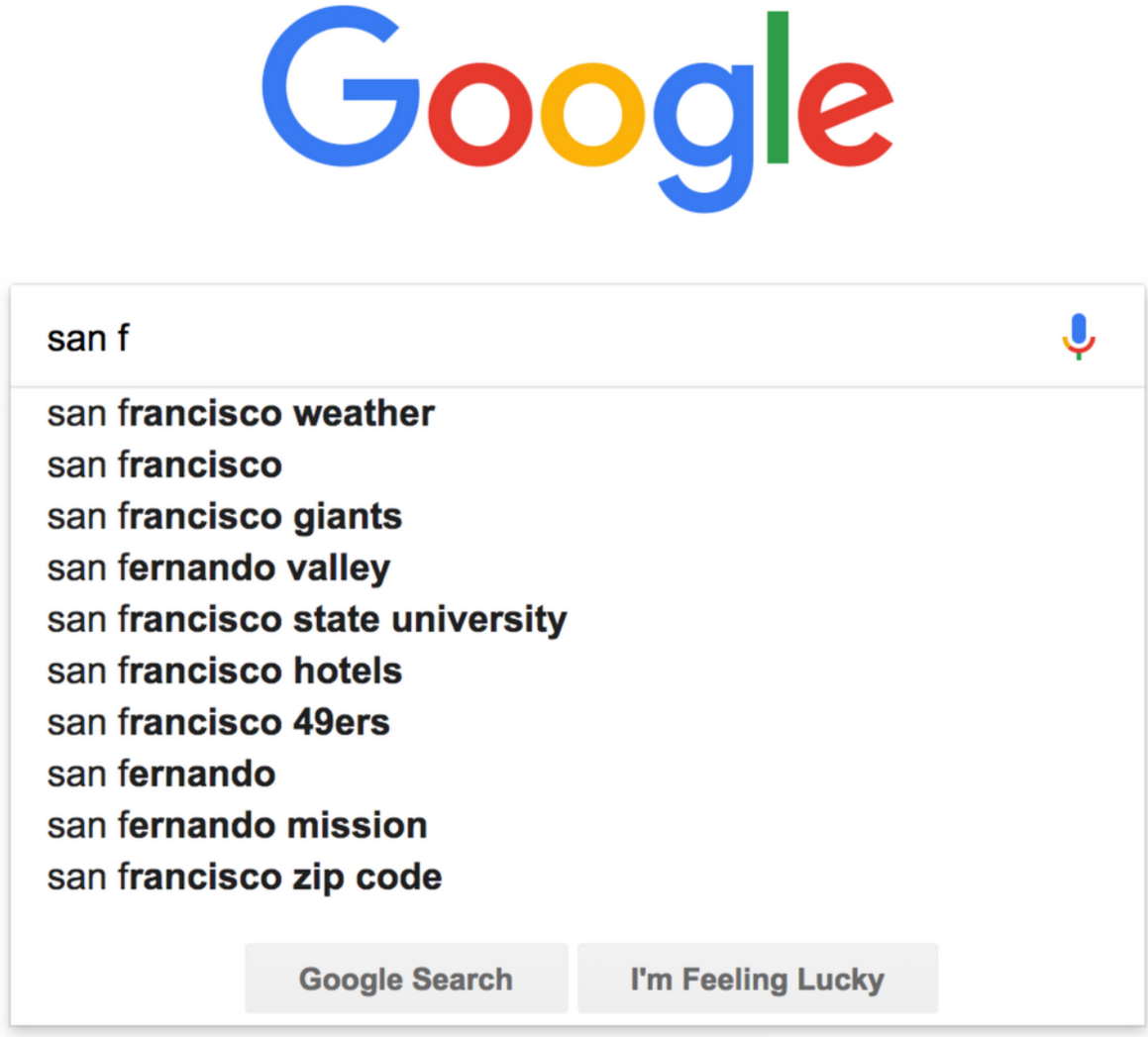
  Where $\boldsymbol{x}^{(t+1)}$ can be any word in the vocabulary $V = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{|V|}\}$

- A system which can compute this is called a "Language Model"

You use Language Models every day!



You use Language Models every day!



Statistical Language Models

- Pre-deep learning era: Statistical language models were used

> Idea: Collect statistics about how frequently different word sequences (n-grams) occur and use that to predict next word

Definition: *An n-gram is a chunk of n consecutive words.*

- unigrams: “the”, “students”, “opened”, ”their”
- bigrams: “the students”, “students opened”, “opened their”
- trigrams: “the students opened”, “students opened their”
- 4-grams: “the students opened their”

Statistical Langauge Models

- Learning language models from large text corpora
  - Given a large text corpus – say all sentences in Wikipedia

- Count frequency of occurrence of words – unigrams, bigrams, tri-grams ,…

$$p(w_1) = \frac{\text{count}(w_1)}{\sum \text{count}(w_i)} \qquad p(w_2|w_1) = \frac{\text{count}(w_1, w_2)}{\text{count}(w_1)} \qquad p(w_3|w_1, w_2) = \frac{\text{count}(w_1, w_2, w_3)}{\text{count}(w_1, w_2)}$$

- Why is more context better ?
  - Tri-gram > Bi-gram > Unigram

- Typically have a Markovian assumption

P(hot summer the has been)

P(hot the summer has been)

P(the summer has been hot)

$$P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i|w_1, \ldots, w_{i-1}) \approx \prod_{i=1}^{m} P\left(w_i|w_{i-(n-1)}, \ldots, w_{i-1}\right)$$

Problems with Statistical LM

What comes next ? students opened their X

Sparsity Problem 1

Problem: What if *"students opened their* w_j*"* never occurred in data? Then w_j has probability 0!

(Partial) Solution: Add small δ to count for every $w_j \in V$. This is called *smoothing*.

$$p(w_j \mid \text{students opened their}) = \frac{\text{count(students opened their } w_j)}{\text{count(students opened their)}}$$

Sparsity Problem 2

Problem: What if *"students opened their"* never occurred in data? Then we can't calculate probability for *any* w_j!

(Partial) Solution: Just condition on *"opened their"* instead. This is called *backoff*.

Problems with Statistical LM

- Sparsity
  - Need lot of data to estimate long-length n-grams
  - Assume |V| = 10^5. What is the number of contexts for 4-grams ?

$$p(w_3|w_1, w_2) = \frac{\text{count}(w_1, w_2, w_3)}{\text{count}(w_1, w_2)}$$

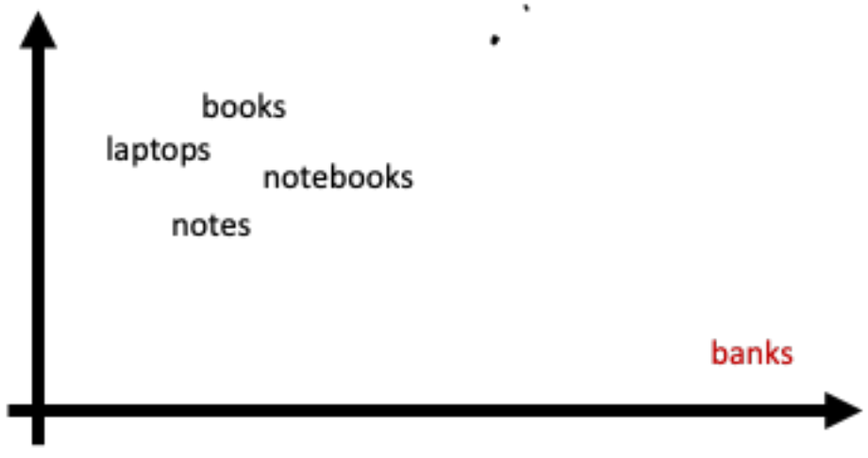
Most of them are not seen

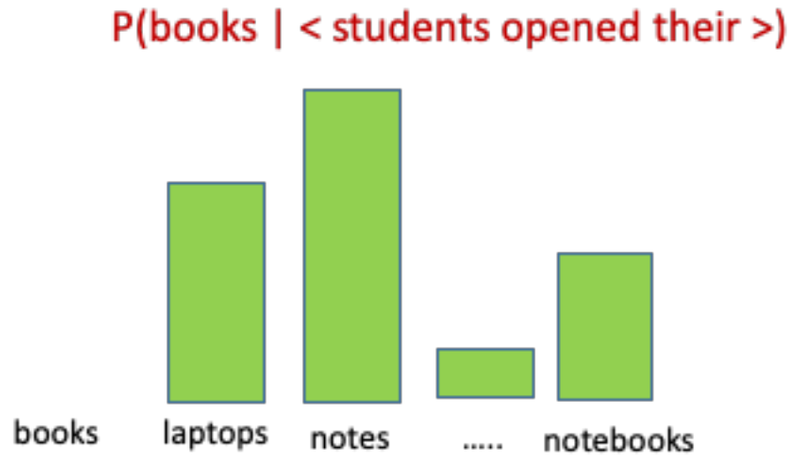
- Insufficient context
  - Markovian assumption imposes a computational limit on the context

- What is needed ?
  - We could estimate the probabilities for n-grams better by exploiting word similarity
  - Why not let a Neural Network estimate these probabilities automatically ?

