
# 89688: Statistical Machine Translation

# Neural Machine Translation (contd.)

May 2020

Roee Aharoni
Computer Science Department
Bar Ilan University

Based in part on slides by Kevin Duh and Hermann Ney from the DL4MT winter school

# What is a Language Model?

# What is a Language Model?

- A language model $p(w_1^N)$ measures how likely is the sentence:

$$w_1^N = x_1, x_2, ..., x_N$$

# What is a Language Model?

- A language model $p(w_1^N)$ measures how likely is the sentence:

$$w_1^N = x_1, x_2, ..., x_N$$

- Usually modelled as a product of conditional probabilities:

$$p(x_1, x_2, \ldots, x_N) = \prod_{t=1}^{T} p(x_t \mid x_1, ..., x_{t-1})$$

# What is a Language Model?

- A language model $p(w_1^N)$ measures how likely is the sentence:

$$w_1^N = x_1, x_2, ..., x_N$$

- Usually modelled as a product of conditional probabilities:

$$p(x_1, x_2, \ldots, x_N) = \prod_{t=1}^{T} p(x_t \mid x_1, ..., x_{t-1})$$

- The conventional approach - assume a Markov chain of order n:

$$p(x_1, x_2, \ldots, x_N) = \prod_{t=1}^{T} p(x_t \mid x_{t-n}, ..., x_{t-1})$$

# What is a Language Model?

- A language model $p(w_1^N)$ measures how likely is the sentence:

$$w_1^N = x_1, x_2, ..., x_N$$

- Usually modelled as a product of conditional probabilities:

$$p(x_1, x_2, \ldots, x_N) = \prod_{t=1}^{T} p(x_t \mid x_1, ..., x_{t-1})$$

- The conventional approach - assume a Markov chain of order n:

$$p(x_1, x_2, \ldots, x_N) = \prod_{t=1}^{T} p(x_t \mid x_{t-n}, ..., x_{t-1})$$

- and count: $p(x_t \mid x_{t-n}, \ldots, x_{t-1}) = \frac{count(x_{t-n}, ..., x_{t-1}, x_t)}{count(x_{t-n}, ..., x_{t-1})}$

# What is a Language Model?

| word | unigram | bigram | trigram | 4-gram |
|---|---|---|---|---|
| i | 6.684 | 3.197 | 3.197 | 3.197 |
| would | 8.342 | 2.884 | 2.791 | 2.791 |
| like | 9.129 | 2.026 | 1.031 | 1.290 |
| to | 5.081 | 0.402 | 0.144 | 0.113 |
| commend | 15.487 | 12.335 | 8.794 | 8.633 |
| the | 3.885 | 1.402 | 1.084 | 0.880 |
| rapporteur | 10.840 | 7.319 | 2.763 | 2.350 |
| on | 6.765 | 4.140 | 4.150 | 1.862 |
| his | 10.678 | 7.316 | 2.367 | 1.978 |
| work | 9.993 | 4.816 | 3.498 | 2.394 |
| . | 4.896 | 3.020 | 1.785 | 1.510 |
| </s> | 4.828 | 0.005 | 0.000 | 0.000 |
| average | 8.051 | 4.072 | 2.634 | 2.251 |
| perplexity | 265.136 | 16.817 | 6.206 | 4.758 |

**Perplexity - The lower, the better**

# What is a Language Model?

- Let's compute:

$$p(i, would, like, to, ..., < /s >)$$

| word | unigram | bigram | trigram | 4-gram |
|---|---|---|---|---|
| i | 6.684 | 3.197 | 3.197 | 3.197 |
| would | 8.342 | 2.884 | 2.791 | 2.791 |
| like | 9.129 | 2.026 | 1.031 | 1.290 |
| to | 5.081 | 0.402 | 0.144 | 0.113 |
| commend | 15.487 | 12.335 | 8.794 | 8.633 |
| the | 3.885 | 1.402 | 1.084 | 0.880 |
| rapporteur | 10.840 | 7.319 | 2.763 | 2.350 |
| on | 6.765 | 4.140 | 4.150 | 1.862 |
| his | 10.678 | 7.316 | 2.367 | 1.978 |
| work | 9.993 | 4.816 | 3.498 | 2.394 |
| . | 4.896 | 3.020 | 1.785 | 1.510 |
| </s> | 4.828 | 0.005 | 0.000 | 0.000 |
| average | 8.051 | 4.072 | 2.634 | 2.251 |
| perplexity | 265.136 | 16.817 | 6.206 | 4.758 |

**Perplexity - The lower, the better**

# What is a Language Model?

- Let's compute:

$$p(i, would, like, to, ..., < /s >)$$

- Unigram LM:

$$p(i)p(would)p(like)...p(< /s >)$$

| word | unigram | bigram | trigram | 4-gram |
|---|---|---|---|---|
| i | 6.684 | 3.197 | 3.197 | 3.197 |
| would | 8.342 | 2.884 | 2.791 | 2.791 |
| like | 9.129 | 2.026 | 1.031 | 1.290 |
| to | 5.081 | 0.402 | 0.144 | 0.113 |
| commend | 15.487 | 12.335 | 8.794 | 8.633 |
| the | 3.885 | 1.402 | 1.084 | 0.880 |
| rapporteur | 10.840 | 7.319 | 2.763 | 2.350 |
| on | 6.765 | 4.140 | 4.150 | 1.862 |
| his | 10.678 | 7.316 | 2.367 | 1.978 |
| work | 9.993 | 4.816 | 3.498 | 2.394 |
| . | 4.896 | 3.020 | 1.785 | 1.510 |
| </s> | 4.828 | 0.005 | 0.000 | 0.000 |
| average | 8.051 | 4.072 | 2.634 | 2.251 |
| perplexity | 265.136 | 16.817 | 6.206 | 4.758 |

**Perplexity - The lower, the better**

# What is a Language Model?

- Let's compute:

$$p(i, would, like, to, ..., < /s >)$$

- Unigram LM:

$$p(i)p(would)p(like)...p(< /s >)$$

- Bi-gram LM:

$$p(i)p(would \mid i)p(like \mid would)...p(< /s > \mid .)$$

| word | unigram | bigram | trigram | 4-gram |
|---|---|---|---|---|
| i | 6.684 | 3.197 | 3.197 | 3.197 |
| would | 8.342 | 2.884 | 2.791 | 2.791 |
| like | 9.129 | 2.026 | 1.031 | 1.290 |
| to | 5.081 | 0.402 | 0.144 | 0.113 |
| commend | 15.487 | 12.335 | 8.794 | 8.633 |
| the | 3.885 | 1.402 | 1.084 | 0.880 |
| rapporteur | 10.840 | 7.319 | 2.763 | 2.350 |
| on | 6.765 | 4.140 | 4.150 | 1.862 |
| his | 10.678 | 7.316 | 2.367 | 1.978 |
| work | 9.993 | 4.816 | 3.498 | 2.394 |
| . | 4.896 | 3.020 | 1.785 | 1.510 |
| </s> | 4.828 | 0.005 | 0.000 | 0.000 |
| average | 8.051 | 4.072 | 2.634 | 2.251 |
| perplexity | 265.136 | 16.817 | 6.206 | 4.758 |

**Perplexity - The lower, the better**

# What is a Language Model?

- Let's compute:

$$p(i, would, like, to, ..., < /s >)$$

- Unigram LM:

$$p(i)p(would)p(like)...p(< /s >)$$

- Bi-gram LM:

$$p(i)p(would \mid i)p(like \mid would)...p(< /s > \mid .)$$

- Tri-gram LM:

$$p(i)p(would \mid i)p(like \mid i, would)...p(< /s > \mid work, .)$$

| word | unigram | bigram | trigram | 4-gram |
|---|---|---|---|---|
| i | 6.684 | 3.197 | 3.197 | 3.197 |
| would | 8.342 | 2.884 | 2.791 | 2.791 |
| like | 9.129 | 2.026 | 1.031 | 1.290 |
| to | 5.081 | 0.402 | 0.144 | 0.113 |
| commend | 15.487 | 12.335 | 8.794 | 8.633 |
| the | 3.885 | 1.402 | 1.084 | 0.880 |
| rapporteur | 10.840 | 7.319 | 2.763 | 2.350 |
| on | 6.765 | 4.140 | 4.150 | 1.862 |
| his | 10.678 | 7.316 | 2.367 | 1.978 |
| work | 9.993 | 4.816 | 3.498 | 2.394 |
| . | 4.896 | 3.020 | 1.785 | 1.510 |
| </s> | 4.828 | 0.005 | 0.000 | 0.000 |
| average | 8.051 | 4.072 | 2.634 | 2.251 |
| perplexity | 265.136 | 16.817 | 6.206 | 4.758 |

**Perplexity - The lower, the better**

# A Note on Perplexity

# A Note on Perplexity

- Perplexity is the inverse probability of the unseen test set, normalised by the number of words:

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

# A Note on Perplexity

- Perplexity is the inverse probability of the unseen test set, normalised by the number of words:

$$PP(W) = P(w_1 w_2 ... w_N)^{-\frac{1}{N}}$$

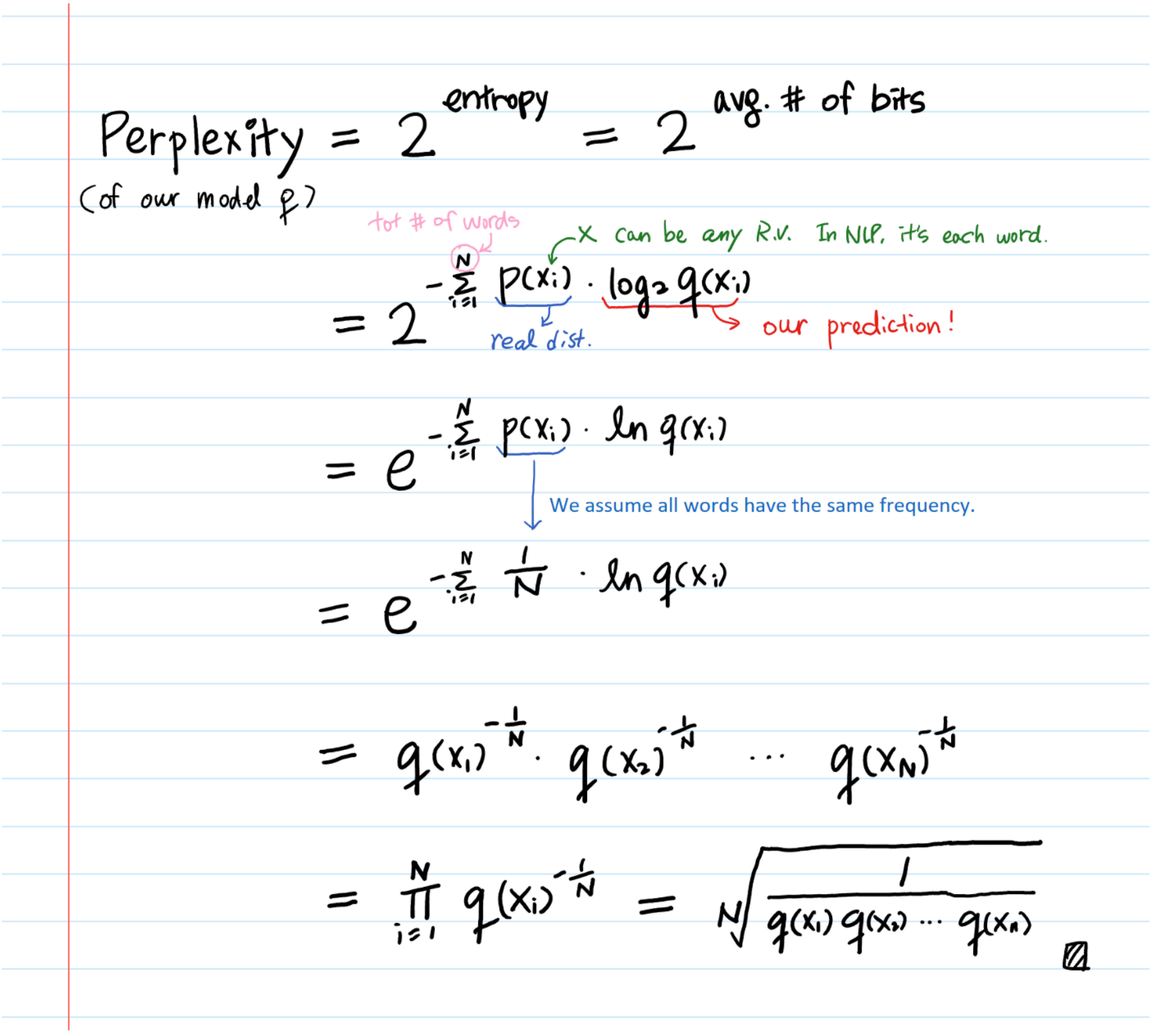
- Can be seen as an exponentiation of the entropy:

$$2^{H(p)} = 2^{-\sum_x p(x) \log_2 p(x)}$$

# A Note on Perplexity

from a blog post by Aerin Kim

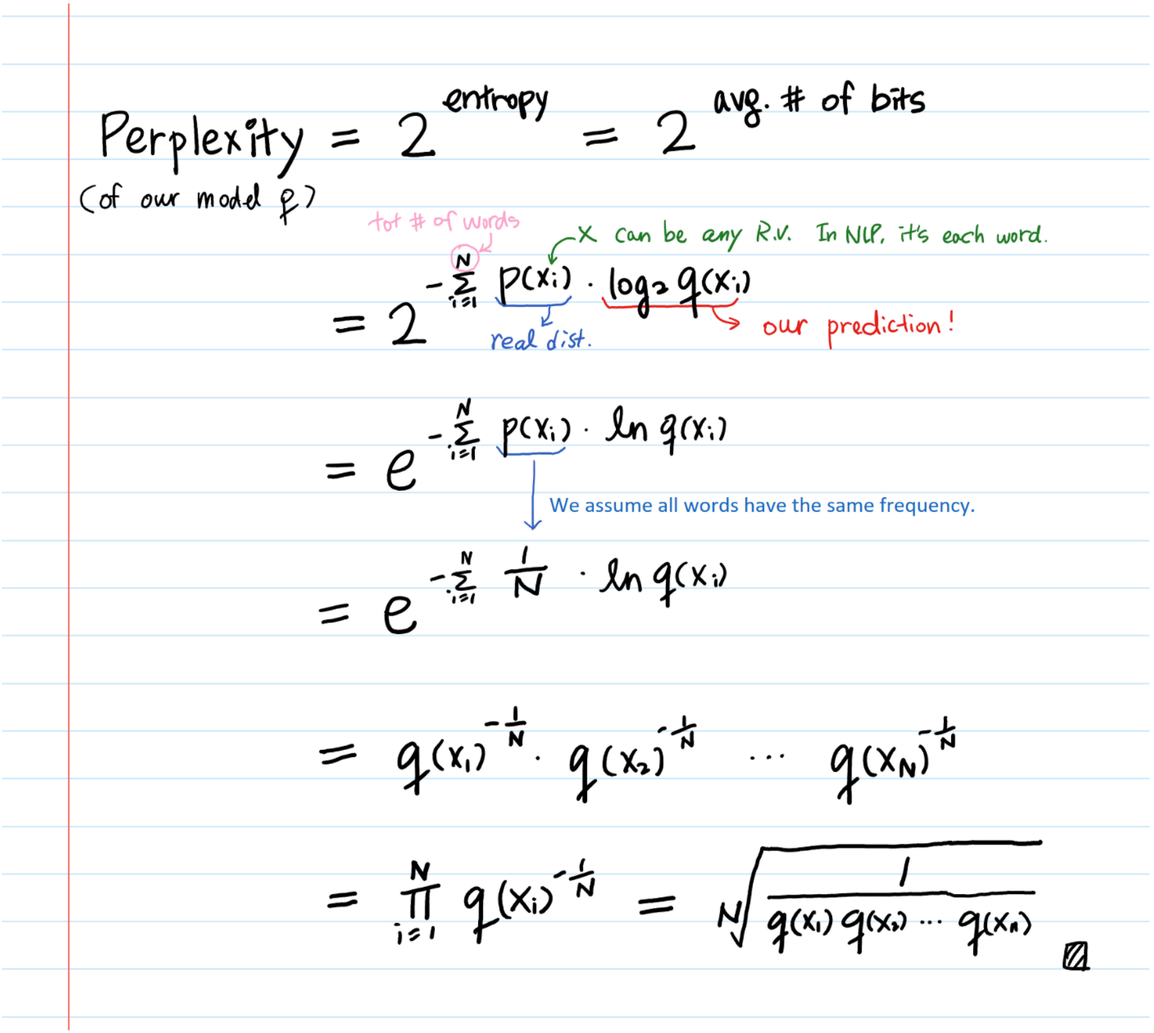
- Perplexity is the inverse probability of the unseen test set, normalised by the number of words:
- Can be seen as an exponentiation of the entropy:



# A Note on Perplexity

from a blog post by Aerin Kim

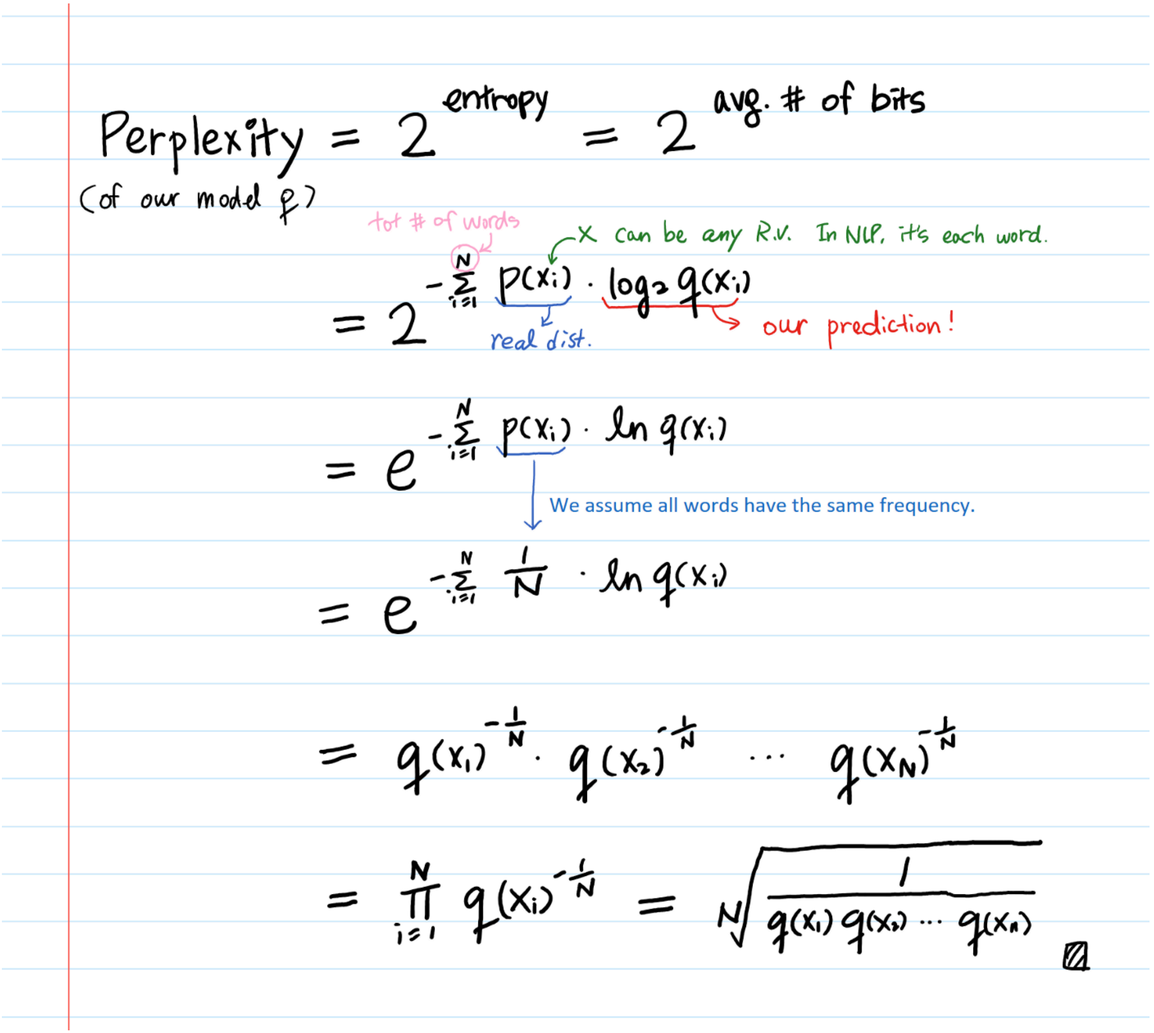
- Perplexity is the inverse probability of the unseen test set, normalised by the number of words:
- Can be seen as an exponentiation of the entropy:
- Lower perplexity means lower entropy which means **less uncertainty**



# A Note on Perplexity

from a blog post by Aerin Kim

- Perplexity is the inverse probability of the unseen test set, normalised by the number of words:
- Can be seen as an exponentiation of the entropy:
- Lower perplexity means lower entropy which means **less uncertainty**
  - So lower = better



# The Conventional Approach: Issues

# The Conventional Approach: Issues

- Data sparsity - many n-grams do not appear in the training data

# The Conventional Approach: Issues

- Data sparsity - many n-grams do not appear in the training data
  - Can be handled by smoothing, back-off

# The Conventional Approach: Issues

- Data sparsity - many n-grams do not appear in the training data
  - Can be handled by smoothing, back-off
- Lack of generalization

# The Conventional Approach: Issues

- Data sparsity - many n-grams do not appear in the training data
  - Can be handled by smoothing, back-off
- Lack of generalization
  - “chases a cat”, “chases a dog”…

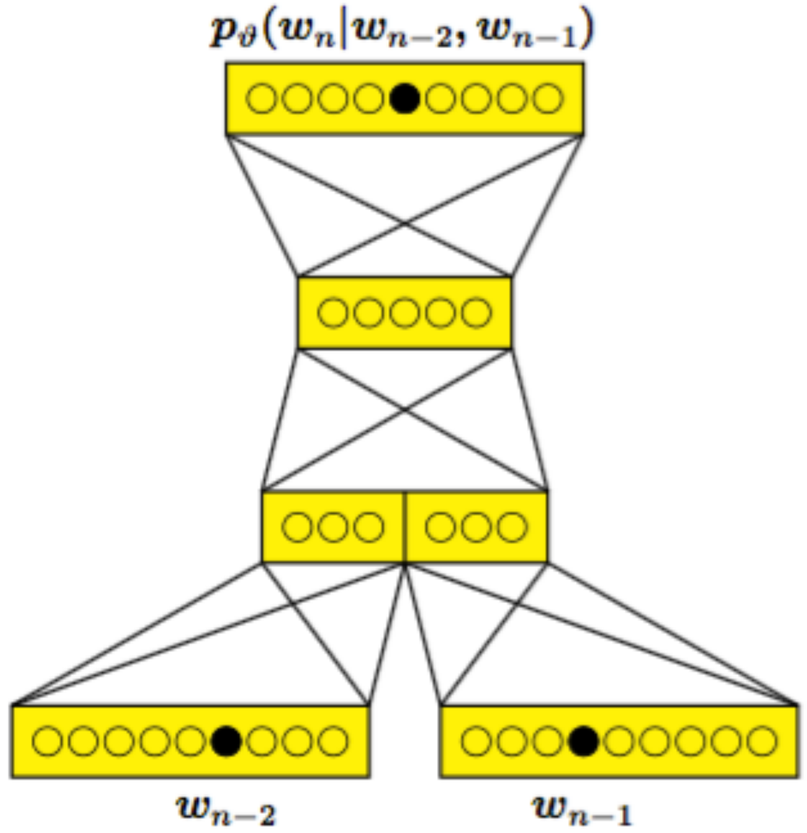
# The Conventional Approach: Issues

- Data sparsity - many n-grams do not appear in the training data
  - Can be handled by smoothing, back-off
- Lack of generalization
  - “chases a cat”, “chases a dog”…
  - “chases an ostrich”?

# The Conventional Approach: Issues

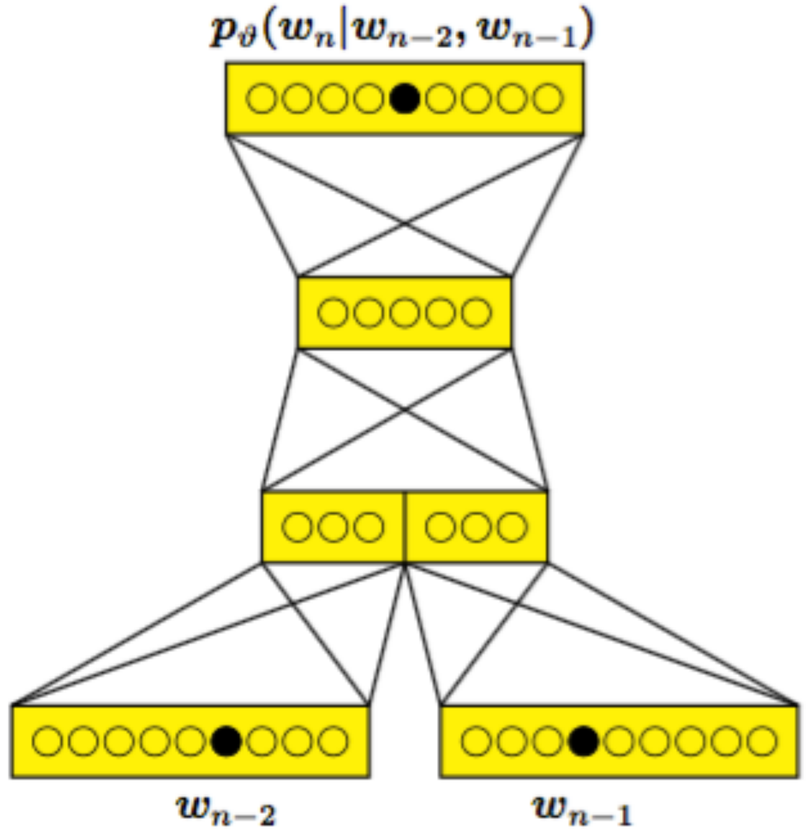
- Data sparsity - many n-grams do not appear in the training data
  - Can be handled by smoothing, back-off
- Lack of generalization
  - “chases a cat”, “chases a dog”…
  - “chases an ostrich”?

# Language Modelling with an MLP



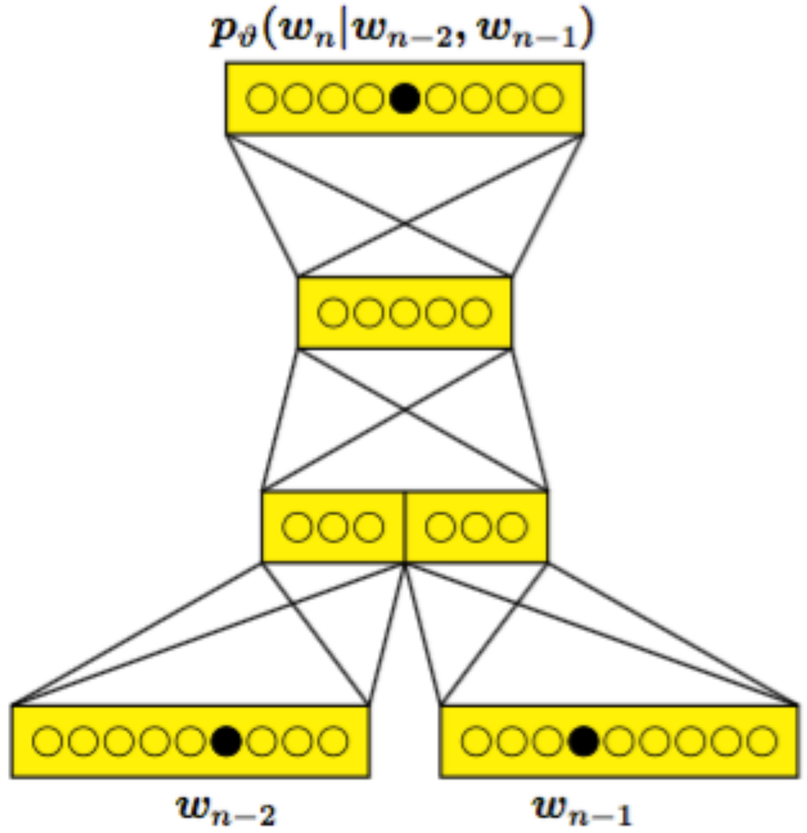
# Language Modelling with an MLP

- Start with one-hot encoding of each word



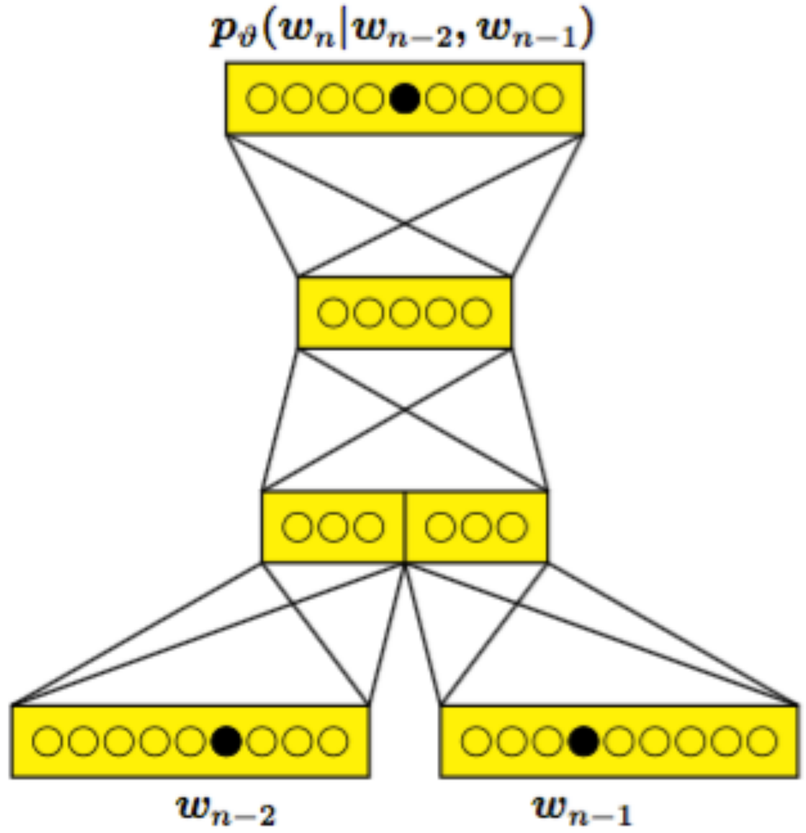
# Language Modelling with an MLP

- Start with one-hot encoding of each word
- Learn word representations in a continuous space



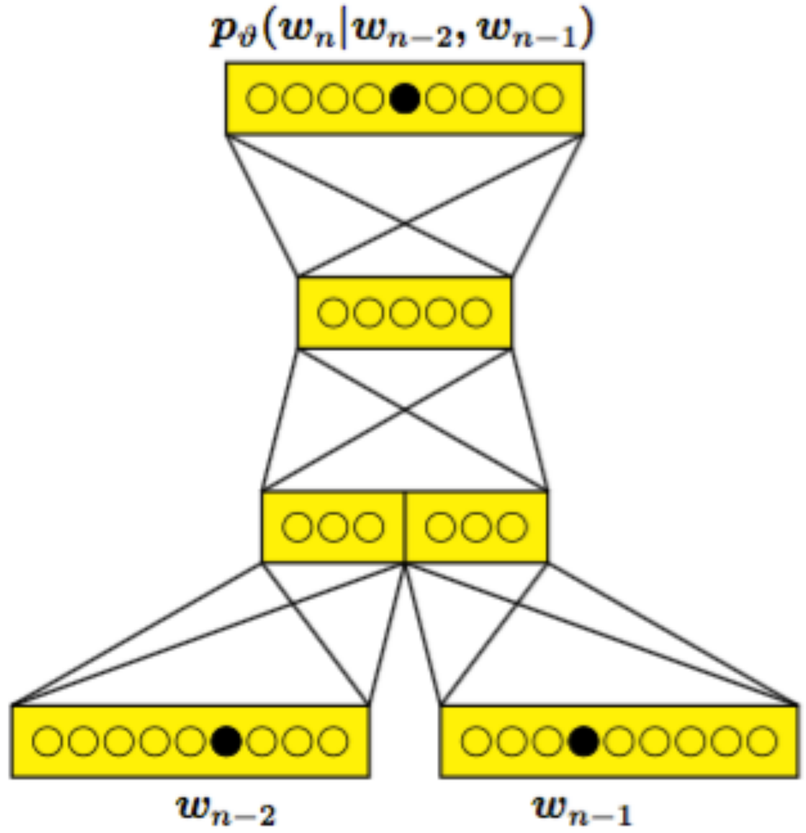
# Language Modelling with an MLP

- Start with one-hot encoding of each word
- Learn word representations in a continuous space
- Hidden layer using a non-linear activation

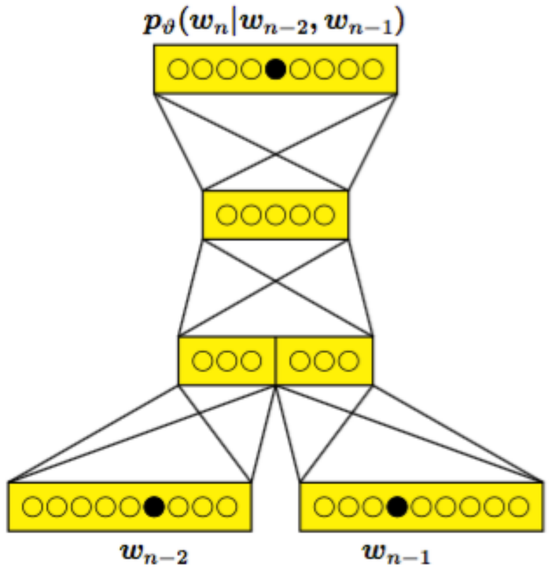


# Language Modelling with an MLP

- Start with one-hot encoding of each word
- Learn word representations in a continuous space
- Hidden layer using a non-linear activation
- Output probabilities using softmax

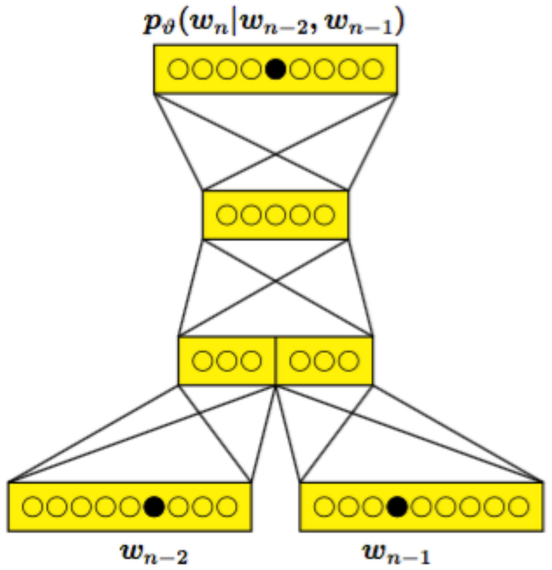


# Language Modelling with an MLP



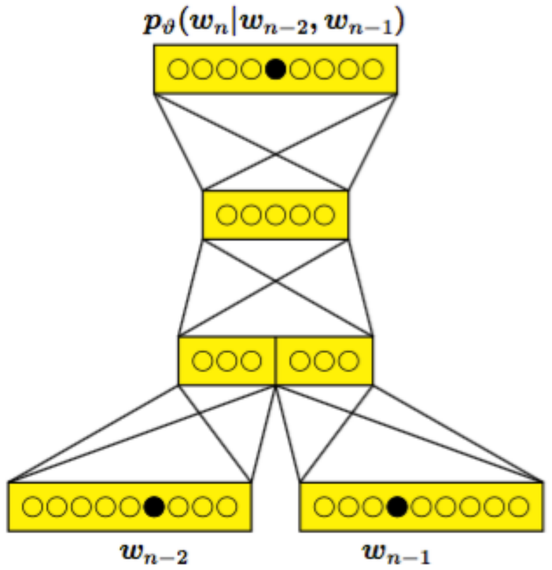
# Language Modelling with an MLP

- Experiment details (Sundermeyer, Ney & Schülter, 2015):



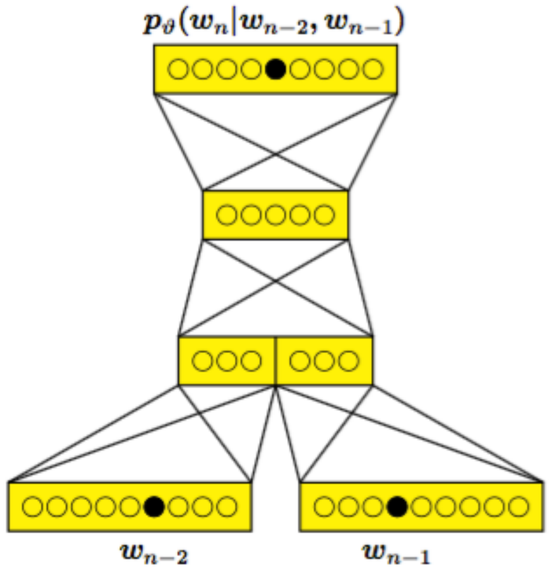
# Language Modelling with an MLP

- Experiment details (Sundermeyer, Ney & Schülter, 2015):
  - vocabulary size: 128k words



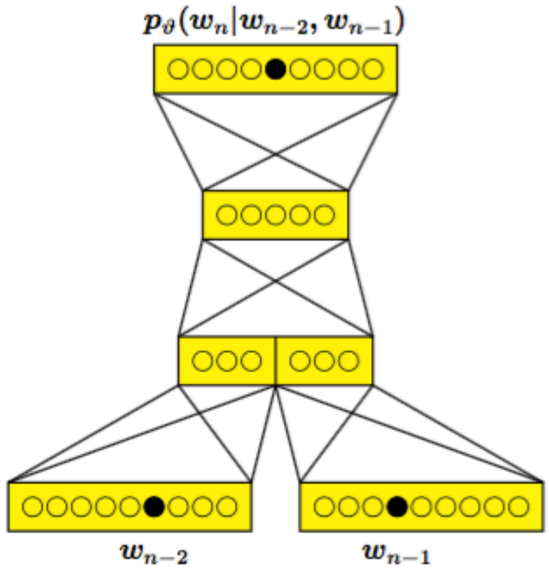
# Language Modelling with an MLP

- Experiment details (Sundermeyer, Ney & Schülter, 2015):
  - vocabulary size: 128k words
  - training text: 50M words