Neural Language Models

- How does neural LM solve the sparsity problem ?
  - Words as dense distributed vectors so there can be sharing of statistical weight between similar words
  - Doing just this solves the sparseness problem of conventional n-gram models
    - **students opened their** {books, laptops, notes, notebooks, ….}
    - Even if books is not observed or P(books | < **students opened their** >) = 0

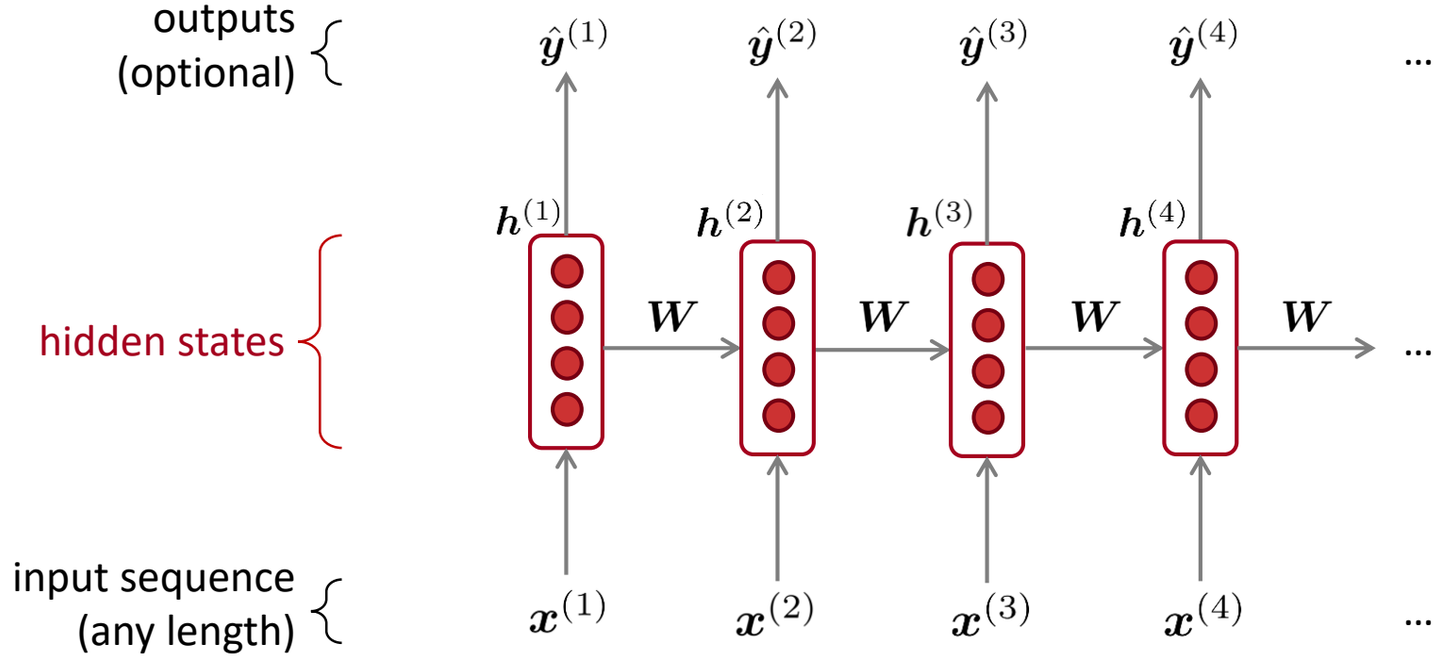




Recap: Recurrent Neural Networks

- Given a function f: $h^{(i)}, y^{(i)} = f(h^{(i-1)}, x, W)$

$h^{(i)}$ and $h^{(i-1)}$ are vectors with the same dimension

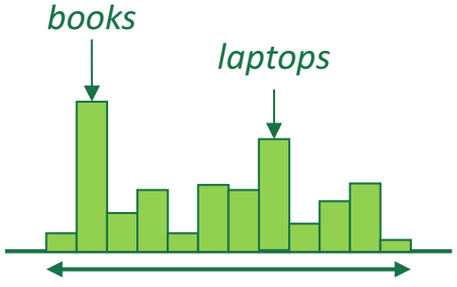


Core idea: Apply same function f with same weights W repeatedly

RNN Language Models

- Input
  - Word sequence represented as **word2vec embeddings** or **one-hot encodings**

$$\hat{\boldsymbol{y}}^{(4)} = P\left(\boldsymbol{x}^{(5)}|\text{ the students opened their }\right)$$



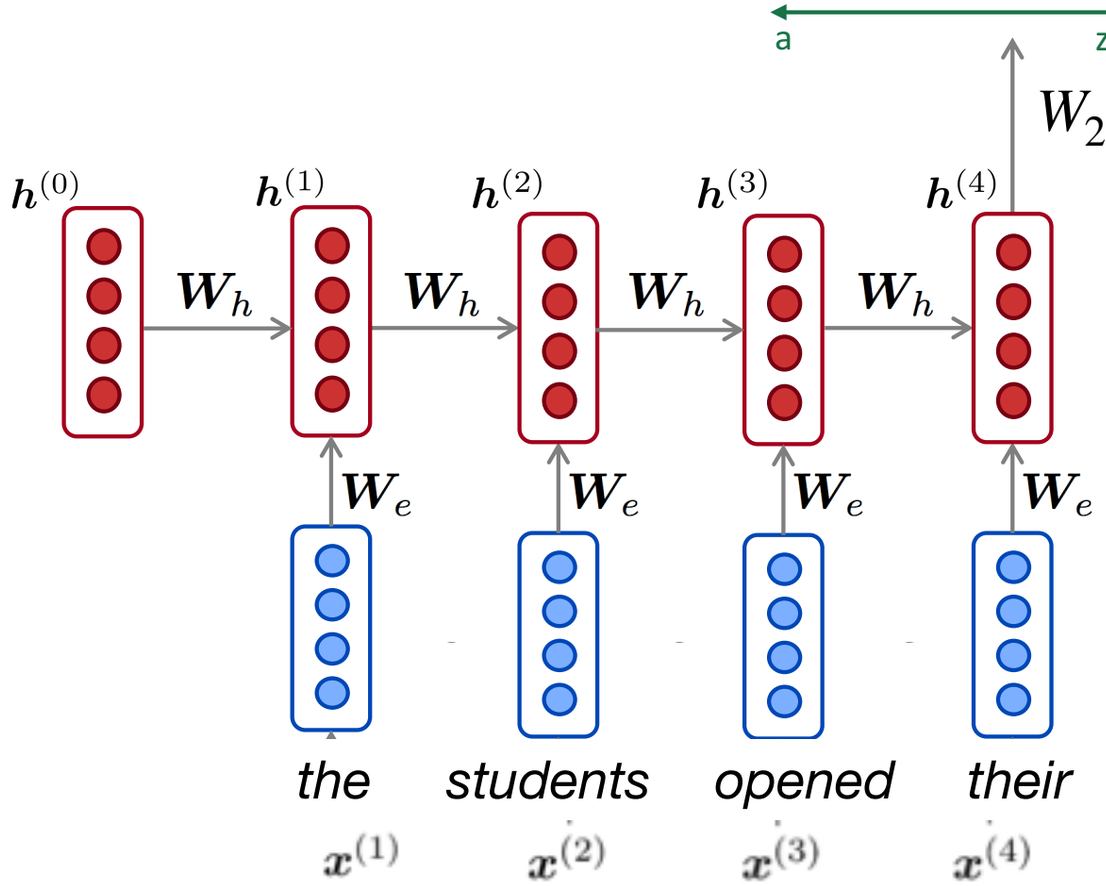
- hidden states

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

$\boldsymbol{h}^{(0)}$ Initial hidden state

- Output

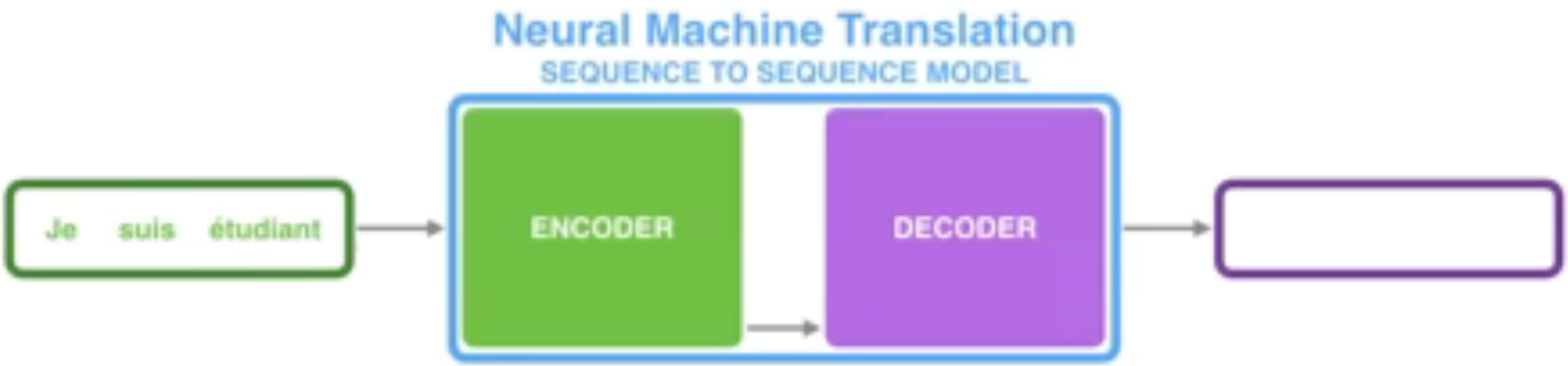
$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$



Sequence-to-Sequence Models

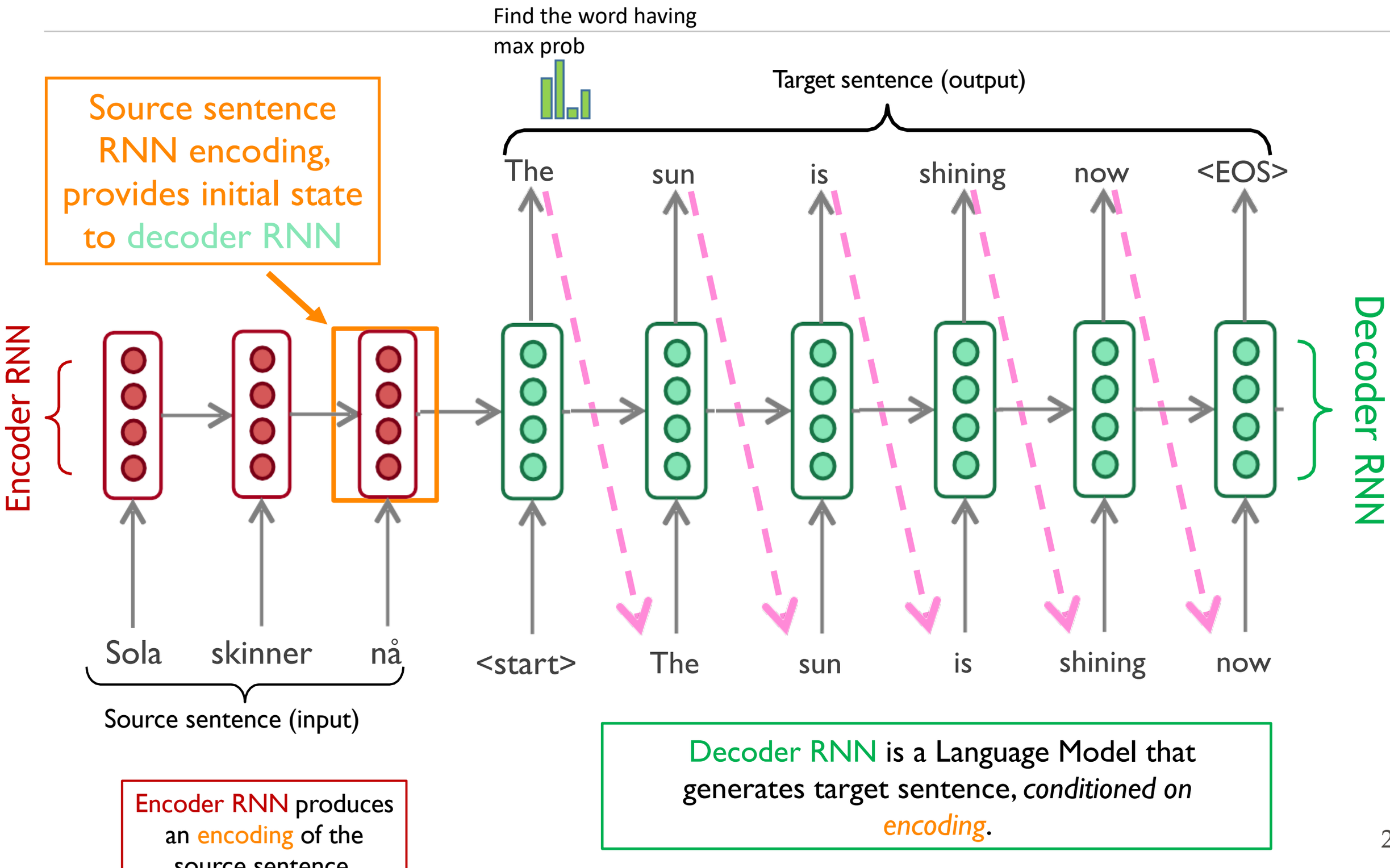
- Seq2Seq turns one sequence into another sequence
- Many NLP tasks can be phrased as Seq2Seq:
  - Summarization (long text → short text)
  - Dialogue (previous utterances → next utterance)
  - Parsing (input text → output parse as sequence)
  - Neural Machine Translation (Norwegian → English)

Seq2Seq Neural Machine Translation



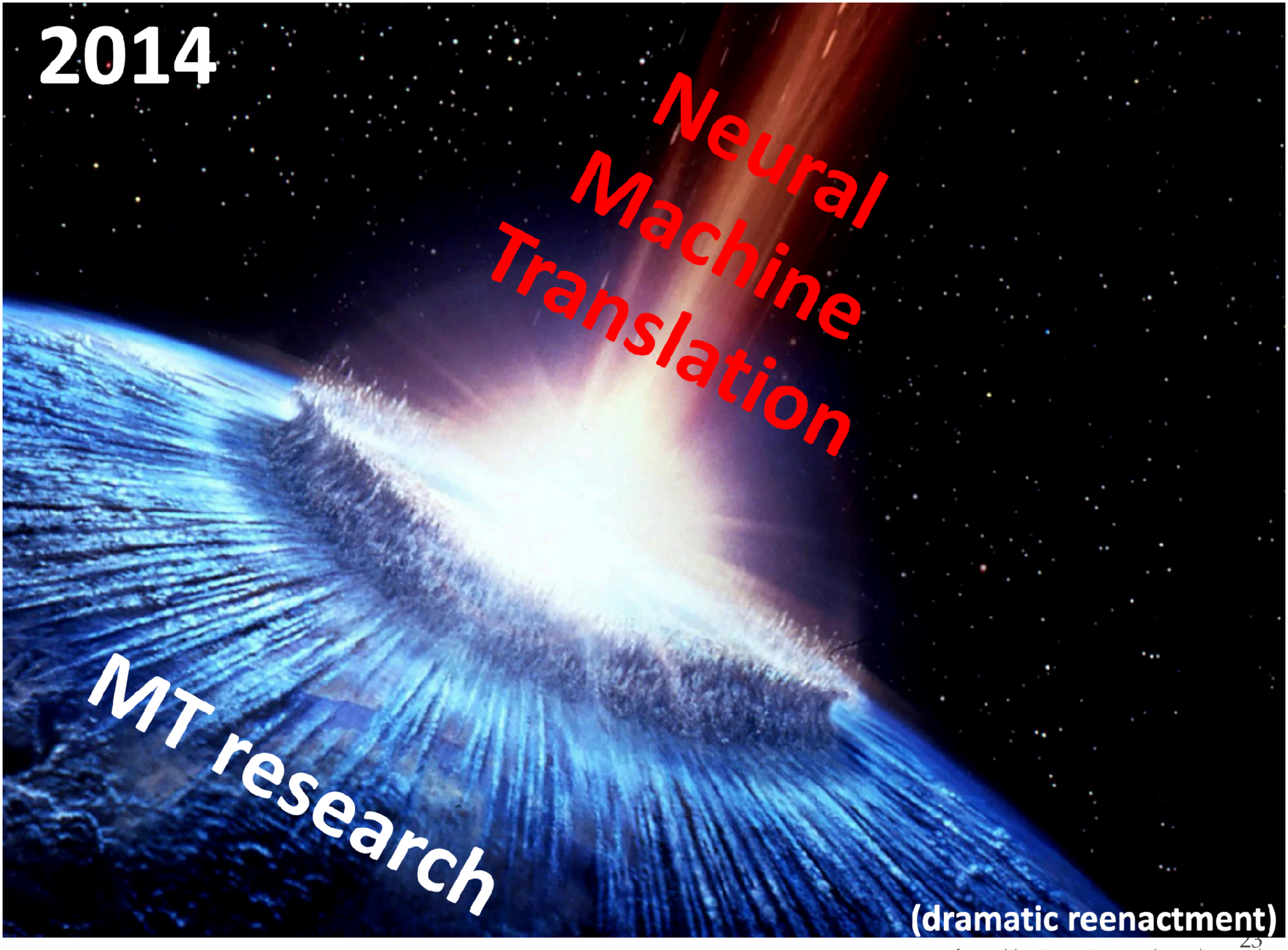
Source: http://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

RNN Neural Machine Translation



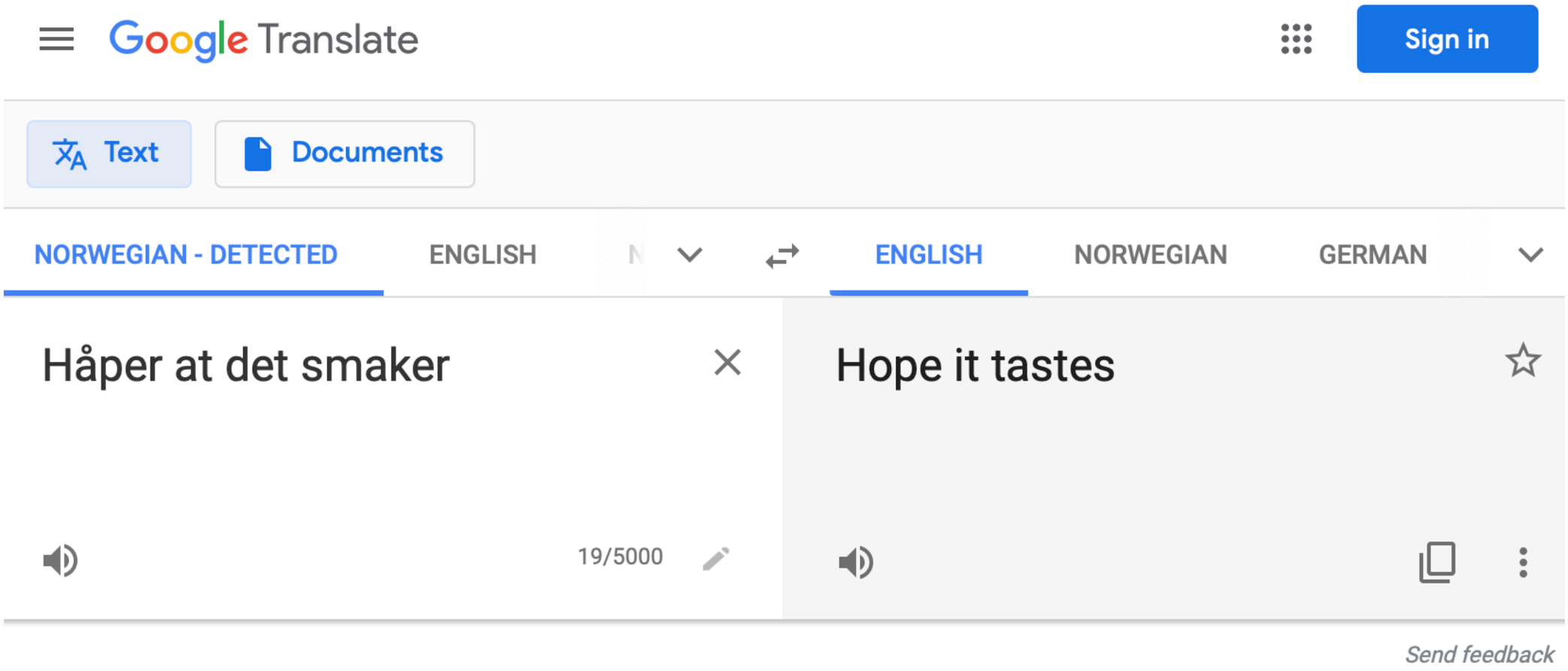
Encoder RNN produces an encoding of the source sentence.

Decoder RNN is a Language Model that generates target sentence, *conditioned on encoding*.