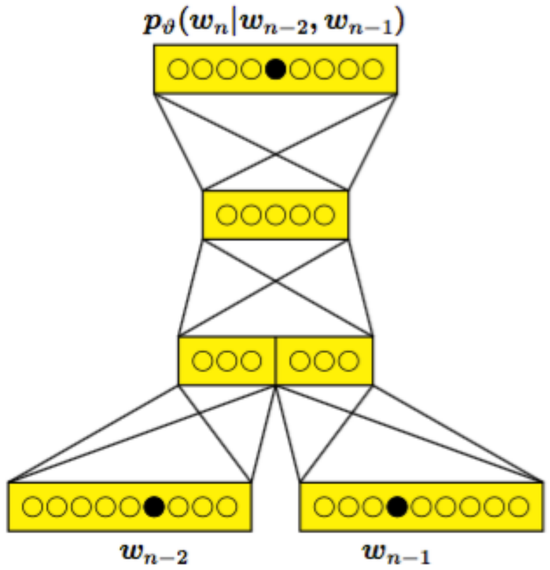
# Language Modelling with an MLP

- Experiment details (Sundermeyer, Ney & Schülter, 2015):
  - vocabulary size: 128k words
  - training text: 50M words
  - development corpus: 39k words



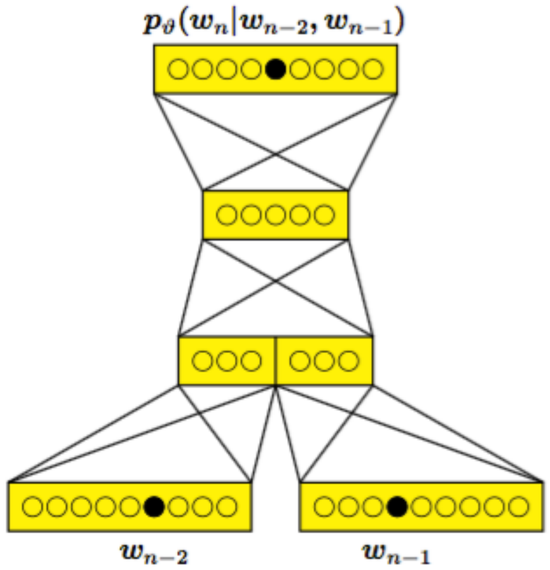
# Language Modelling with an MLP

- Experiment details (Sundermeyer, Ney & Schülter, 2015):
  - vocabulary size: 128k words
  - training text: 50M words
  - development corpus: 39k words
  - evaluation corpus: 35k words



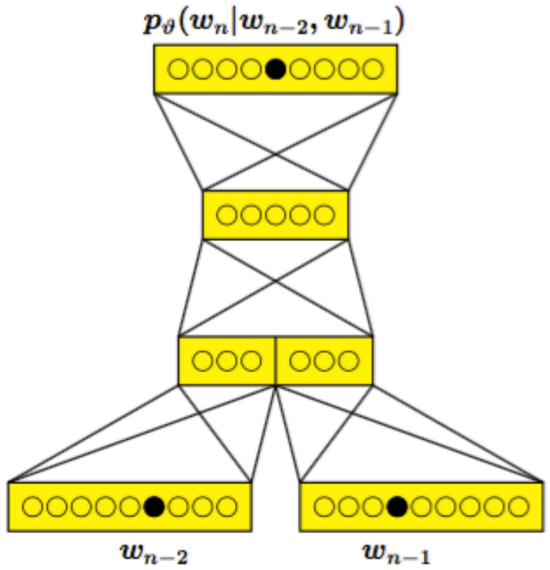
# Language Modelling with an MLP

- Experiment details (Sundermeyer, Ney & Schülter, 2015):
  - vocabulary size: 128k words
  - training text: 50M words
  - development corpus: 39k words
  - evaluation corpus: 35k words
- Network structure:



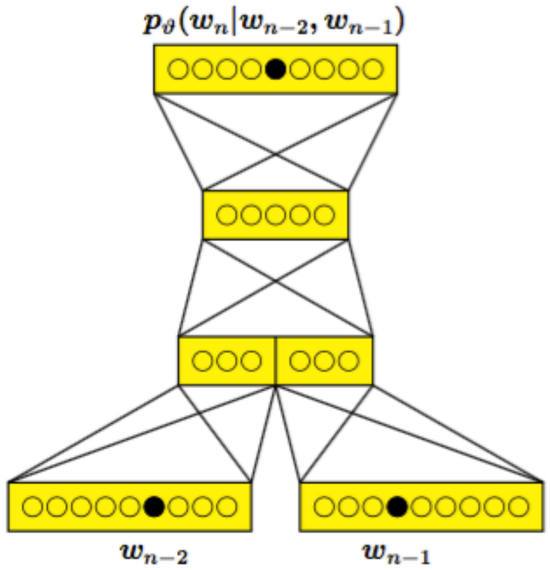
# Language Modelling with an MLP

- Experiment details (Sundermeyer, Ney & Schülter, 2015):
  - vocabulary size: 128k words
  - training text: 50M words
  - development corpus: 39k words
  - evaluation corpus: 35k words
- Network structure:
  - projection layer: 300 nodes (per word)



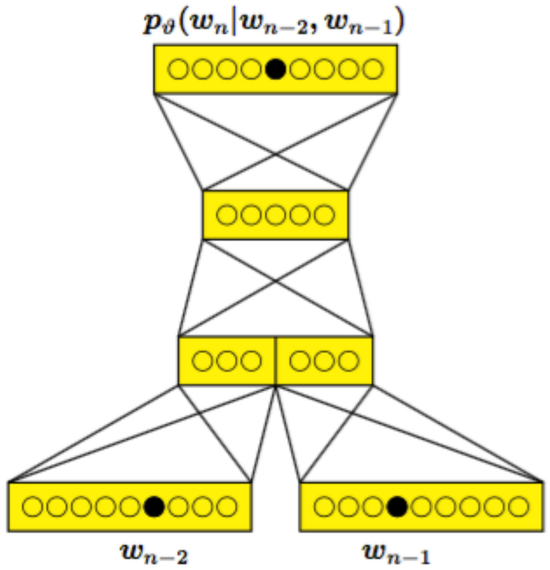
# Language Modelling with an MLP

- Experiment details (Sundermeyer, Ney & Schülter, 2015):
  - vocabulary size: 128k words
  - training text: 50M words
  - development corpus: 39k words
  - evaluation corpus: 35k words
- Network structure:
  - projection layer: 300 nodes (per word)
  - hidden layer: 600 nodes



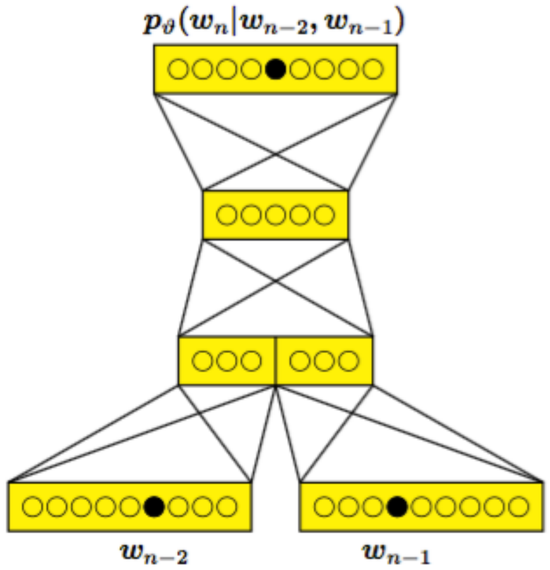
# Language Modelling with an MLP

- Experiment details (Sundermeyer, Ney & Schülter, 2015):
  - vocabulary size: 128k words
  - training text: 50M words
  - development corpus: 39k words
  - evaluation corpus: 35k words
- Network structure:
  - projection layer: 300 nodes (per word)
  - hidden layer: 600 nodes
  - total number of params: $128k \cdot 300 + 600 \cdot 128k = 115M$



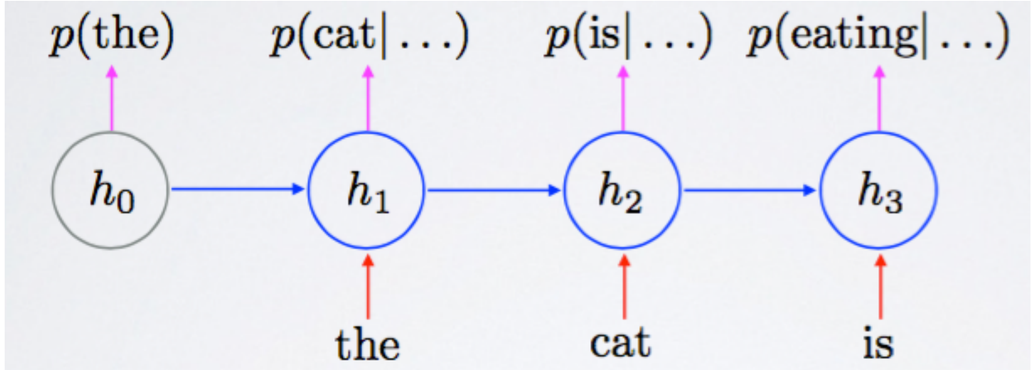
# Language Modelling with an MLP

- Experiment details (Sundermeyer, Ney & Schülter, 2015):
  - vocabulary size: 128k words
  - training text: 50M words
  - development corpus: 39k words
  - evaluation corpus: 35k words
- Network structure:
  - projection layer: 300 nodes (per word)
  - hidden layer: 600 nodes
  - total number of params: $128k \cdot 300 + 600 \cdot 128k = 115M$



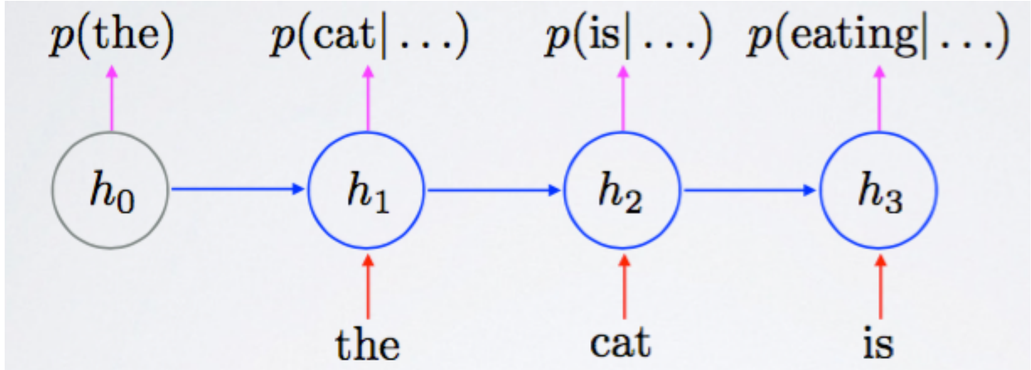

| Approach | PPL |
|---|---|
| 4-gram count model | 163.7 |
| 10-gram MLP | 136.5 |
| 10-gram MLP with 2 layers | 130.9 |

# Language Modelling with RNNs



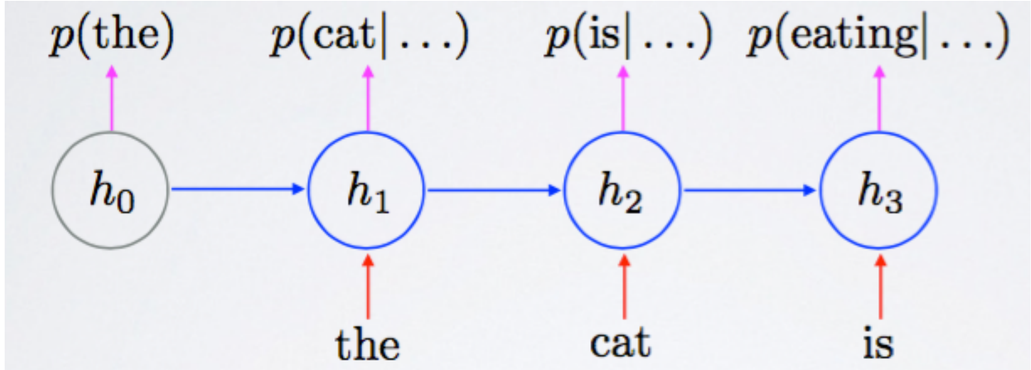
# Language Modelling with RNNs

- MLP LM's are still limited in history (use n-gram assumption)



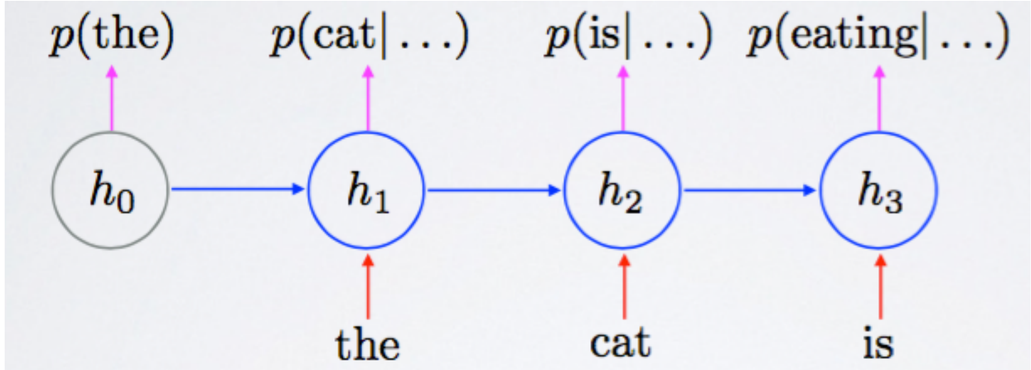
# Language Modelling with RNNs

- MLP LM's are still limited in history (use n-gram assumption)
- We would like to use RNN's to model the entire sentence "at once"



# Language Modelling with RNNs

- MLP LM's are still limited in history (use n-gram assumption)
- We would like to use RNN's to model the entire sentence "at once"
- Every input is a 1-hot vector, every output is the LM probabilities as softmax



# Language Modelling with RNNs

# Language Modelling with RNNs

- RNN’s provide significant improvements over previous models

| Approach | PPL |
|---|---|
| count model | 163.7 |
| 10-gram MLP | 136.5 |
| RNN | 125.2 |
| LSTM-RNN | 107.8 |
| 10-gram MLP with 2 layers | 130.9 |
| LSTM-RNN with 2 layers | 100.5 |

# Language Modelling with RNNs

- RNN's provide significant improvements over previous models
- A price to pay: longer training time

| Approach | PPL |
|---|---|
| count model | 163.7 |
| 10-gram MLP | 136.5 |
| RNN | 125.2 |
| LSTM-RNN | 107.8 |
| 10-gram MLP with 2 layers | 130.9 |
| LSTM-RNN with 2 layers | 100.5 |

| Models | PPL | CPU Time (Order) |
|---|---|---|
| Count model | 163.7 | 30 min |
| MLP | 136.5 | 1 week |
| LSTM-RNN | 107.8 | 3 weeks |

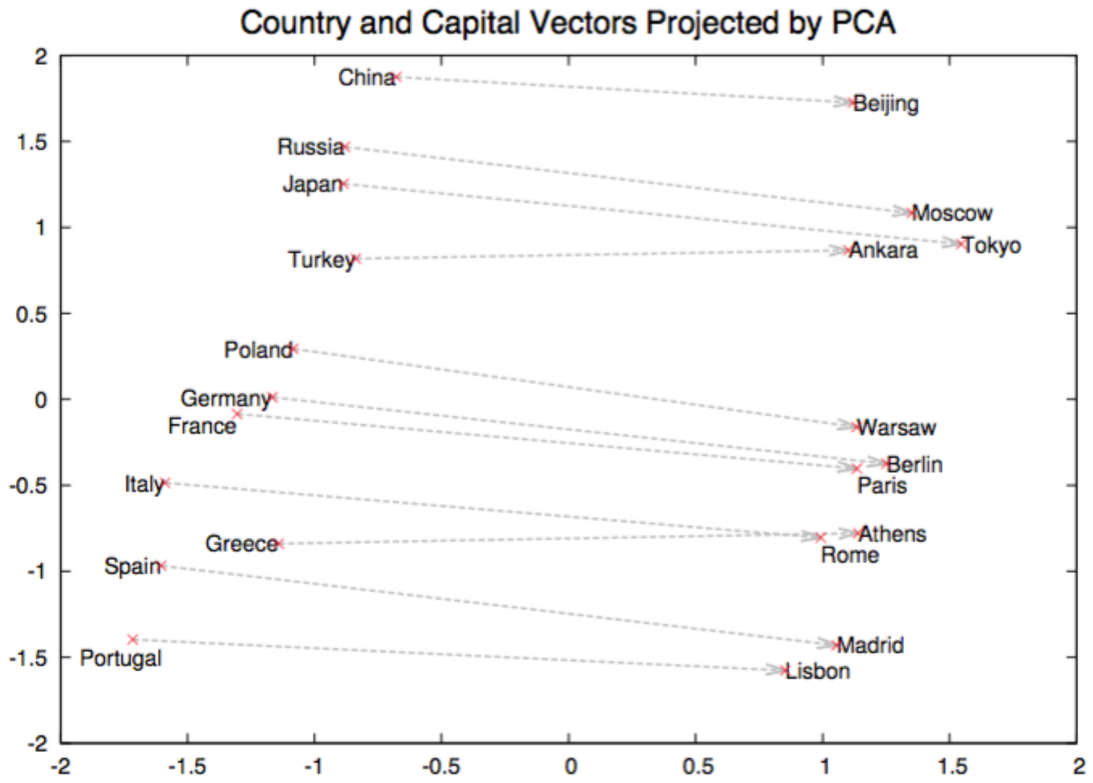
# Distributed Word Representations using word2vec

# Distributed Word Representations using word2vec

- Continuous word representations are learned for each word during network training (a.k.a “word embeddings”)

# Distributed Word Representations using word2vec

- Continuous word representations are learned for each word during network training (a.k.a “word embeddings”)
- These representations can be useful for various tasks like word similarity and word analogies:



# Distributed Word Representations using word2vec

# Distributed Word Representations using word2vec

- word2vec (mikolov et al. 2003) introduced two similar models:

# Distributed Word Representations using word2vec

- word2vec (mikolov et al. 2003) introduced two similar models:

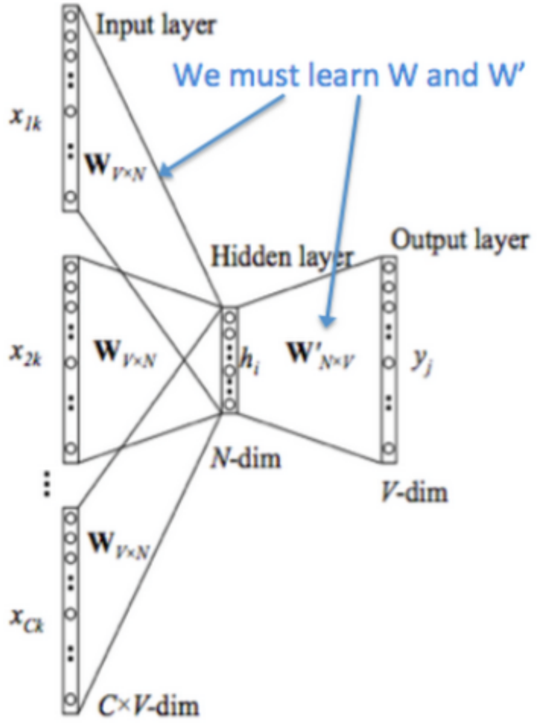



Figure 1: This image demonstrates how CBOW works and how we must learn the transfer matrices

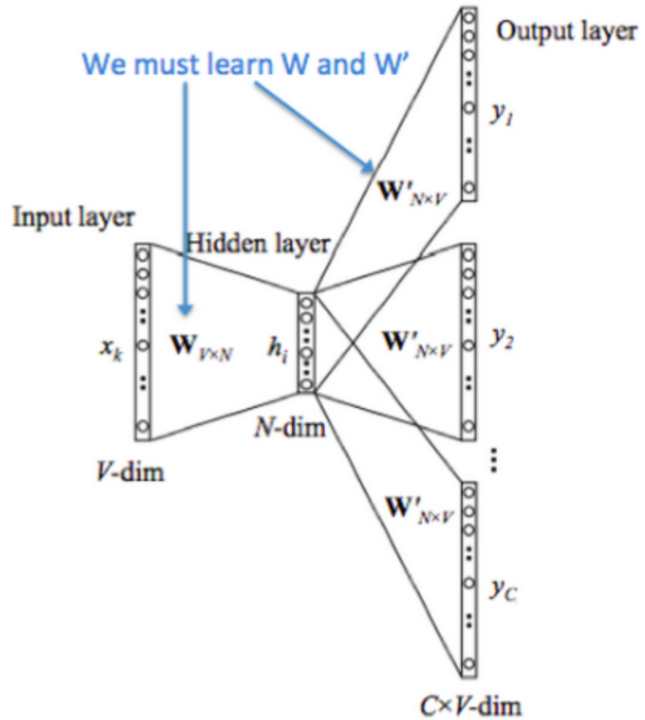



Figure 2: This image demonstrates how Skip-Gram works and how we must learn the transfer matrices

# Distributed Word Representations using word2vec

- word2vec (mikolov et al. 2003) introduced two similar models:
- CBOW (left) and skip-gram (right), both can be seen as MLP's

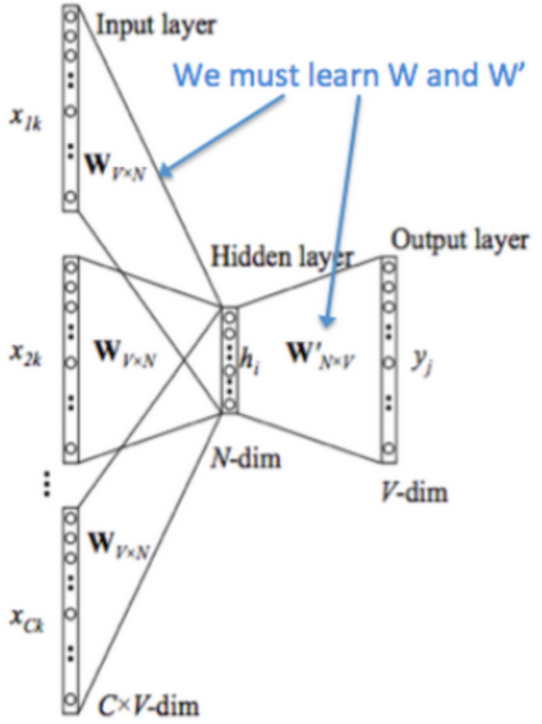



Figure 1: This image demonstrates how CBOW works and how we must learn the transfer matrices

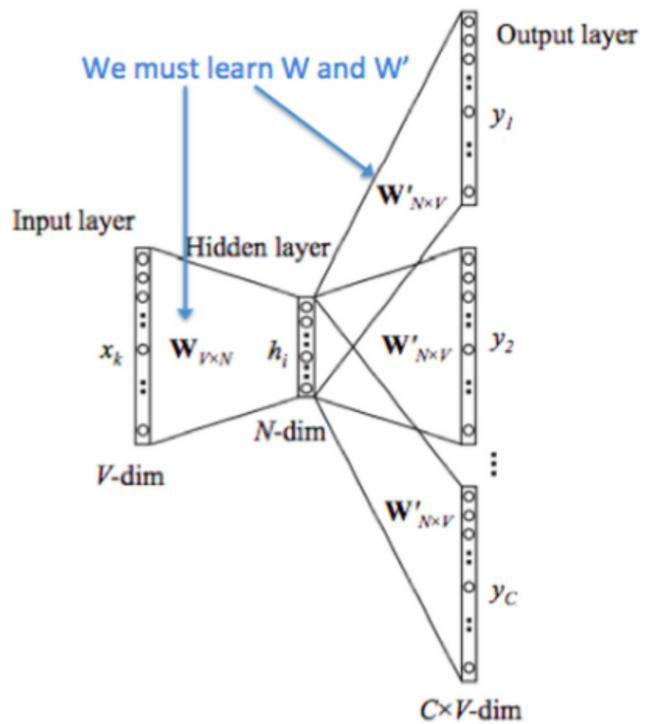



Figure 2: This image demonstrates how Skip-Gram works and how we must learn the transfer matrices

# Distributed Word Representations using word2vec

- word2vec (mikolov et al. 2003) introduced two similar models:
- CBOW (left) and skip-gram (right), both can be seen as MLP's
- Have been shown to approximate the PMI matrix (Levy & Goldberg, 2015)

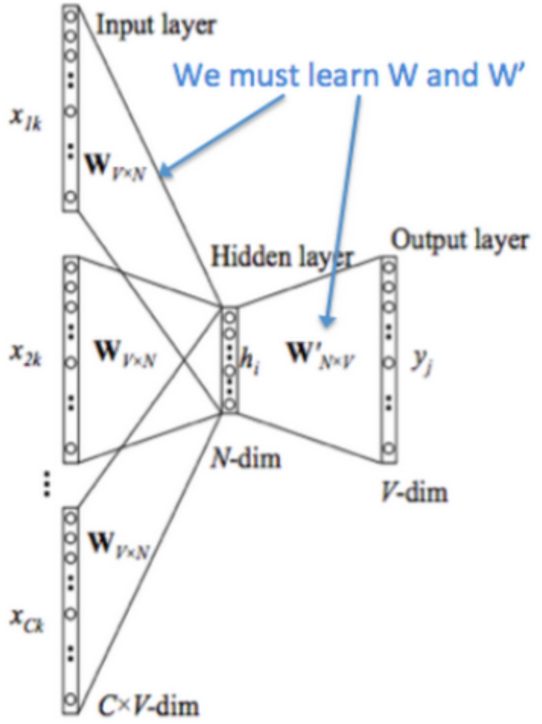



Figure 1: This image demonstrates how CBOW works and how we must learn the transfer matrices

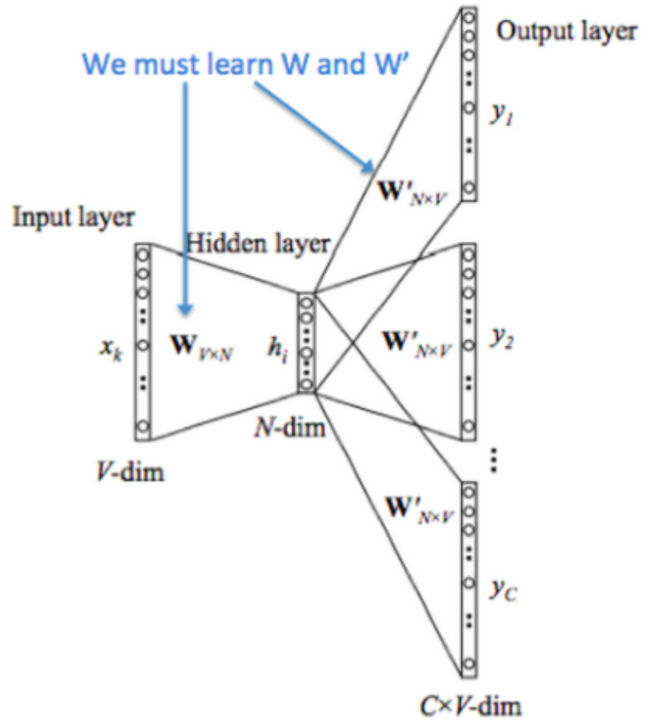
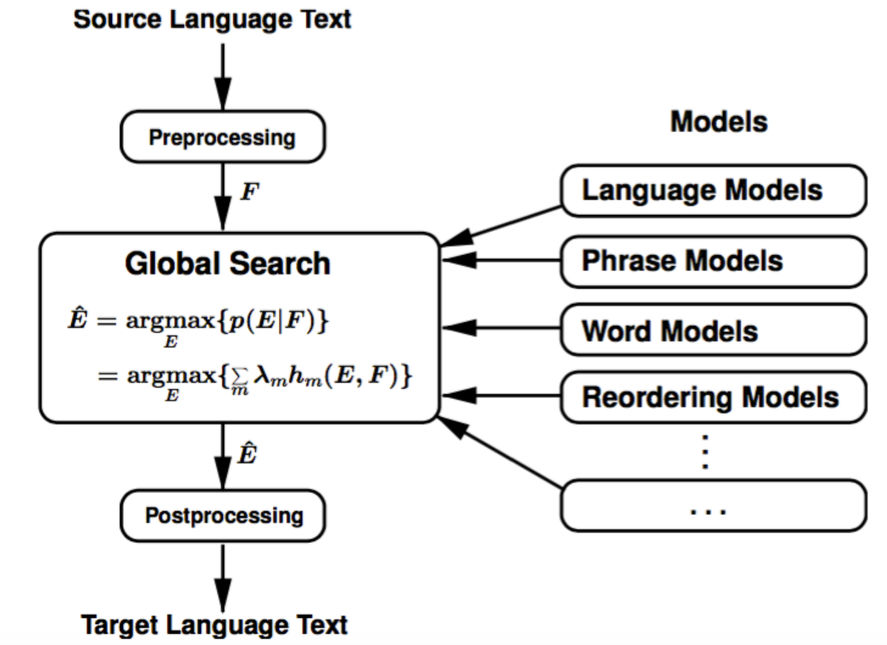



Figure 2: This image demonstrates how Skip-Gram works and how we must learn the transfer matrices

# MLPs for Machine Translation - “Hybrid” SMT

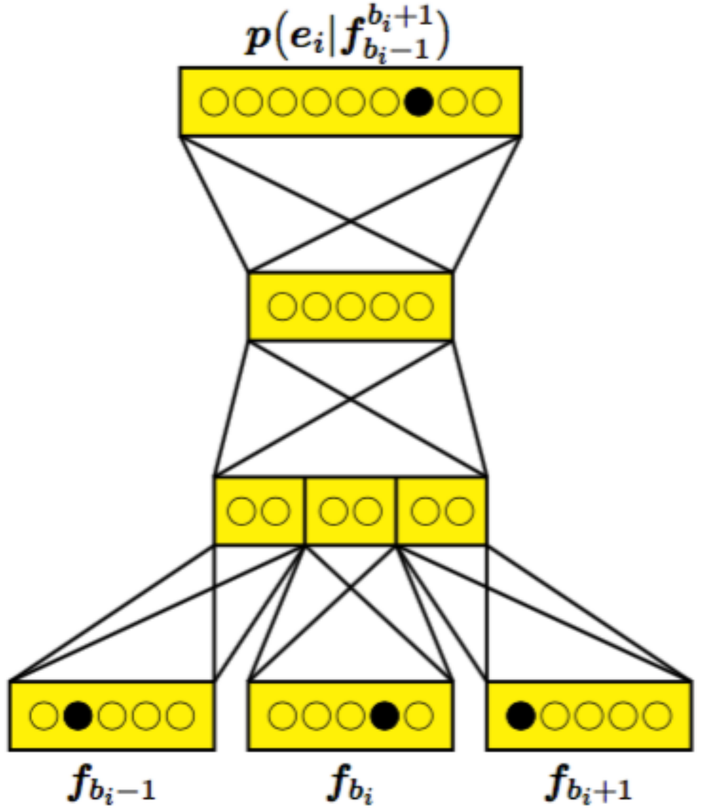
# MLPs for Machine Translation - “Hybrid” SMT

- Reminder - Statistical MT



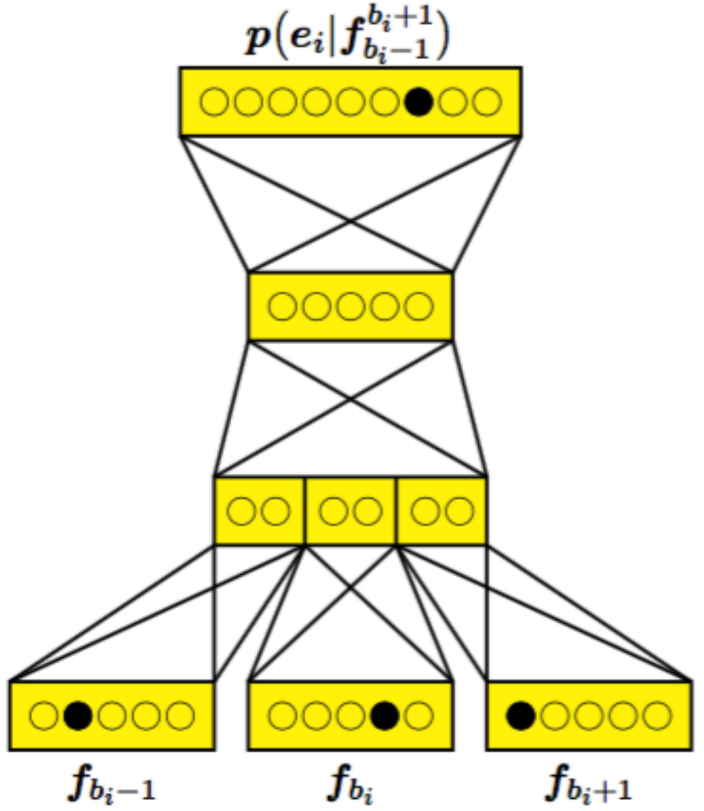
# MLPs for Machine Translation - “Hybrid” SMT

- Reminder - Statistical MT
- Idea - use an MLP to train a translation model



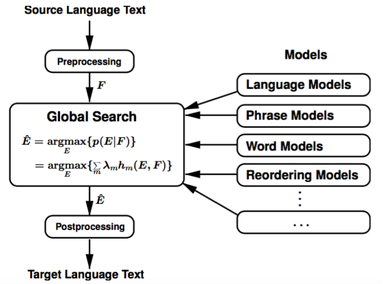
# MLPs for Machine Translation - “Hybrid” SMT

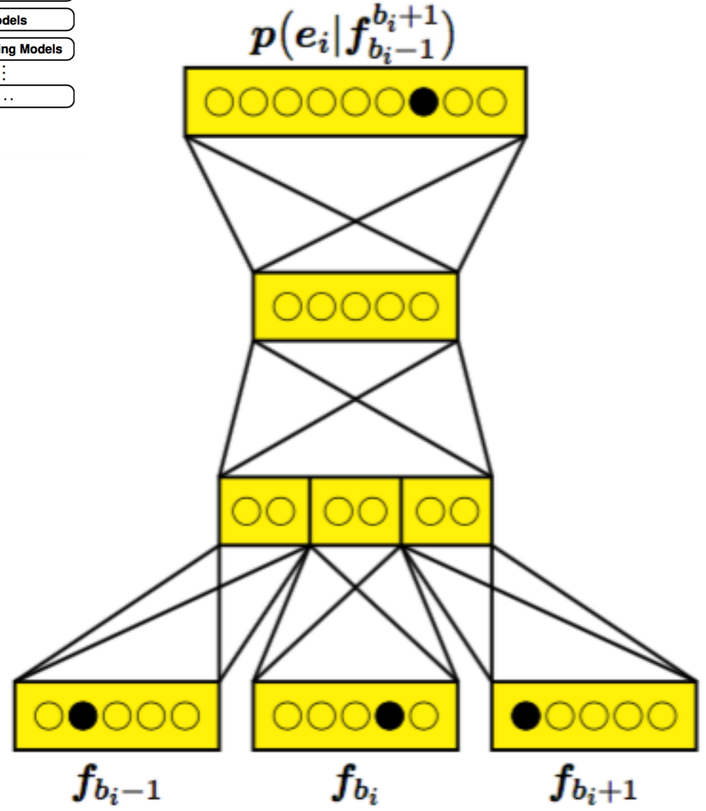
- Reminder - Statistical MT
- Idea - use an MLP to train a translation model
- Inputs are 1-hot encodings of the words in the aligned source language window



# MLPs for Machine Translation - “Hybrid” SMT

- Reminder - Statistical MT
- Idea - use an MLP to train a translation model
- Inputs are 1-hot encodings of the words in the aligned source language window
- Combine this model with the rest while decoding or rescoring

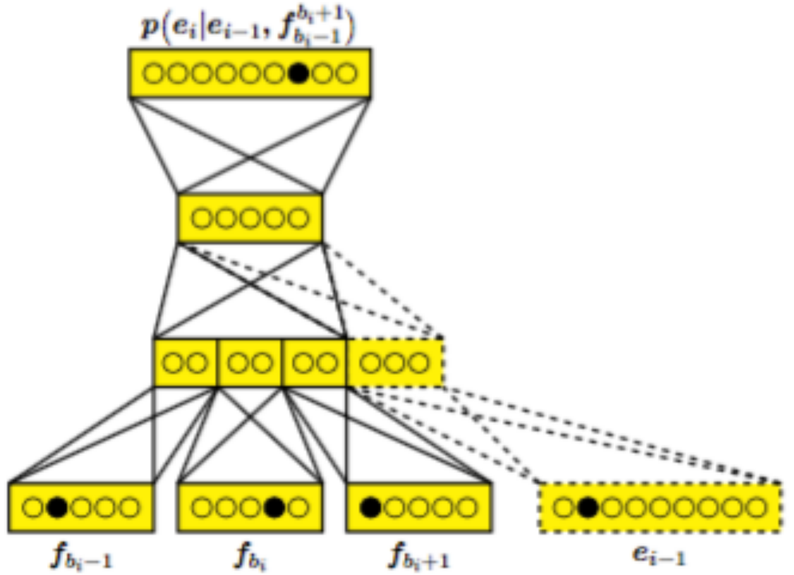


# MLPs for Machine Translation - “Hybrid” SMT

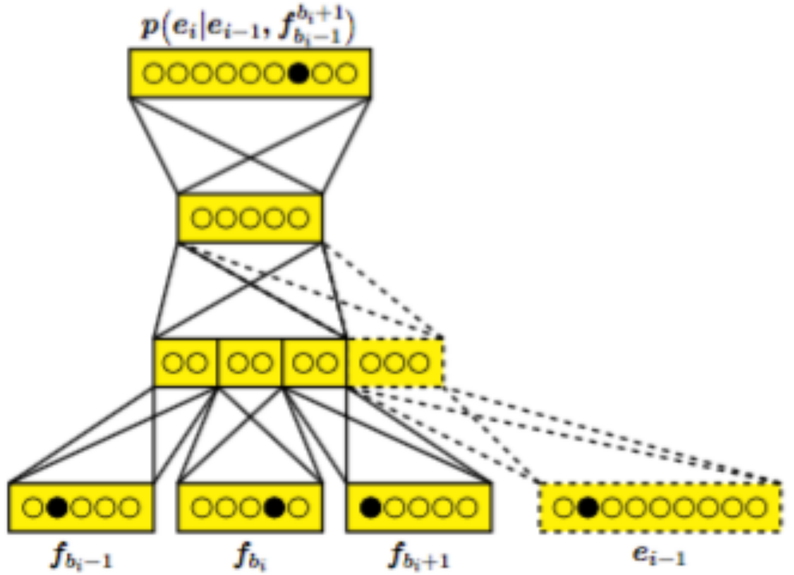
# MLPs for Machine Translation - “Hybrid” SMT