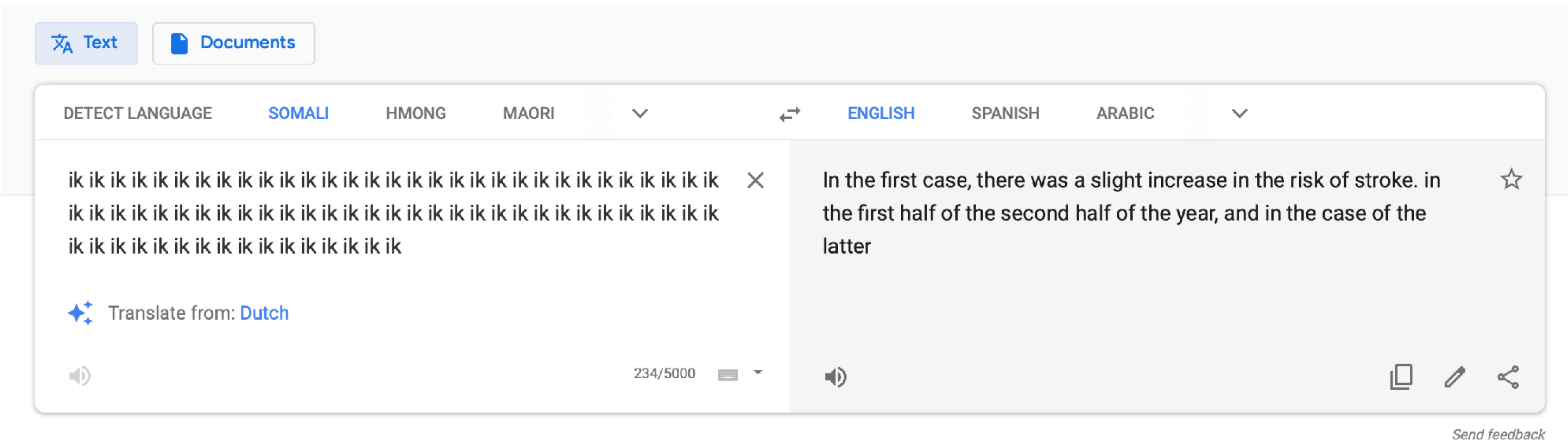


Source: https://web.stanford.edu/class/cs224n/

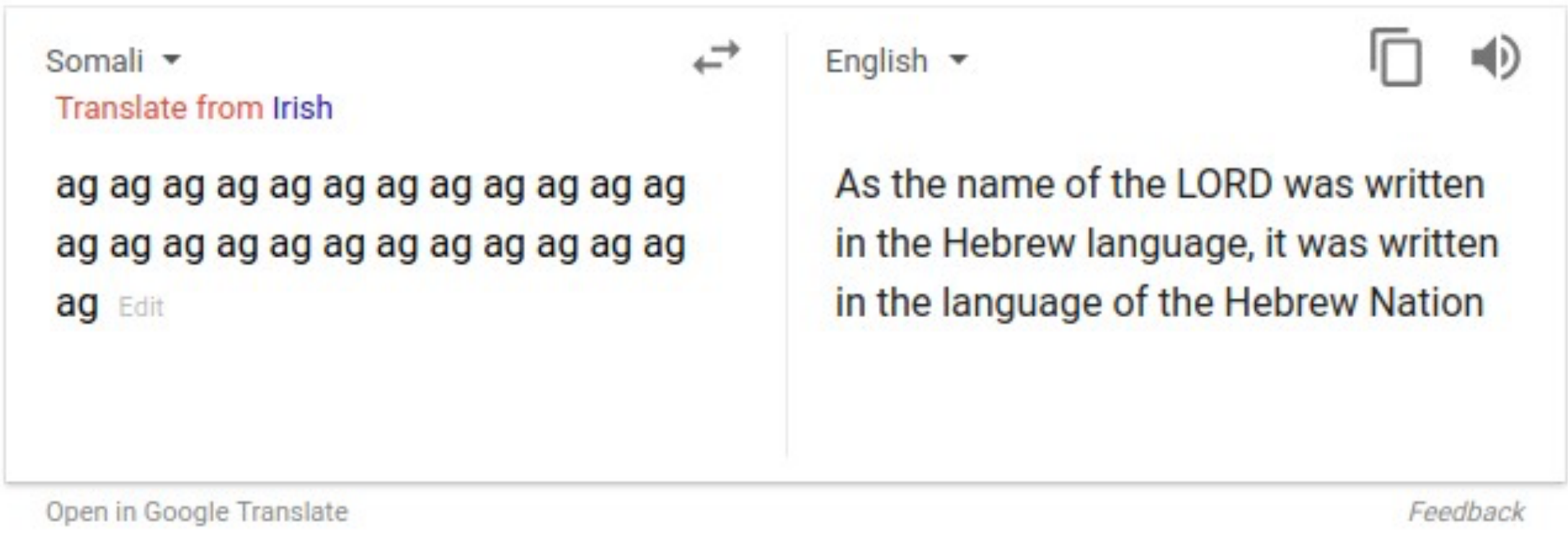
Is Machine Translation Solved?



Is Machine Translation Solved?



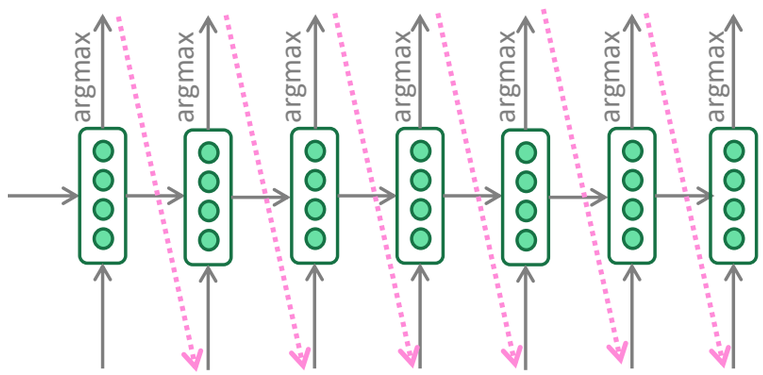
Is Machine Translation Solved?

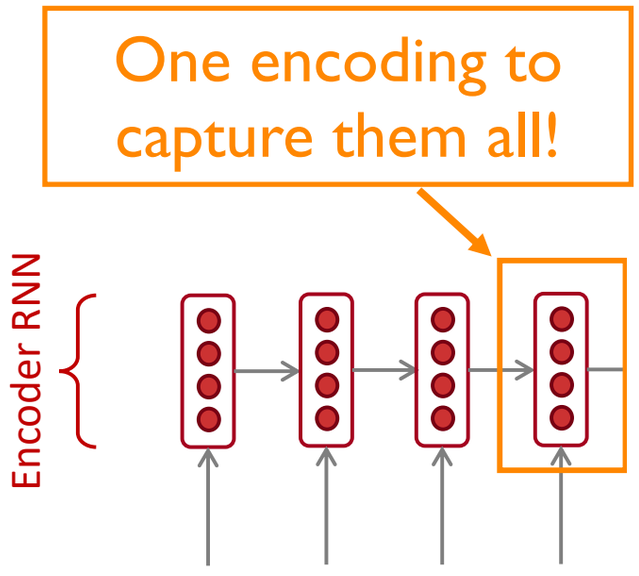


Source: https://www.skynettoday.com/briefs/google-nmt-prophecies

Problems with RNN NMT

- Decoding problem: is choosing best candidate at each time step optimal ?
  - Problem of decoding – argmax not good enough
    - Solution: **Beam search** (beyond the scope for today)
- Bottleneck problem: Encoding everything into one vector
  - **Long sentences and distant contexts**





Seq-to-Seq Bottleneck Problem

"You can't cram the meaning of a whole %&!$ing sentence into a single $&!*ing vector!"

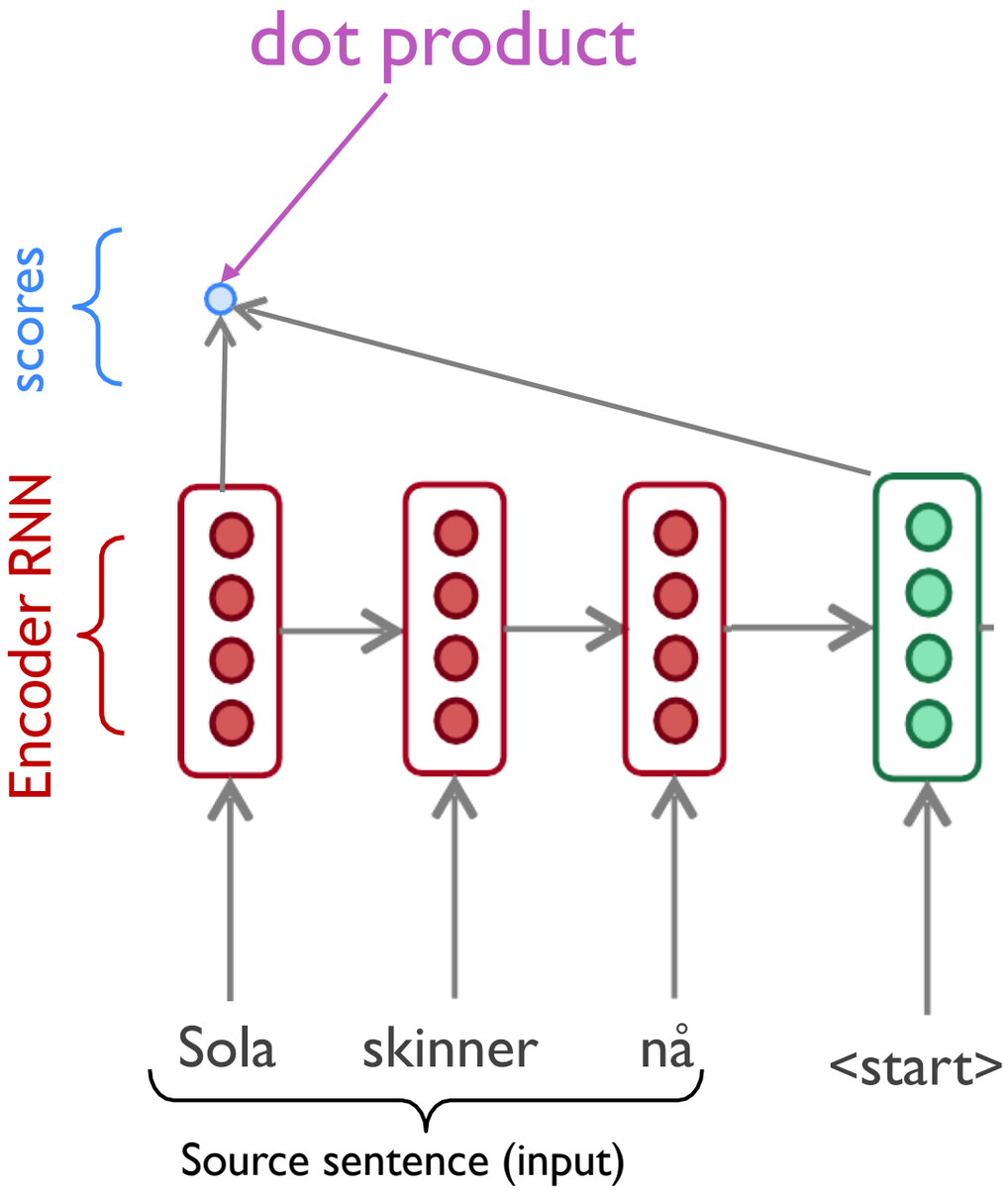
Ray Mooney

- How to solve the bottleneck problem?
  - Given an already decoded output we should be able to refer back to the original statement
- Problem of memory
  - Decoder can't remember all the information from the single encoding of the source sentence