- Another idea - Bilingual LM



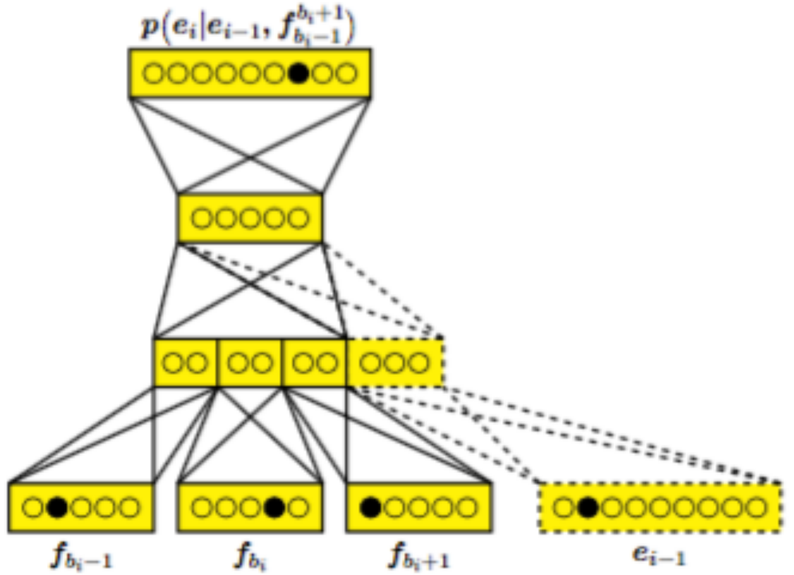
# MLPs for Machine Translation - “Hybrid” SMT

- Another idea - Bilingual LM
- Inputs are 1-hot encodings of:



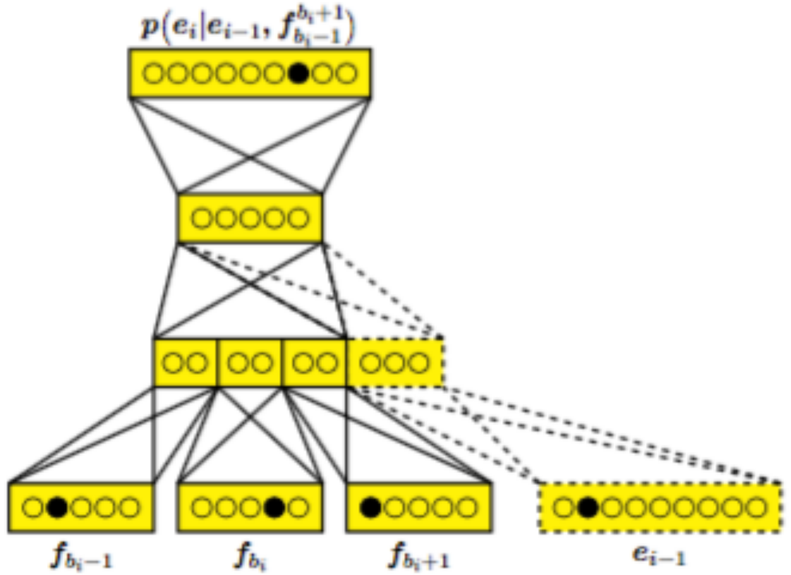
# MLPs for Machine Translation - “Hybrid” SMT

- Another idea - Bilingual LM
- Inputs are 1-hot encodings of:
  - The words in the aligned source language window



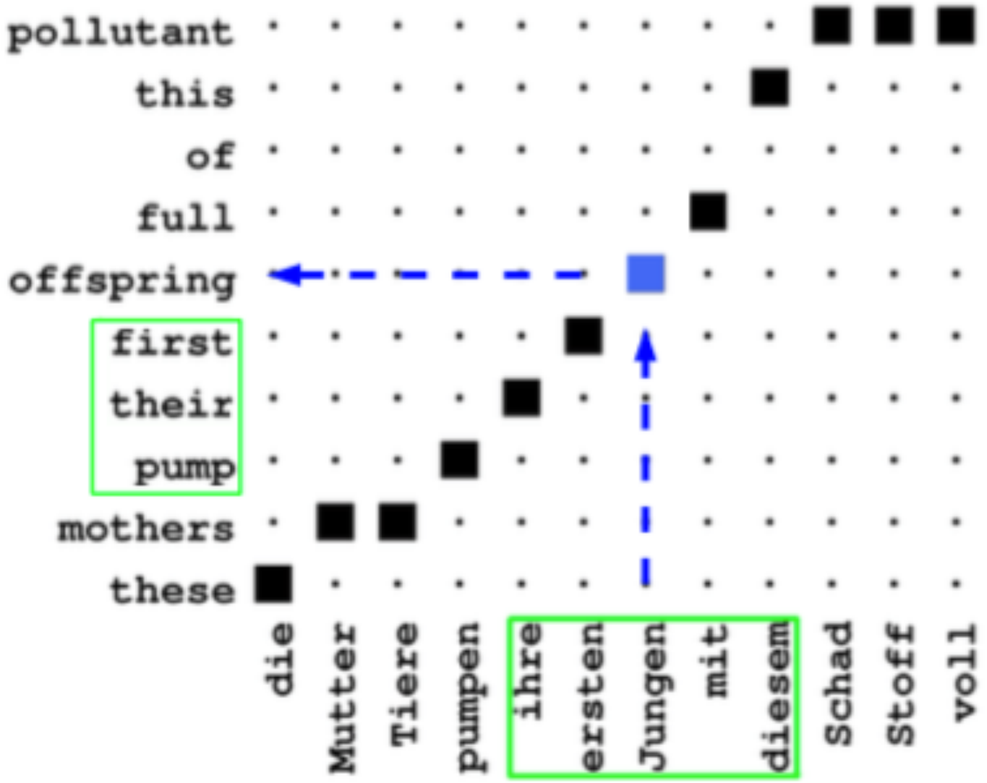
# MLPs for Machine Translation - “Hybrid” SMT

- Another idea - Bilingual LM
- Inputs are 1-hot encodings of:
  - The words in the aligned source language window
  - Previous words in the translation hypothesis



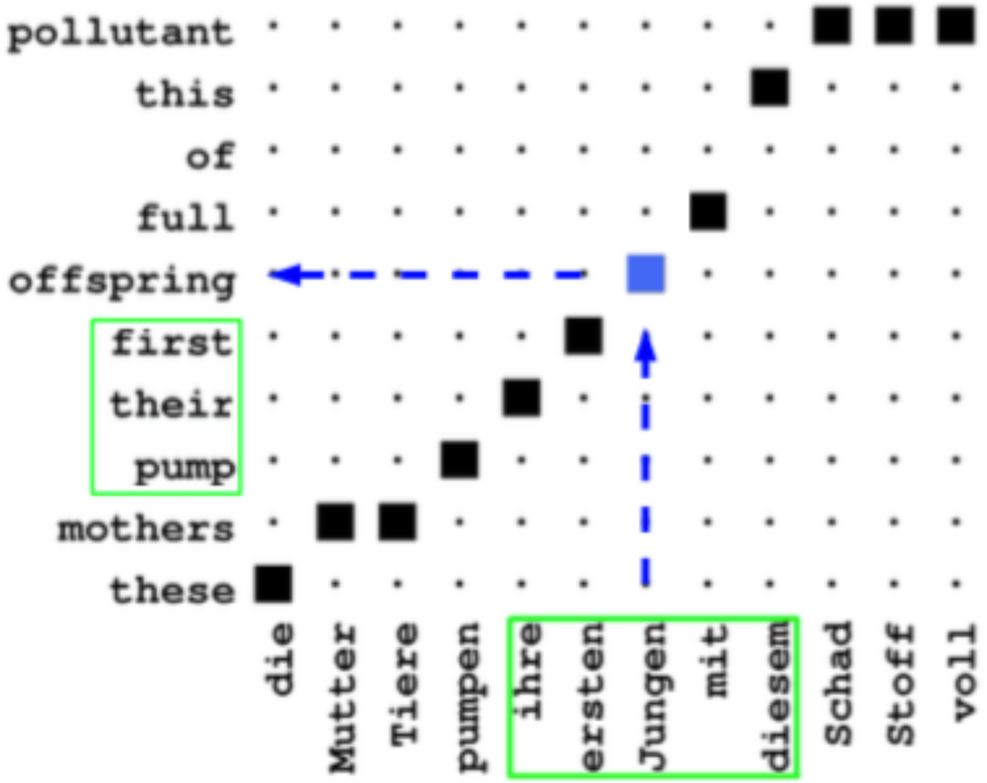
# MLPs for Machine Translation - “Hybrid” SMT

- Another idea - Bilingual LM
- Inputs are 1-hot encodings of:
  - The words in the aligned source language window
  - Previous words in the translation hypothesis



# MLPs for Machine Translation - “Hybrid” SMT

- Another idea - Bilingual LM
- Inputs are 1-hot encodings of:
  - The words in the aligned source language window
  - Previous words in the translation hypothesis
- ACL 2014 Best Paper (Devlin et al, 2014)



# Machine Translation with RNNs

# Machine Translation with RNNs

- First (modern) models for end-to-end Neural Machine Translation presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014

# Machine Translation with RNNs

- First (modern) models for end-to-end Neural Machine Translation presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modelling

# Machine Translation with RNNs

- First (modern) models for end-to-end Neural Machine Translation presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modelling
- The Idea - 2 RNN’s, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture, sequence-to-sequence learning, seq2seq)

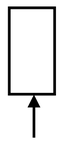
# Machine Translation with RNNs

- First (modern) models for end-to-end Neural Machine Translation presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modelling
- The Idea - 2 RNN’s, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture, sequence-to-sequence learning, seq2seq)

Encoder

# Machine Translation with RNNs

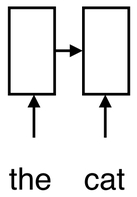
- First (modern) models for end-to-end Neural Machine Translation presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modelling
- The Idea - 2 RNN’s, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture, sequence-to-sequence learning, seq2seq)




Encoder

# Machine Translation with RNNs

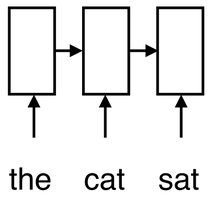
- First (modern) models for end-to-end Neural Machine Translation presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modelling
- The Idea - 2 RNN’s, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture, sequence-to-sequence learning, seq2seq)




Encoder

# Machine Translation with RNNs

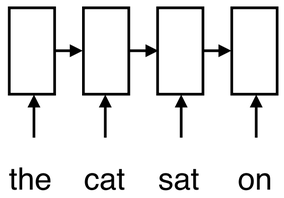
- First (modern) models for end-to-end Neural Machine Translation presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modelling
- The Idea - 2 RNN’s, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture, sequence-to-sequence learning, seq2seq)




Encoder

# Machine Translation with RNNs

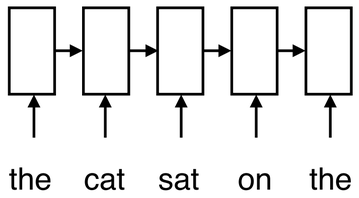
- First (modern) models for end-to-end Neural Machine Translation presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modelling
- The Idea - 2 RNN’s, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture, sequence-to-sequence learning, seq2seq)




Encoder

# Machine Translation with RNNs

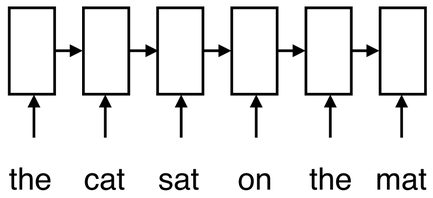
- First (modern) models for end-to-end Neural Machine Translation presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modelling
- The Idea - 2 RNN’s, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture, sequence-to-sequence learning, seq2seq)




Encoder

# Machine Translation with RNNs

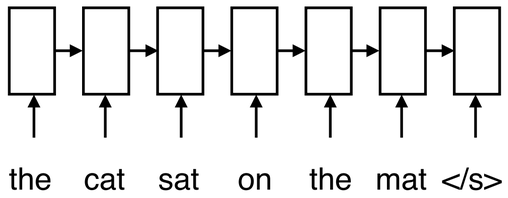
- First (modern) models for end-to-end Neural Machine Translation presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modelling
- The Idea - 2 RNN’s, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture, sequence-to-sequence learning, seq2seq)




Encoder

# Machine Translation with RNNs

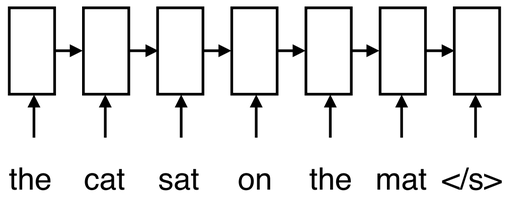
- First (modern) models for end-to-end Neural Machine Translation presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modelling
- The Idea - 2 RNN’s, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture, sequence-to-sequence learning, seq2seq)




Encoder

# Machine Translation with RNNs

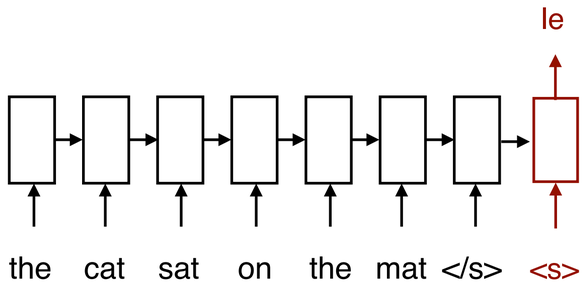
- First (modern) models for end-to-end Neural Machine Translation presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modelling
- The Idea - 2 RNN’s, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture, sequence-to-sequence learning, seq2seq)




Encoder Decoder

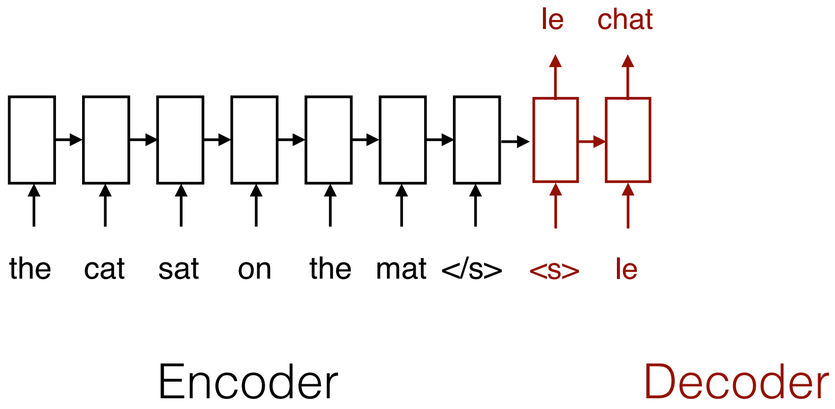
# Machine Translation with RNNs

- First (modern) models for end-to-end Neural Machine Translation presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modelling
- The Idea - 2 RNN’s, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture, sequence-to-sequence learning, seq2seq)



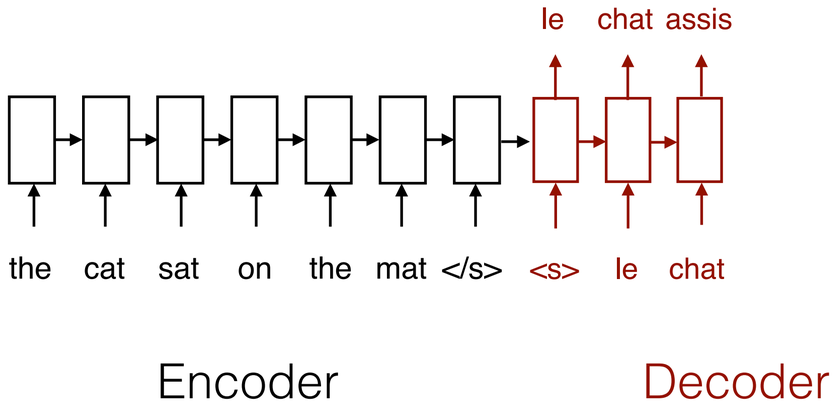
# Machine Translation with RNNs

- First (modern) models for end-to-end Neural Machine Translation presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modelling
- The Idea - 2 RNN's, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture, sequence-to-sequence learning, seq2seq)



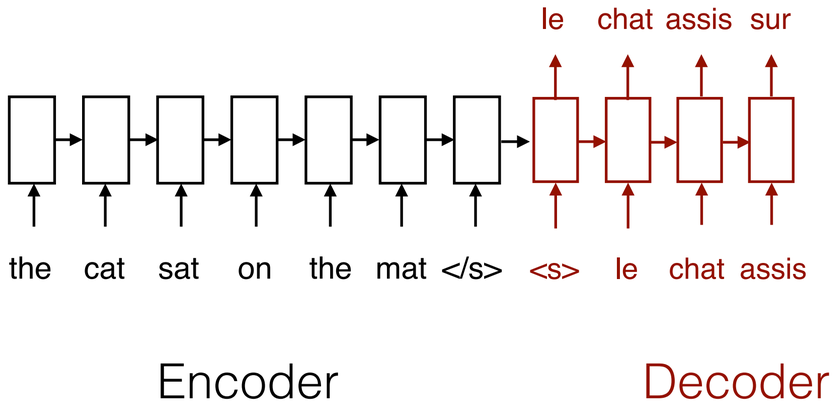
# Machine Translation with RNNs

- First (modern) models for end-to-end Neural Machine Translation presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modelling
- The Idea - 2 RNN's, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture, sequence-to-sequence learning, seq2seq)