Attention Mechanism

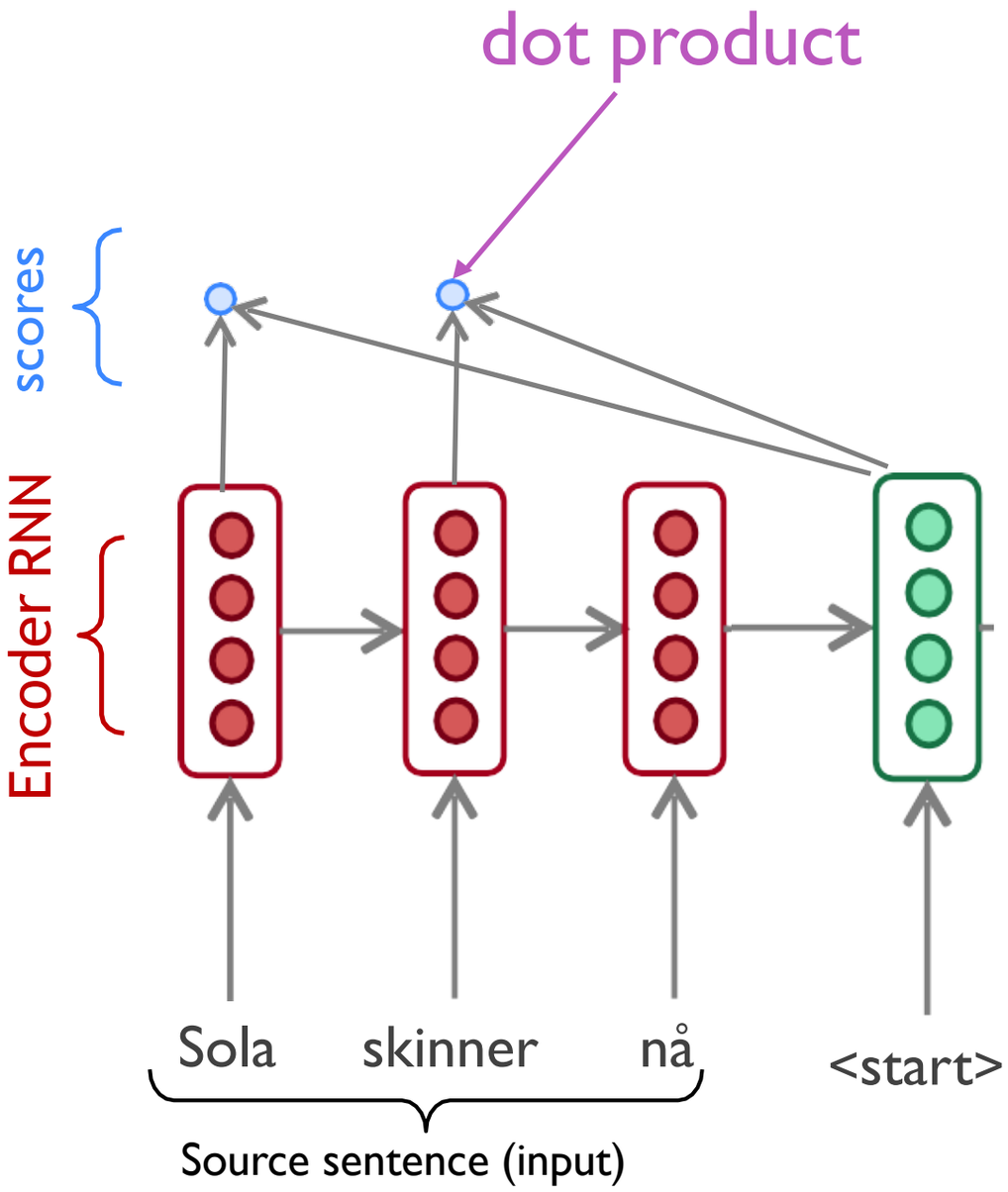
- Attention research is an enormous field with a long history in **cognitive neuroscience**
  - Humans do not process all sensory inputs to make a decision
  - We **focus on small fraction of inputs**
- **Attention mechanism in seq2seq models**
  - Provides solution to the bottleneck problem
  - Allows the decoder to **focus on part of the source sequence** at each time step

Seq2Seq with Attention



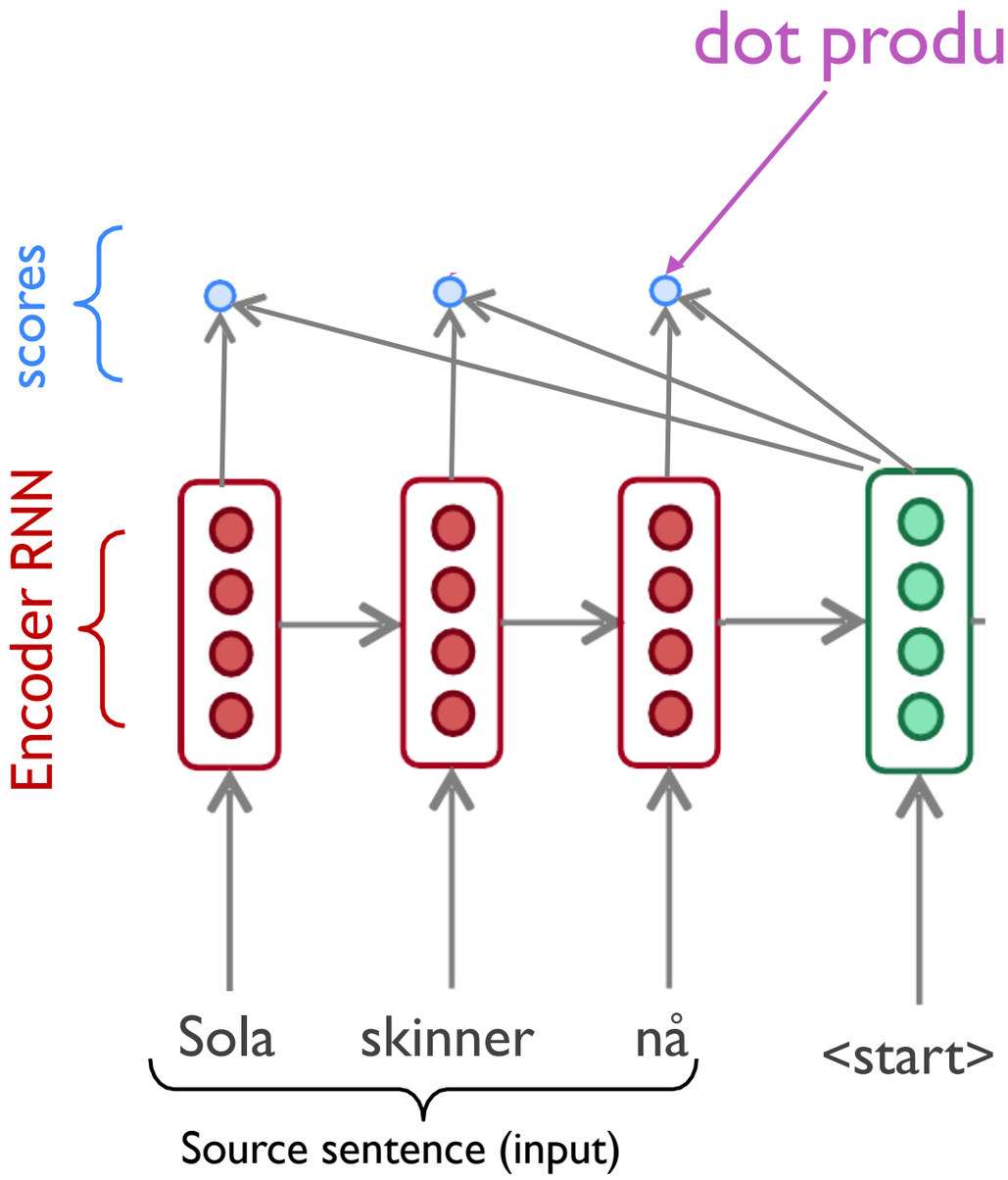
Encoder RNN produces an encoding of the source sentence.

Seq2Seq with Attention



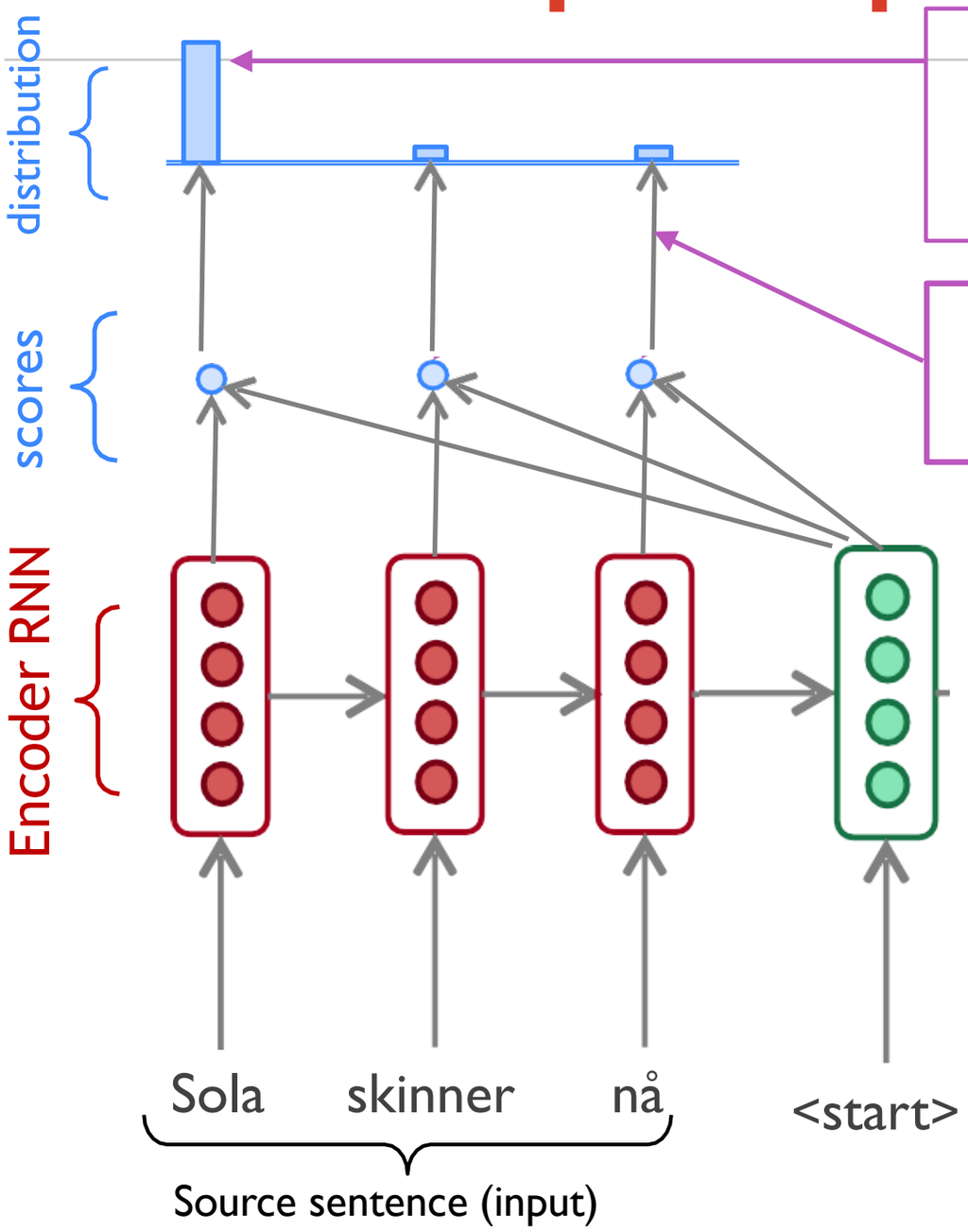
Encoder RNN produces an encoding of the source sentence.