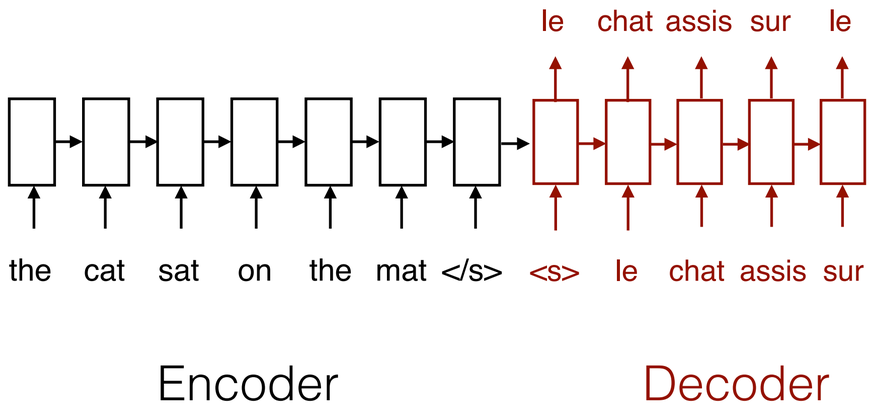
# Machine Translation with RNNs

- First (modern) models for end-to-end Neural Machine Translation presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modelling
- The Idea - 2 RNN’s, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture, sequence-to-sequence learning, seq2seq)



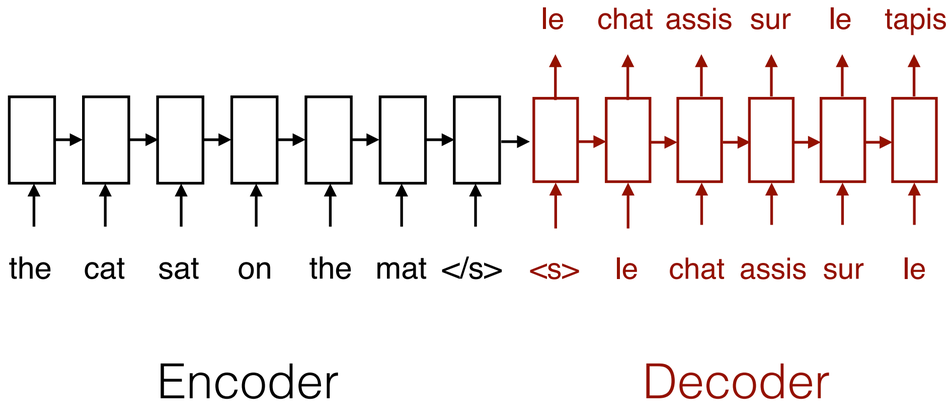
# Machine Translation with RNNs

- First (modern) models for end-to-end Neural Machine Translation presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modelling
- The Idea - 2 RNN’s, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture, sequence-to-sequence learning, seq2seq)



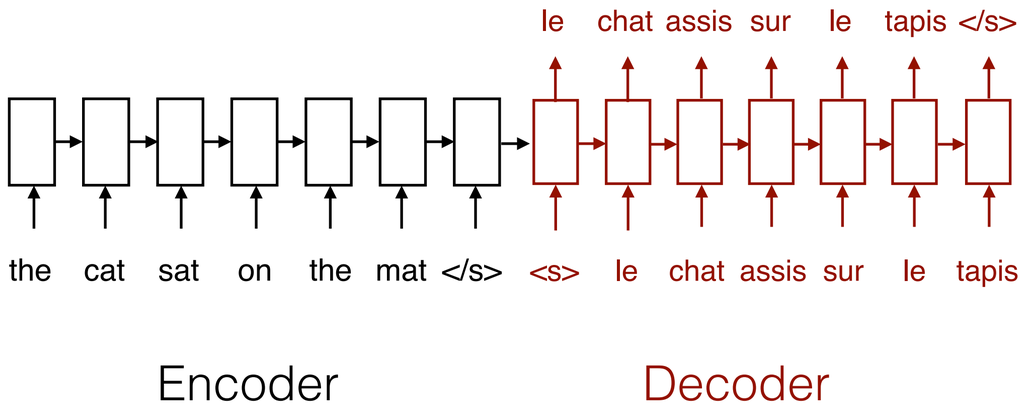
# Machine Translation with RNNs

- First (modern) models for end-to-end Neural Machine Translation presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modelling
- The Idea - 2 RNN's, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture, sequence-to-sequence learning, seq2seq)



# Machine Translation with RNNs

- First (modern) models for end-to-end Neural Machine Translation presented by Kalchbrenner et. al. 2013, Sutskever et al., 2014, Cho et al., 2014
- Inspired by RNN language modelling
- The Idea - 2 RNN's, one for reading the input and one for writing the output (a.k.a the encoder-decoder architecture, sequence-to-sequence learning, seq2seq)



# Machine Translation with RNNs

# Machine Translation with RNNs

- More formally - model $p(y|x)$ using a single neural network:

# Machine Translation with RNNs

- More formally - model $p(y|x)$ using a single neural network:

$$y = y_1 ... y_N$$

# Machine Translation with RNNs

- More formally - model $p(y|x)$ using a single neural network:

$$y = y_1 ... y_N$$

$$p(y|x) = p(y_1|x)p(y_2|y_1, x)p(y_3|y_1, y_2, x) \dots p(y_N|y_1 ... y_{N-1}, x)$$

# Machine Translation with RNNs

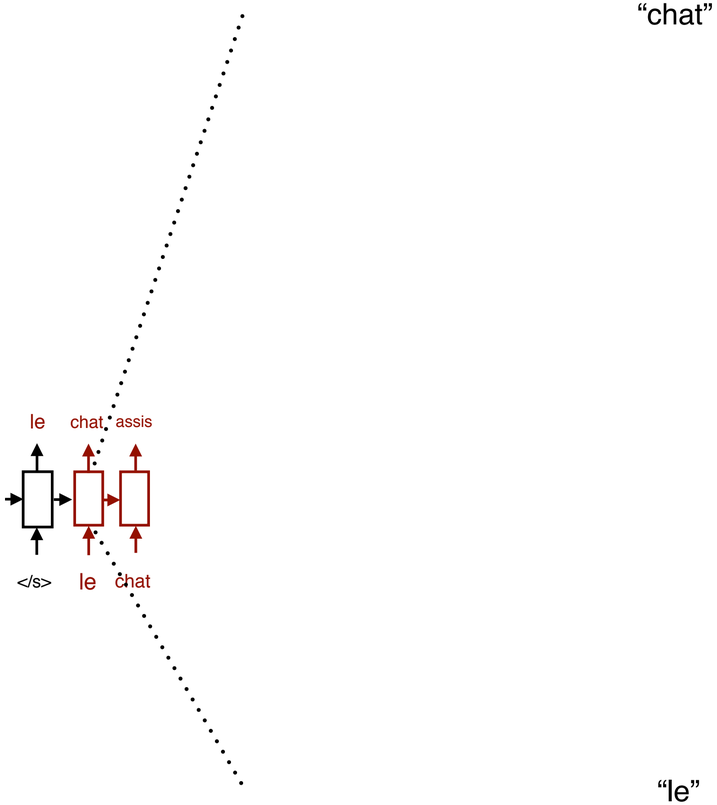
- More formally - model $p(y|x)$ using a single neural network:

$$y = y_1 ... y_N$$

$$p(y|x) = p(y_1|x)p(y_2|y_1, x)p(y_3|y_1, y_2, x) \dots p(y_N|y_1 ... y_{N-1}, x)$$

$$p(y_i = word_k|y_{<i}, x) = softmax_k(NN_\Theta(y_{<i}, x))$$

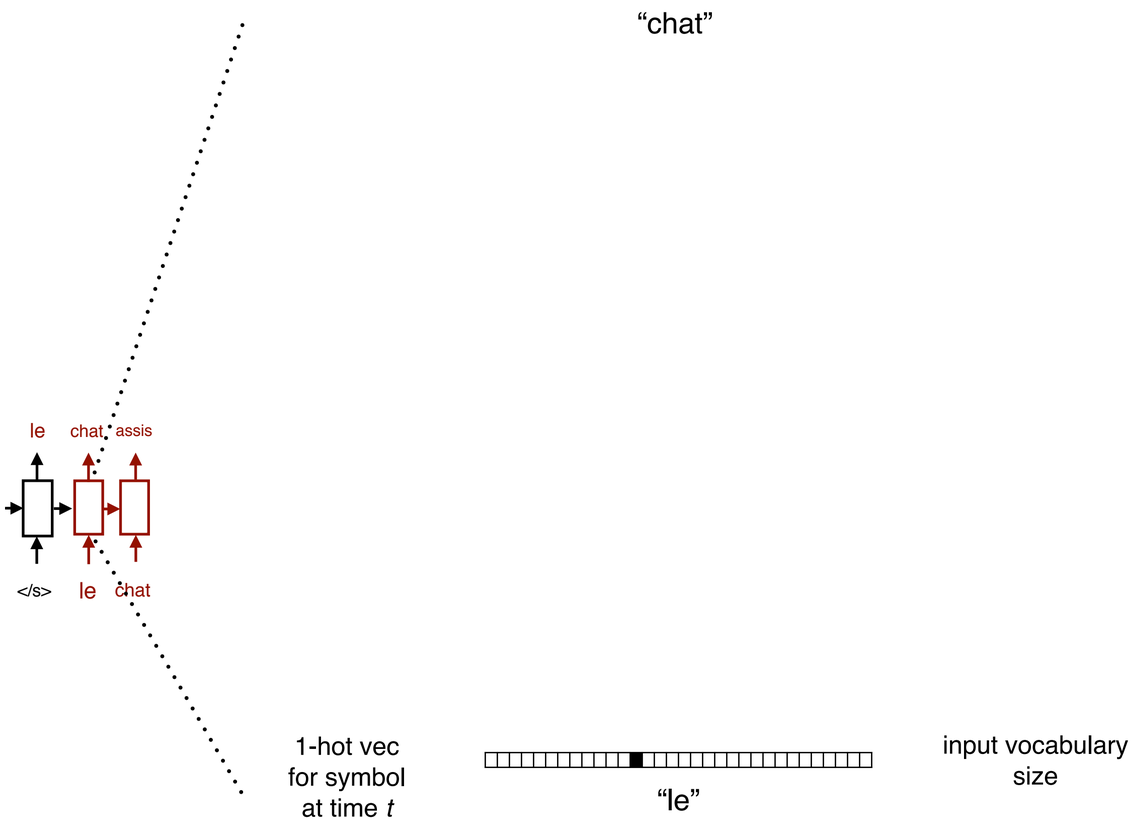
# Decoder Step - Zoom-In



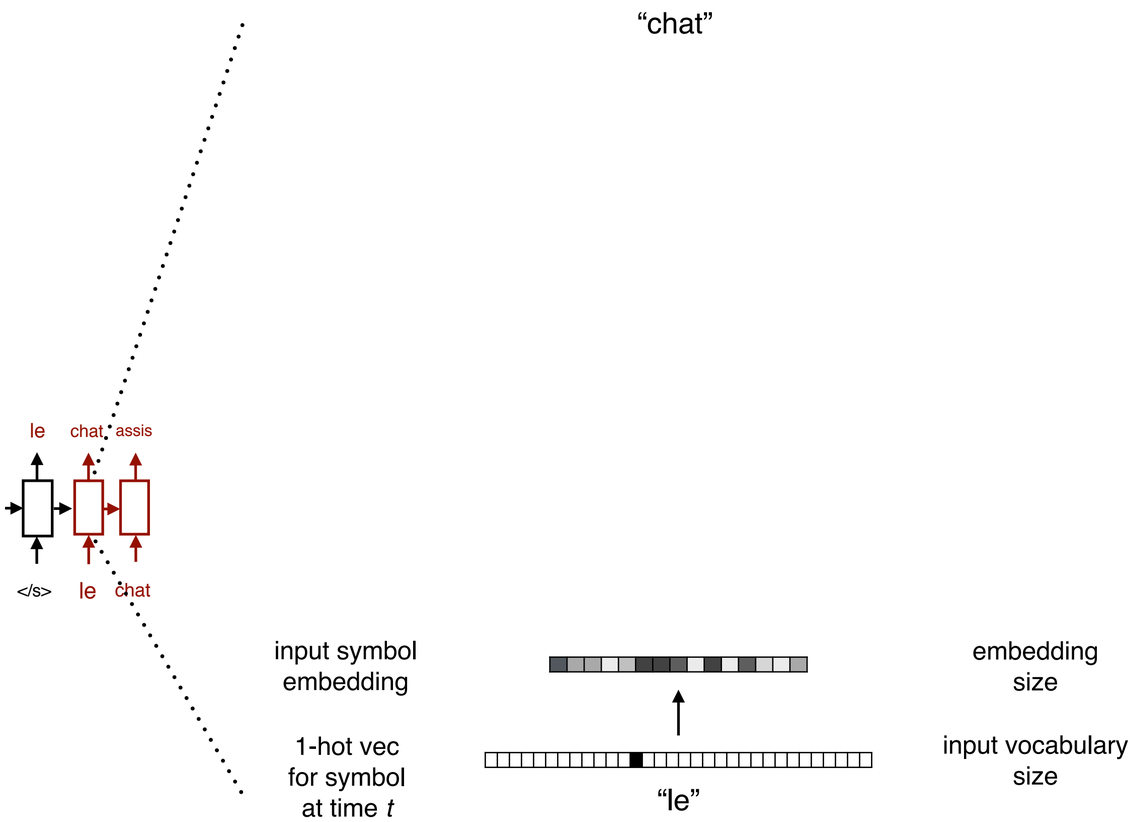

“chat”

“le”

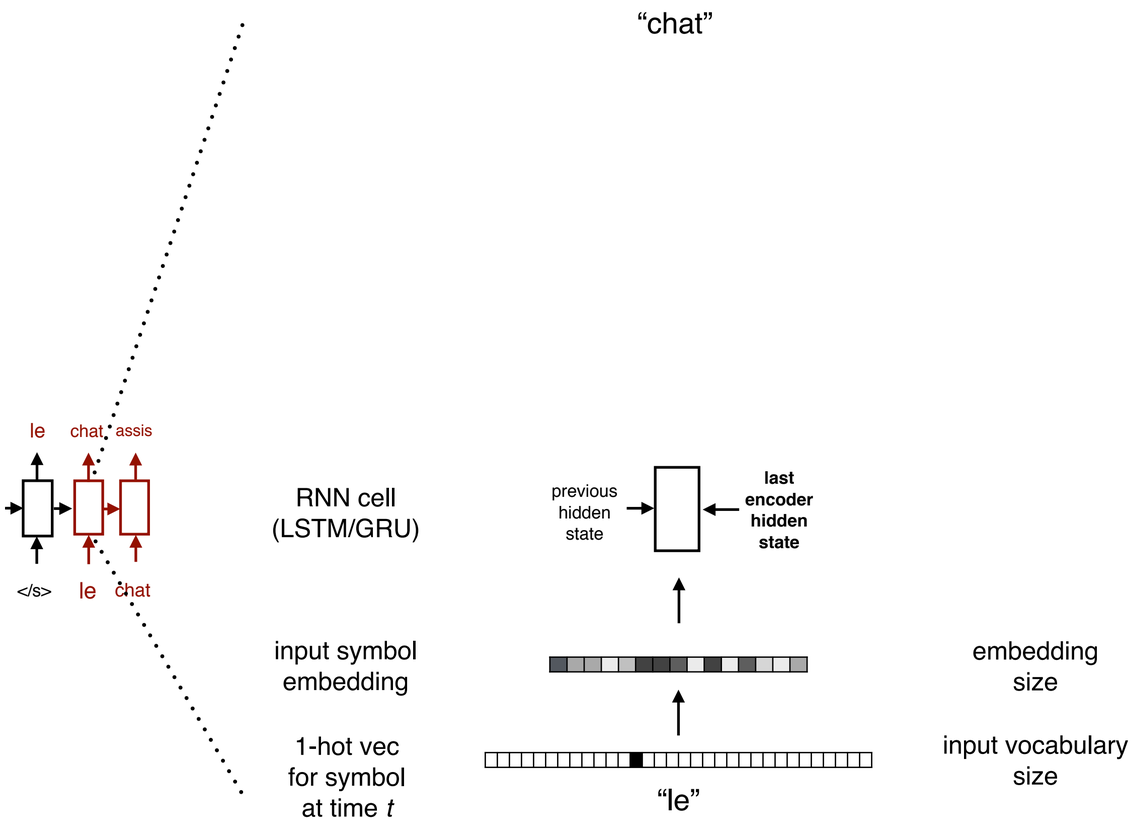
# Decoder Step - Zoom-In



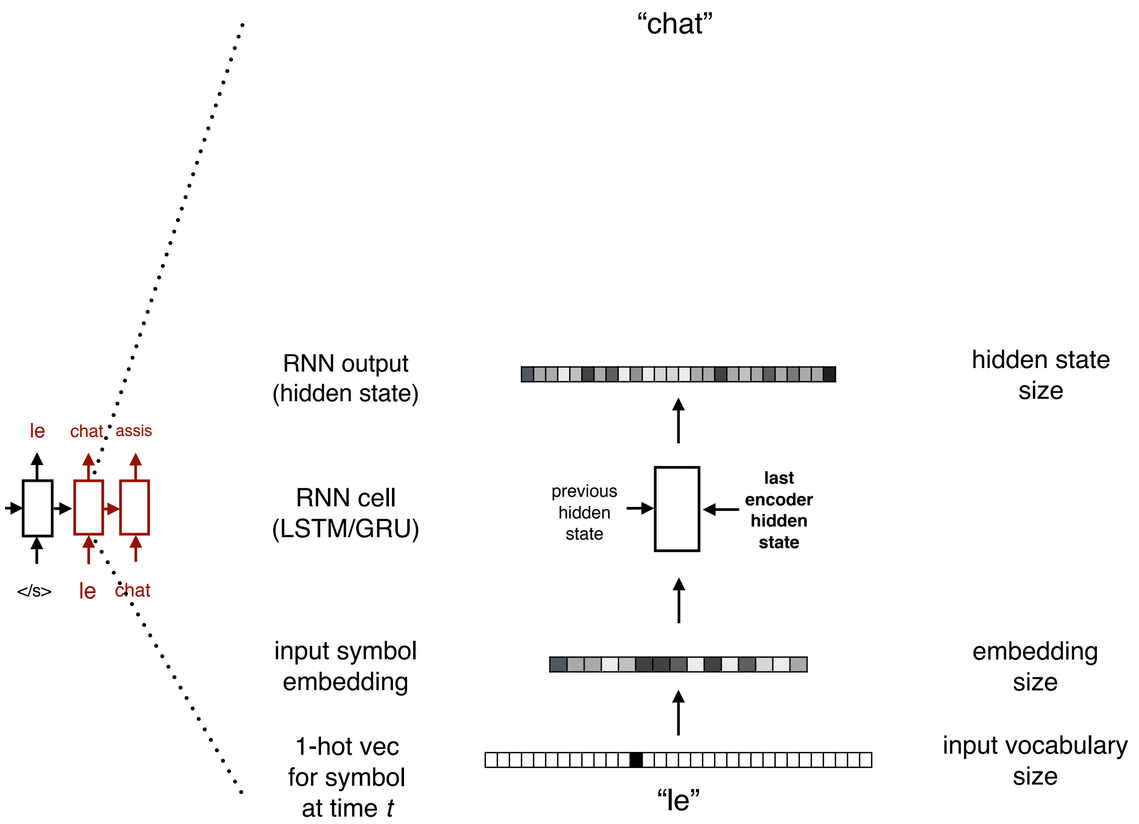
# Decoder Step - Zoom-In



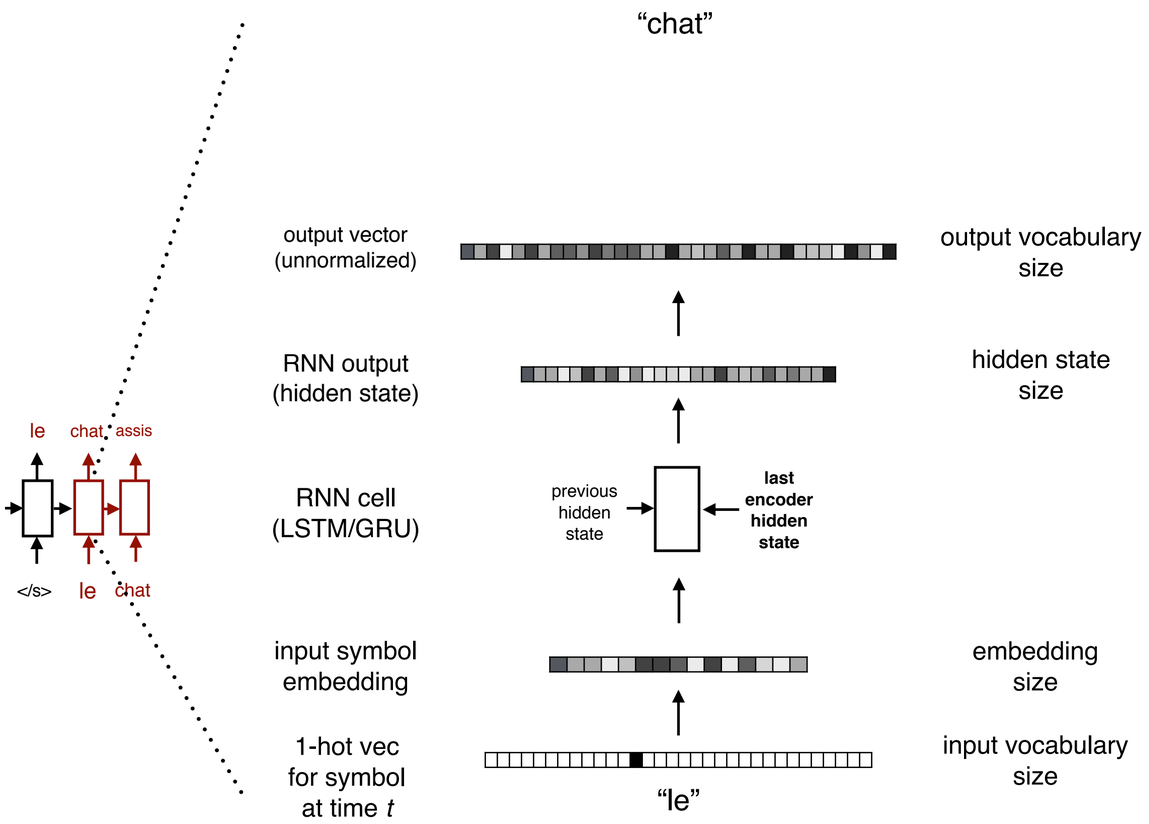
# Decoder Step - Zoom-In



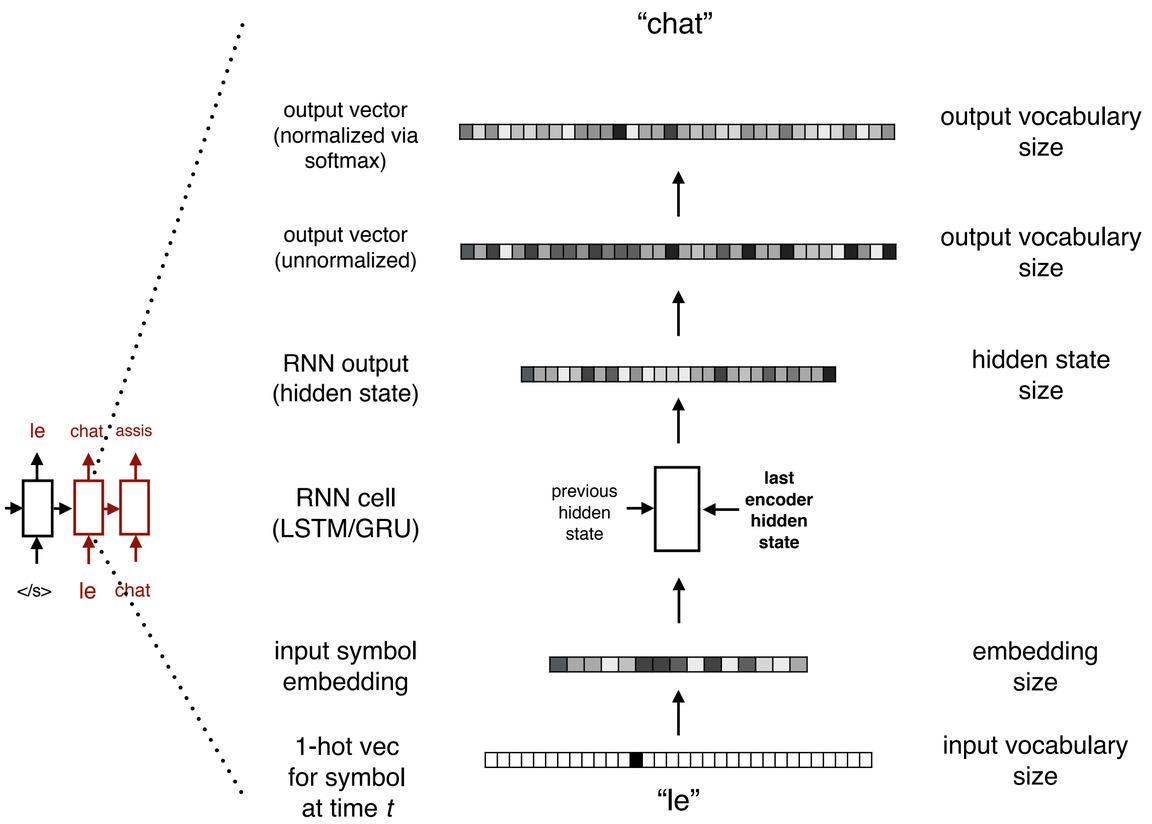
# Decoder Step - Zoom-In



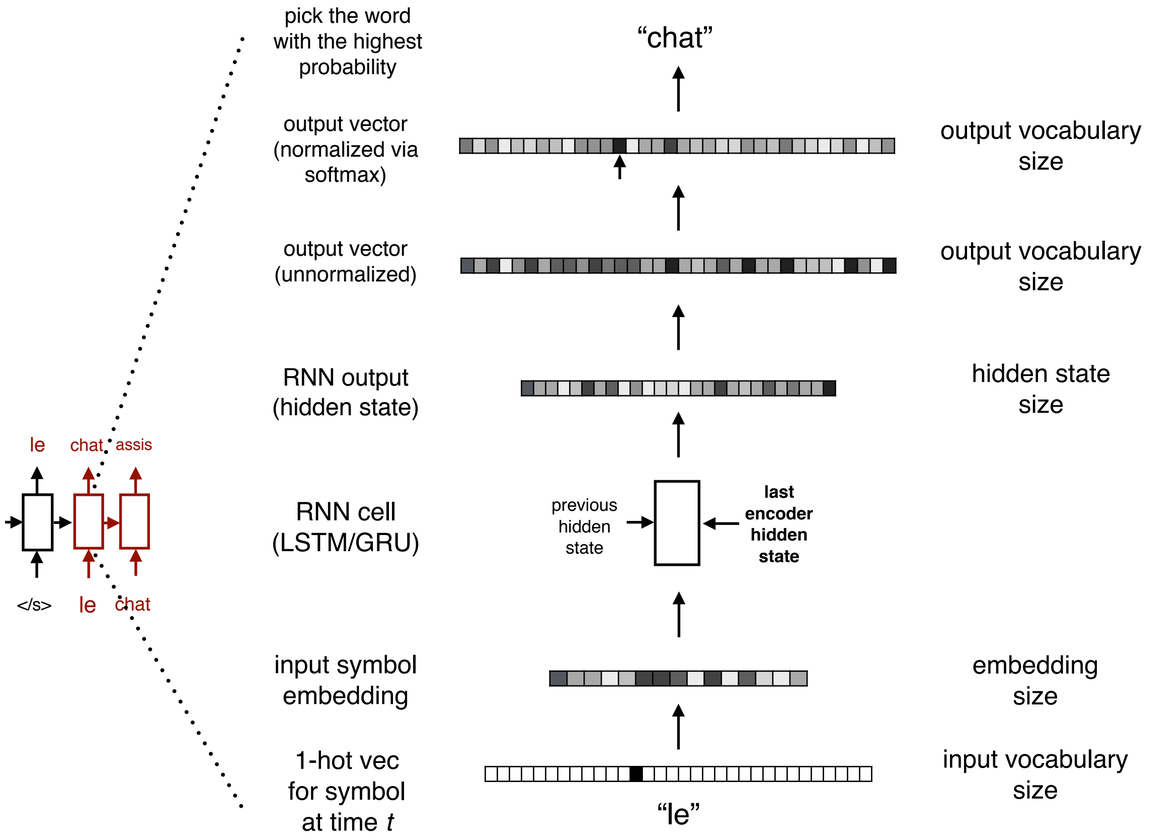
# Decoder Step - Zoom-In



# Decoder Step - Zoom-In



# Decoder Step - Zoom-In



# The Problem with Vanilla seq2seq

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

# The Attention Mechanism

# The Attention Mechanism

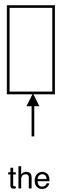
- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input

# The Attention Mechanism

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state

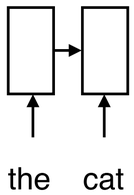
# The Attention Mechanism

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



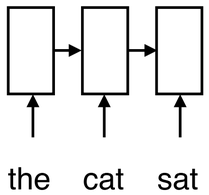
# The Attention Mechanism

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



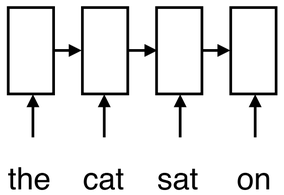
# The Attention Mechanism

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



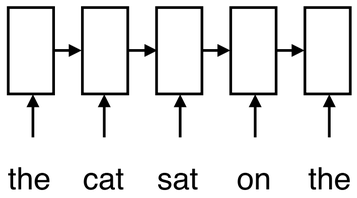
# The Attention Mechanism

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



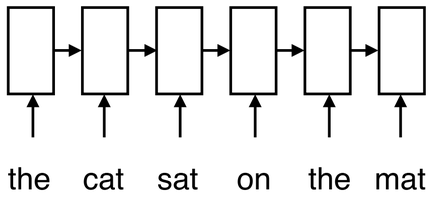
# The Attention Mechanism

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



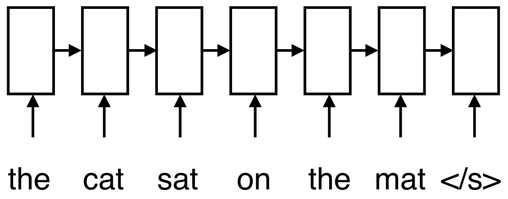
# The Attention Mechanism

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



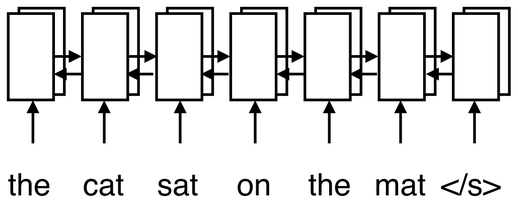
# The Attention Mechanism

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



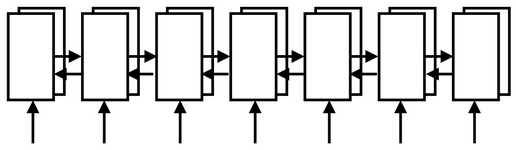
# The Attention Mechanism

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



# The Attention Mechanism

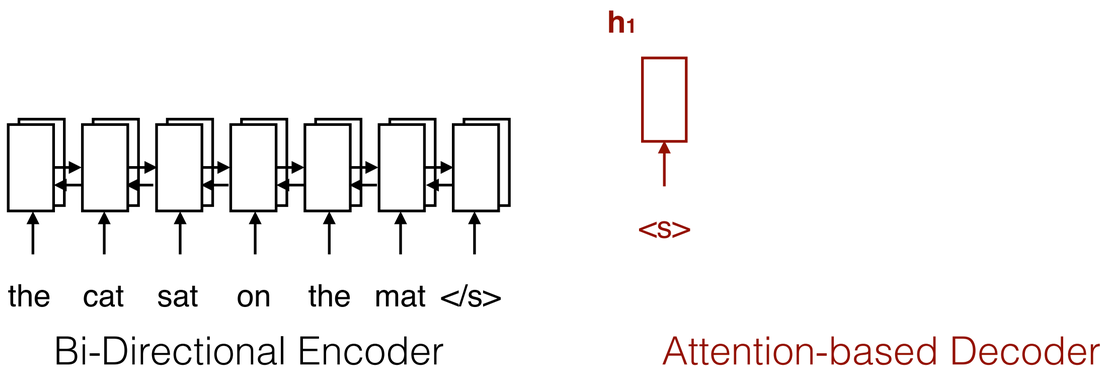
- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state