Seq2Seq with Attention



Encoder RNN produces an encoding of the source sentence.

Seq2Seq with Attention

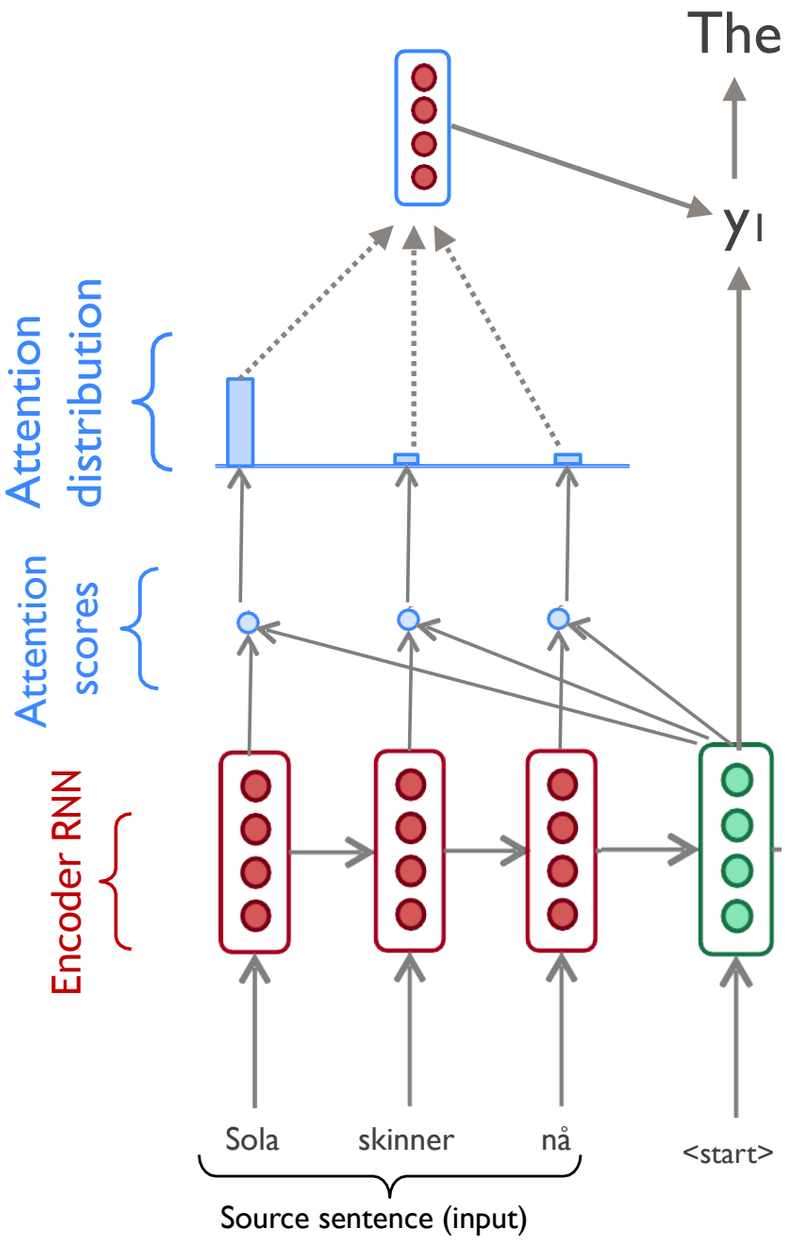


On this decoder timestep, we're mostly focusing on the first encoder hidden state (*"The sun"*)

Take softmax to turn the scores into a probability distribution

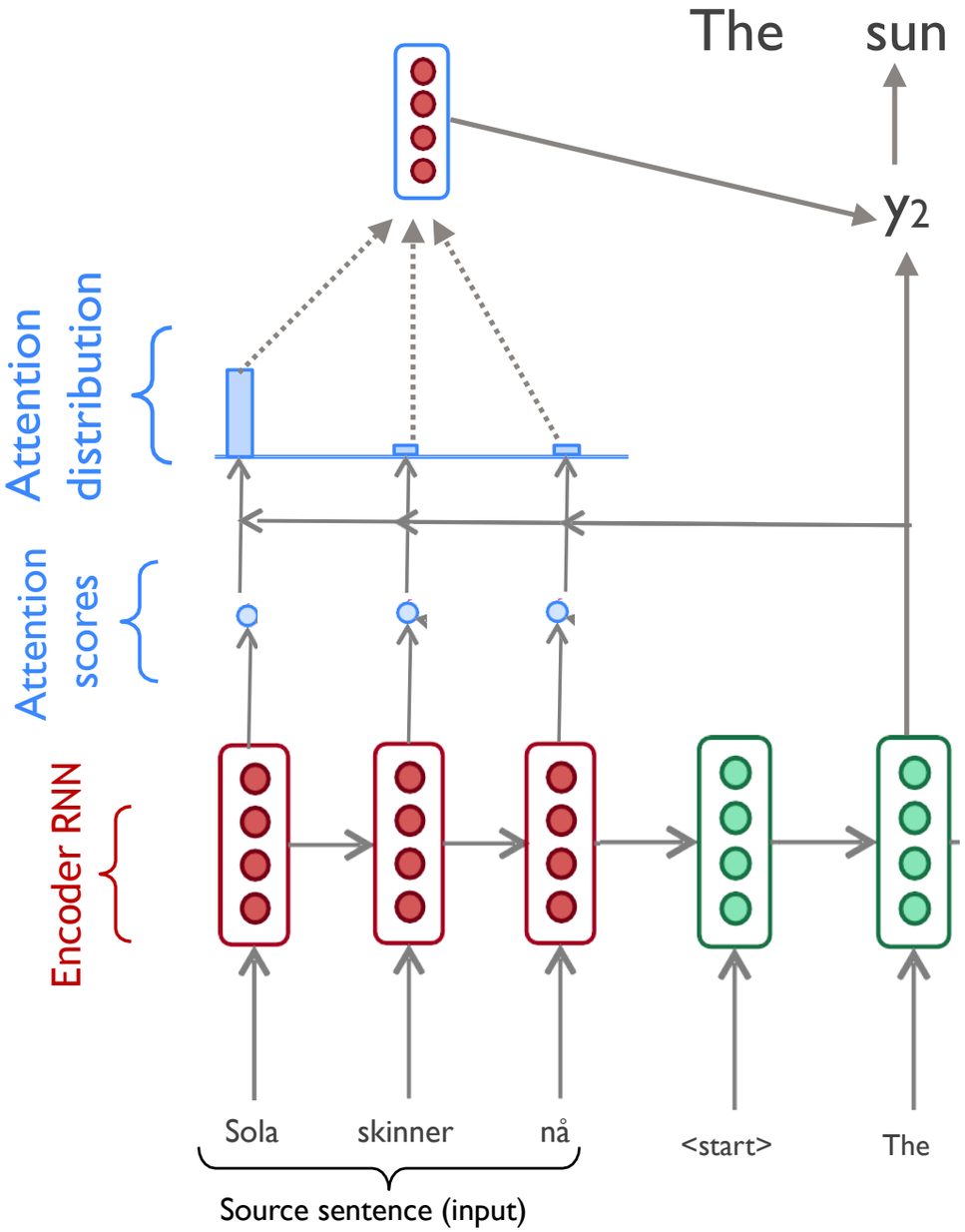
Encoder RNN produces an encoding of the source sentence.

Seq2Seq with Attention

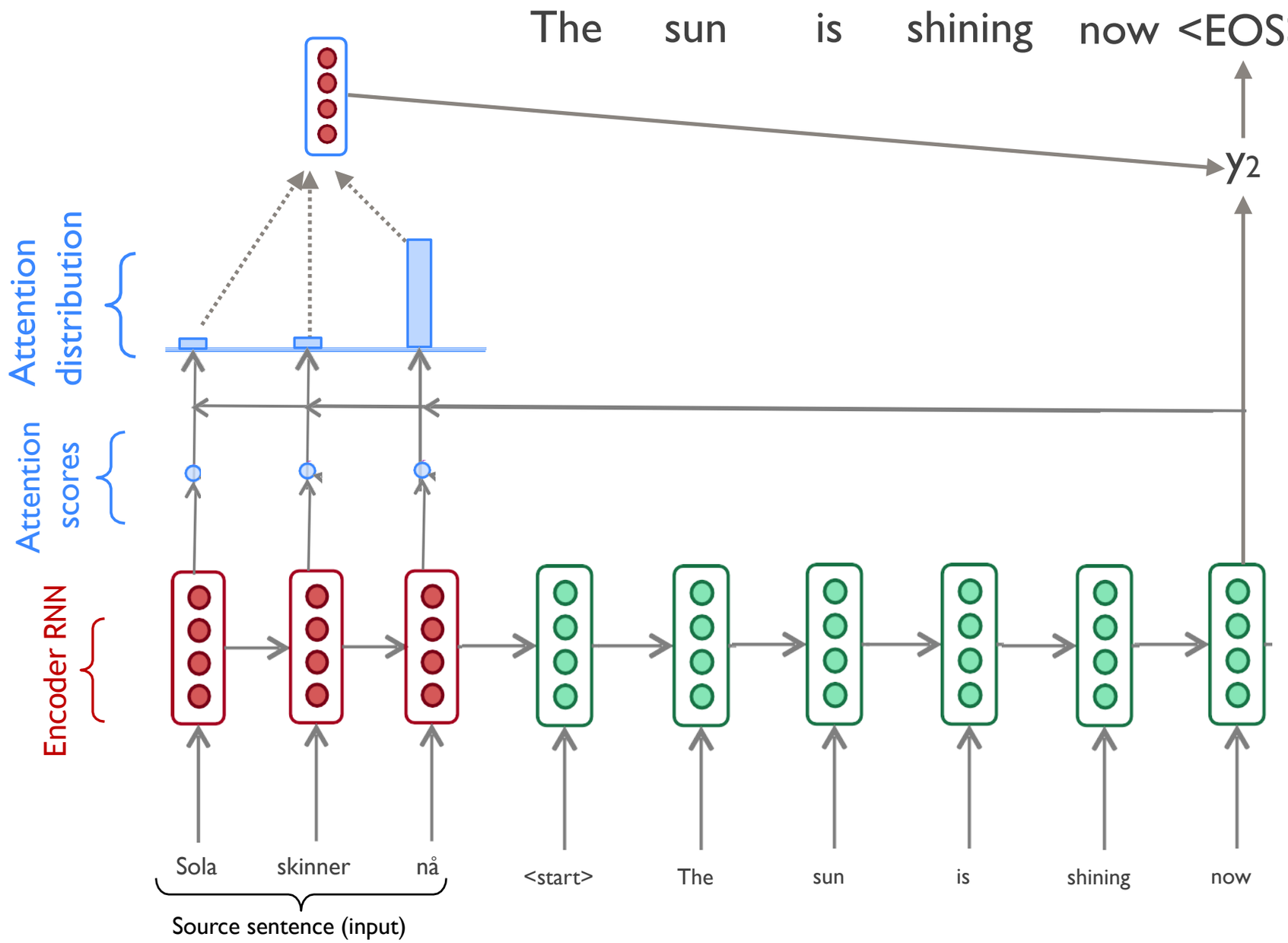


Concatenate attention output with decoder hidden state, then use to compute as before

Seq2Seq with Attention



Seq2Seq with Attention



Attention in General Deep Learning

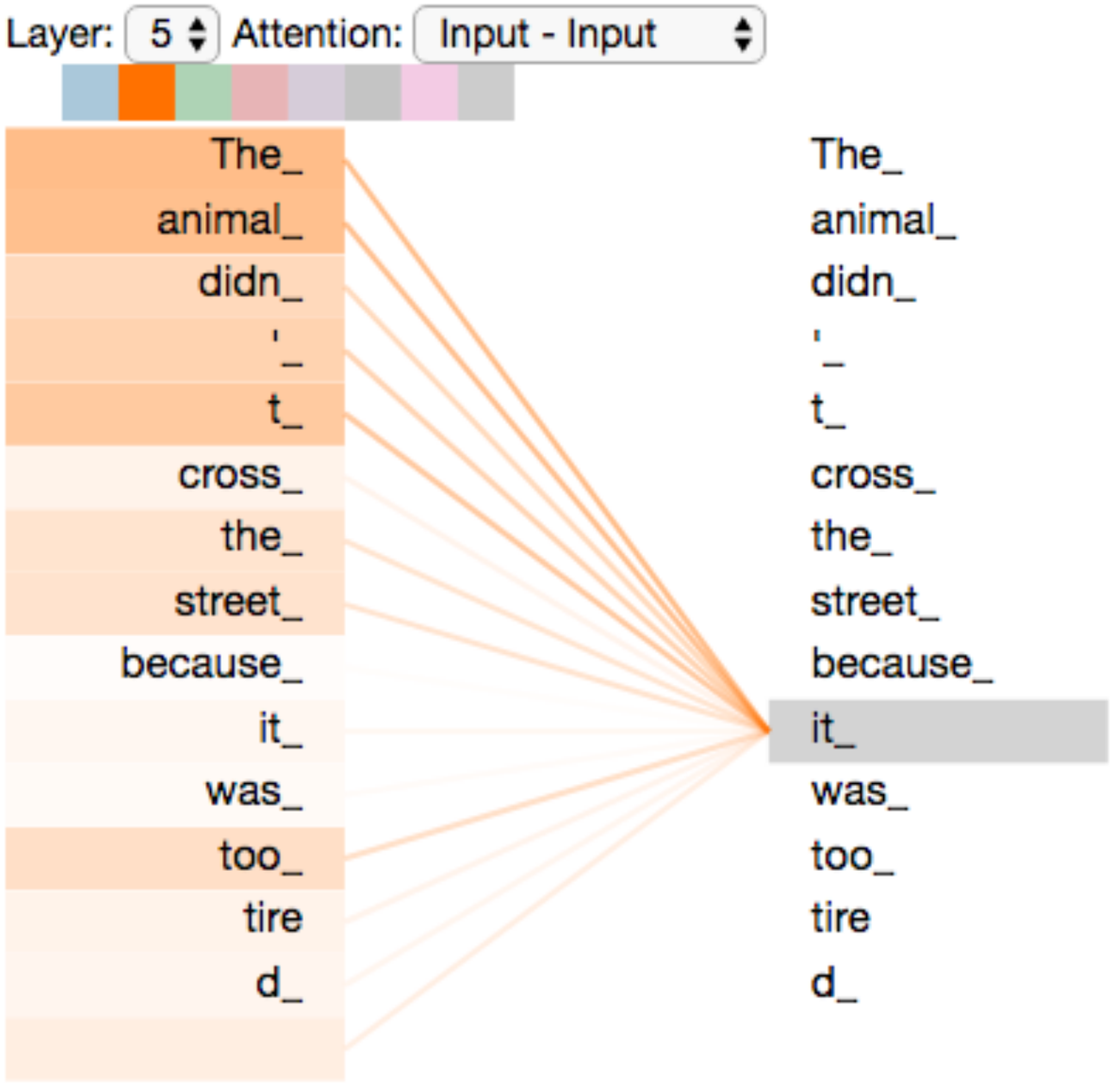
- Attention is not specific to Seq-to-Seq models or NMT tasks
- General Definition

> Given a set of vector values, and a vector query, attention is a technique to compute a weighted sum of the values, dependent on the query.

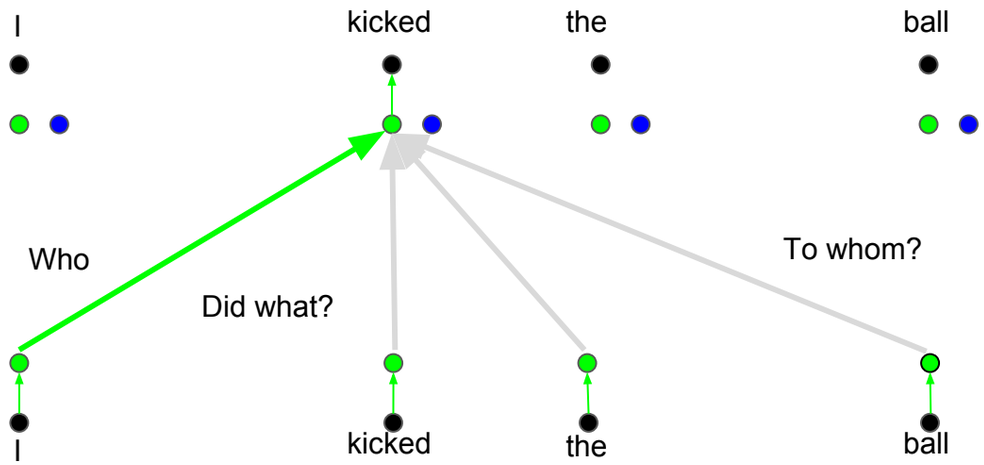
- The query attends to the values
- In Seq2Seq models each decoder hidden state (query) attend to all the encoder hidden state (values)

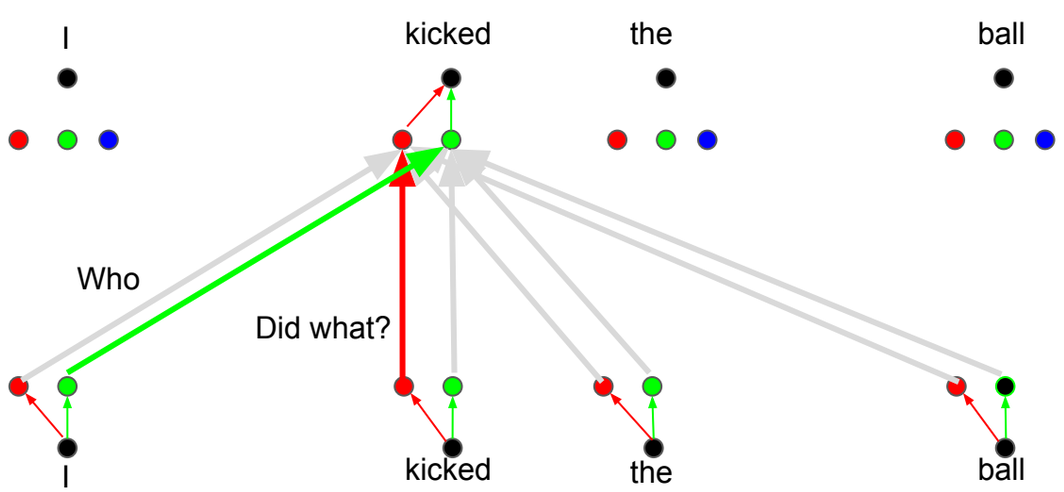
Self-Attention

- ”The animal didn't cross the street because it was too tired”
- What does “it” in this sentence refer to?



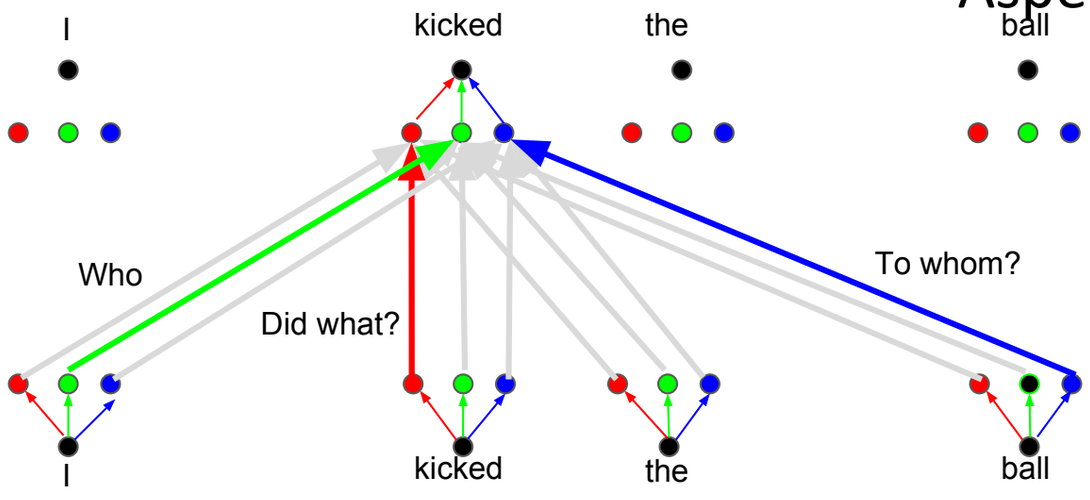
Multi-head Attention





Aspect: “who”

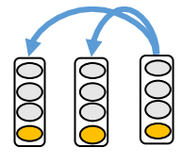
Aspect: “did what”



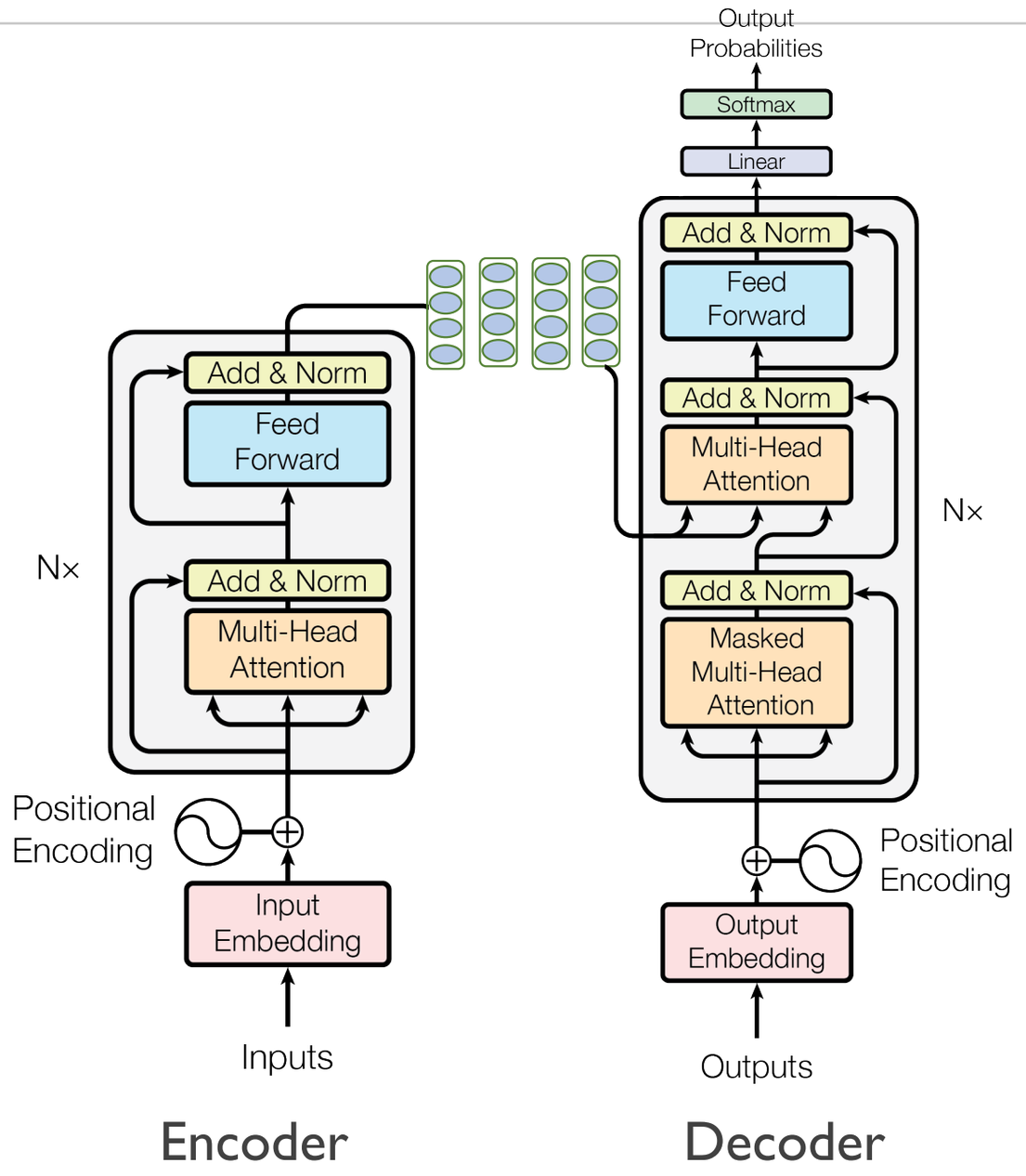
Aspect: “to whom”

Transformer Attention

- Uses three types of attention
  - Encoder-decoder attention
  - Self-attention
  - Masked attention
    - Used in decoder
    - Only attends to the past words



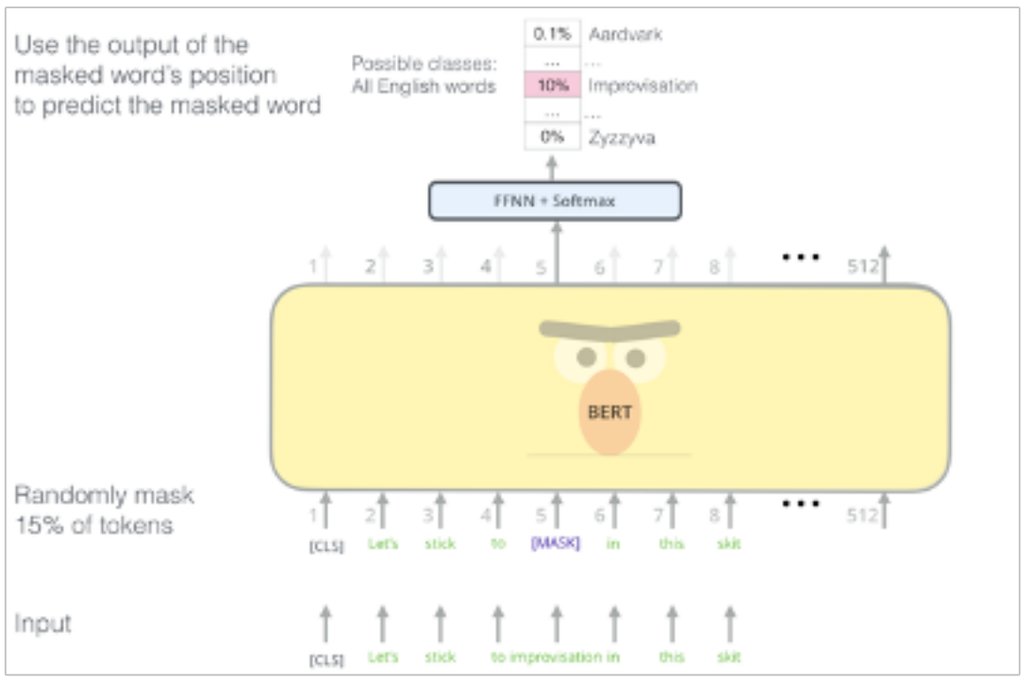
- Incorporates the position information through position encoding



BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

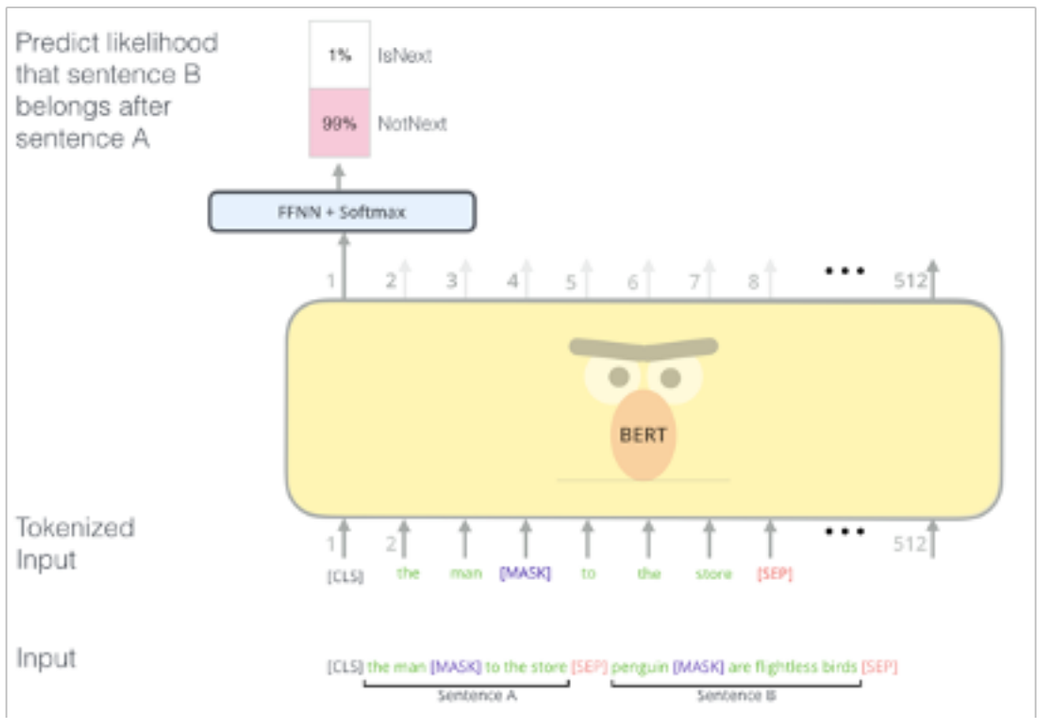
Masked word prediction

- Given a sentence with some words masked at random, can we predict them?
- Randomly select 15% of tokens to be replaced with "<MASK>"



Next sentence prediction

- Given two sentences, does the first follow the second? Teaches BERT about relationship between two sentences
- 50% of the time the actual next sentence, 50% random



References

- Slides credit: Some slides are borrowed from http://web.stanford.edu/class/cs224n/
- http://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/
- http://jalammar.github.io/illustrated-transformer/
- Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. "A neural probabilistic language model." Journal of machine learning research 3, no. Feb (2003): 1137-1155.
- Neural Machine Translation: Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." arXiv preprint arXiv:1508.04025 (2015).
- Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).
- Transformers: Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In Advances in neural information processing systems, pp. 5998-6008. 2017.
- BERT: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).