Bi-Directional Encoder

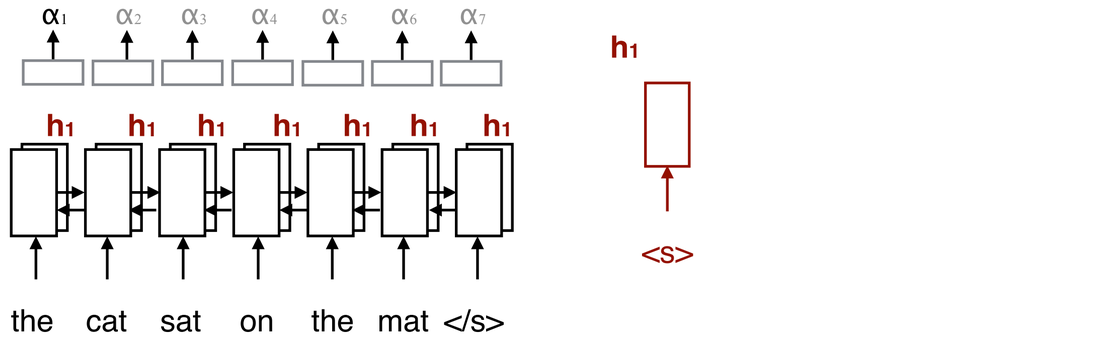
# The Attention Mechanism

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



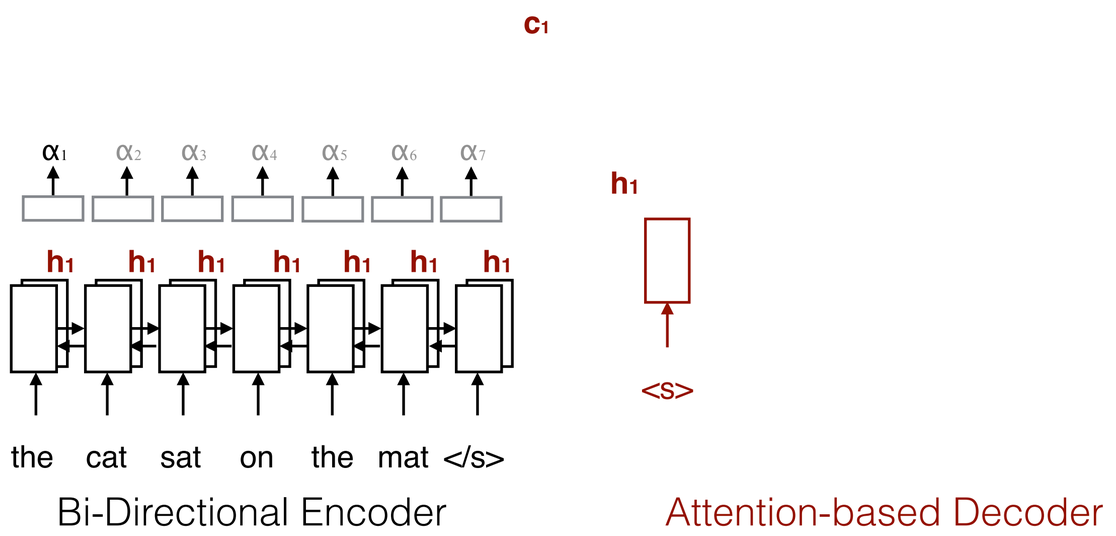
# The Attention Mechanism

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



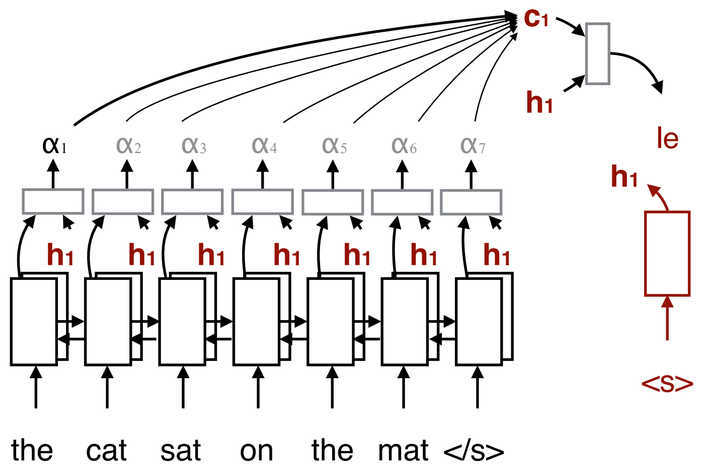
# The Attention Mechanism

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



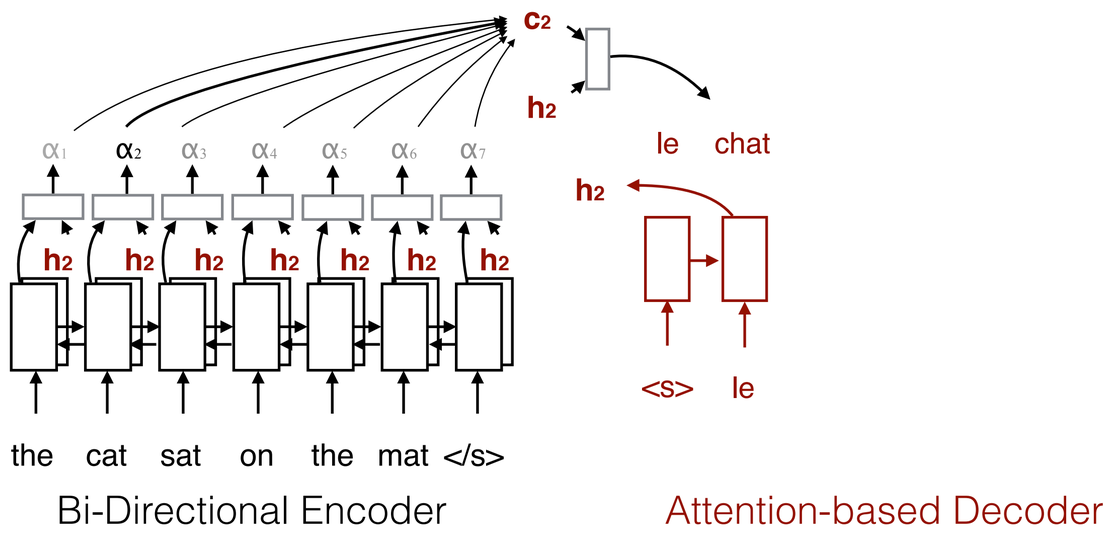
# The Attention Mechanism

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



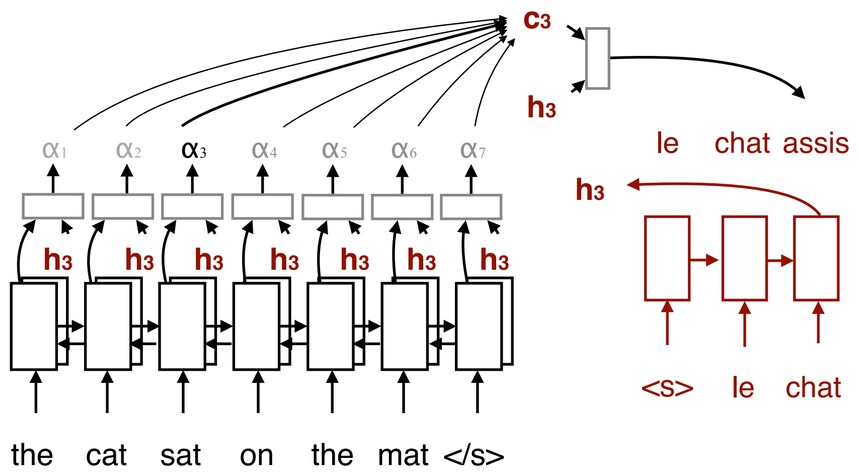
# The Attention Mechanism

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



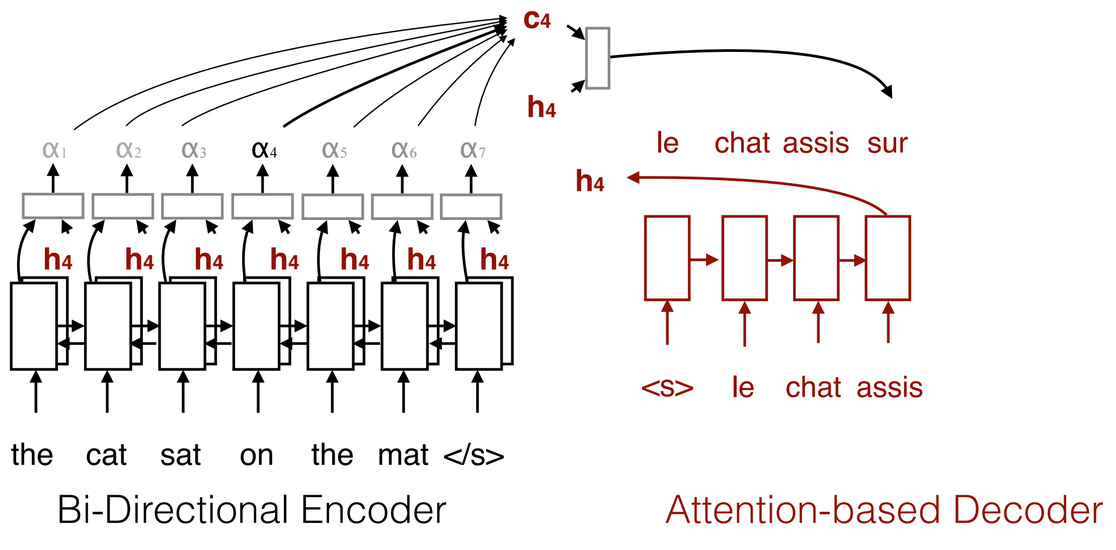
# The Attention Mechanism

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



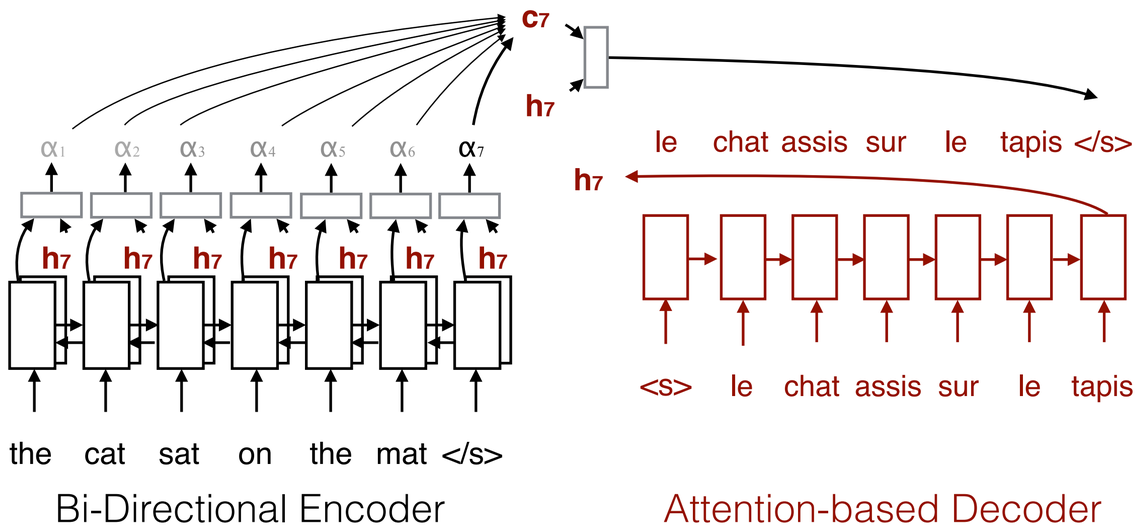
# The Attention Mechanism

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



# The Attention Mechanism

- Instead of using a single vector as a fixed representation of the input sequence, “attend” at each step to the relevant parts of the input
- The “importance” of each input element to the current prediction is computed via a feed-forward network that gets the input element and the current decoder state



# The Attention Mechanism

# The Attention Mechanism

- And a bit more formally - in each decoder step:

# The Attention Mechanism

- And a bit more formally - in each decoder step:
  - Compute attention scores for each input element:

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \tanh(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s])$$

# The Attention Mechanism

- And a bit more formally - in each decoder step:
  - Compute attention scores for each input element:

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \tanh(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s])$$

  - Normalize the attention scores so they sum up to 1:

$$\boldsymbol{a}_t(s) = \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)}$$

# The Attention Mechanism

- And a bit more formally - in each decoder step:
  - Compute attention scores for each input element:

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \tanh(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s])$$

  - Normalize the attention scores so they sum up to 1:

$$\boldsymbol{a}_t(s) = \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)}$$

  - Compute $c_t$:

$$c_t = \sum_{j=1}^{T_x} a_j \bar{h}_j$$

# The Attention Mechanism

- And a bit more formally - in each decoder step:
  - Compute attention scores for each input element:

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \tanh(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s])$$

  - Normalize the attention scores so they sum up to 1:

$$\boldsymbol{a}_t(s) = \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)}$$

  - Compute $c_t$:

$$c_t = \sum_{j=1}^{T_x} a_j \bar{h}_j$$

  - Compute attention output state:

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W_c}[\boldsymbol{c}_t; \boldsymbol{h}_t])$$

# The Attention Mechanism

- And a bit more formally - in each decoder step:
  - Compute attention scores for each input element:

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \tanh(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s])$$

  - Normalize the attention scores so they sum up to 1:

$$\boldsymbol{a}_t(s) = \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)}$$

  - Compute $c_t$:

$$c_t = \sum_{j=1}^{T_x} a_j \bar{h}_j$$

  - Compute attention output state:

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W_c}[\boldsymbol{c}_t; \boldsymbol{h}_t])$$

  - Compute output probability distribution: $p(y_t | y_{<t}, x) = \text{softmax}(\boldsymbol{W_s}\tilde{\boldsymbol{h}}_t)$

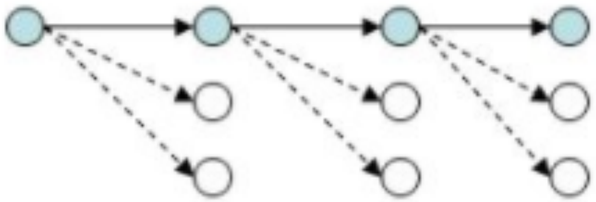
# Decoding with Beam Search

# Decoding with Beam Search

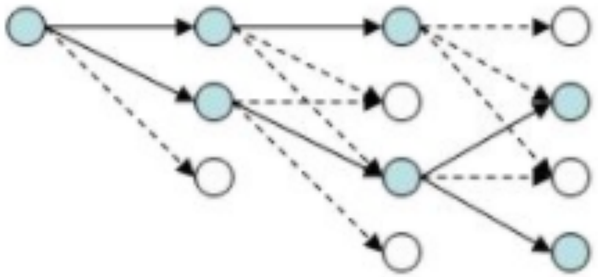
- Instead of keeping one best option on each time step, keep k best options which are updated as-you-go

# Decoding with Beam Search

- Instead of keeping one best option on each time step, keep k best options which are updated as-you-go



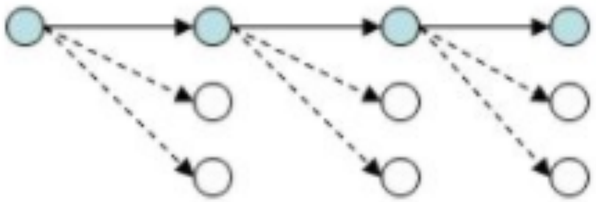
Greedy Search



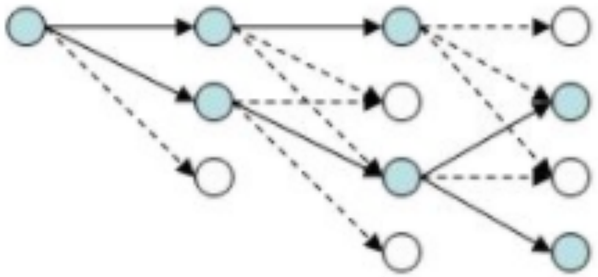
Beam Search (k=2)

# Decoding with Beam Search

- Instead of keeping one best option on each time step, keep k best options which are updated as-you-go
- Requires to maintain different RNN states in-memory for each hypothesis



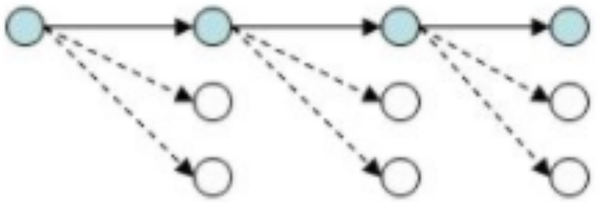
Greedy Search



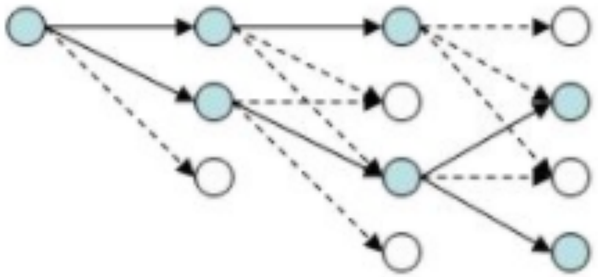
Beam Search (k=2)

# Decoding with Beam Search

- Instead of keeping one best option on each time step, keep k best options which are updated as-you-go
- Requires to maintain different RNN states in-memory for each hypothesis
- Usually a small beam size is enough (5-12)

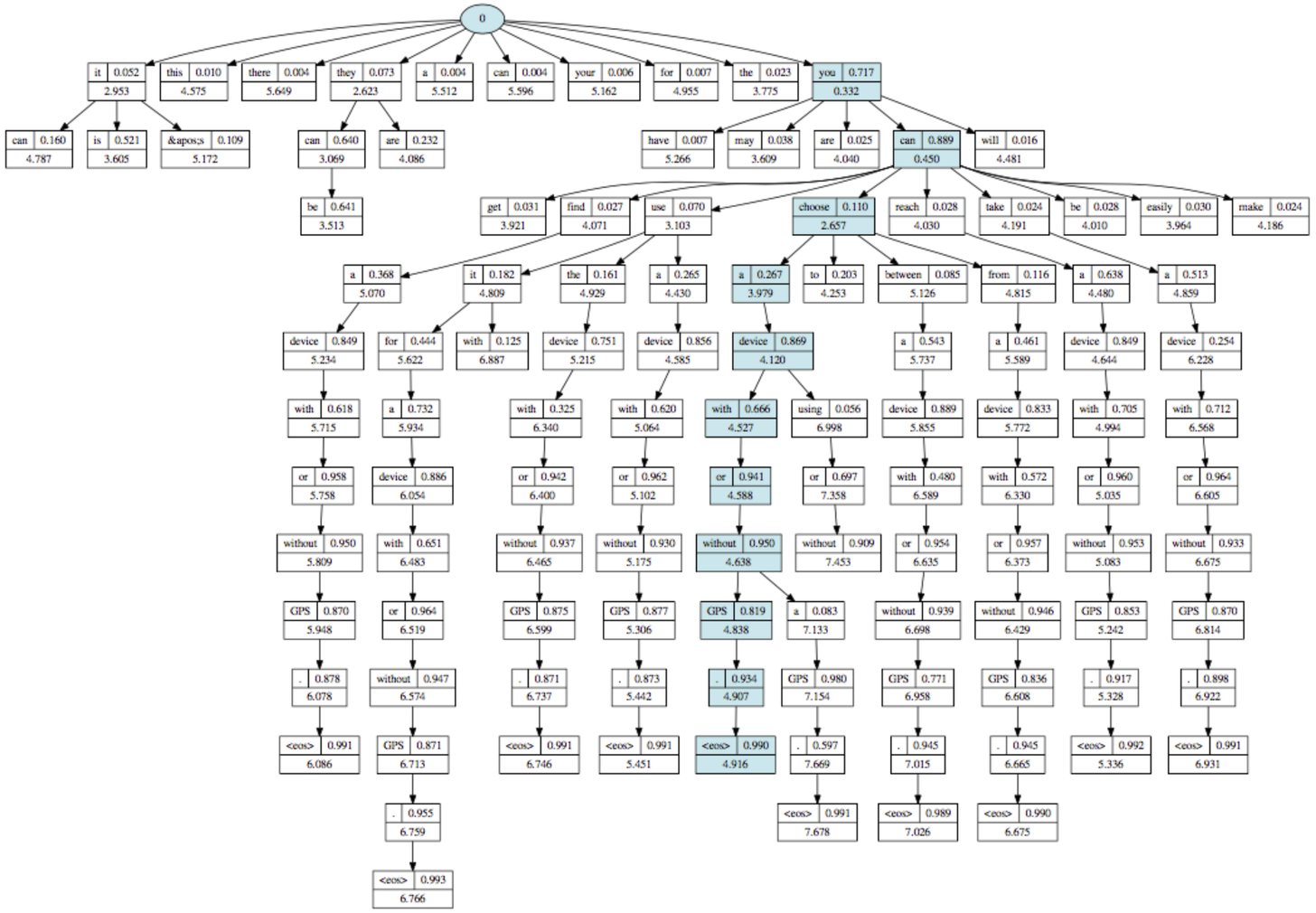


Greedy Search



Beam Search (k=2)

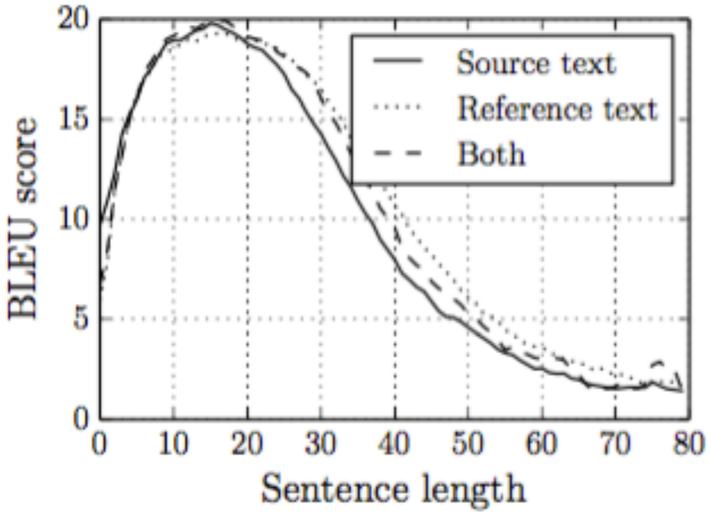
# Decoding with Beam Search



# Effect of Sentence Length

# Effect of Sentence Length

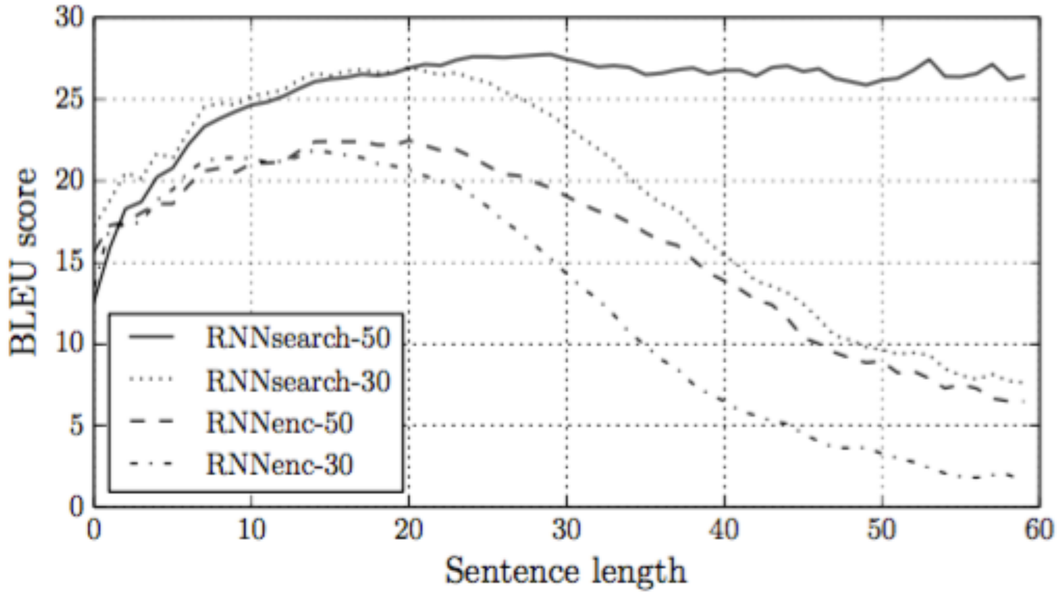
- Results deteriorate for longer sentences as they are “compressed” to a fixed length vector




**Bahdanau, Cho and Bengio (2014)**

# Effect of Sentence Length

- Results deteriorate for longer sentences as they are “compressed” to a fixed length vector
- The attention mechanism “opens the bottleneck”

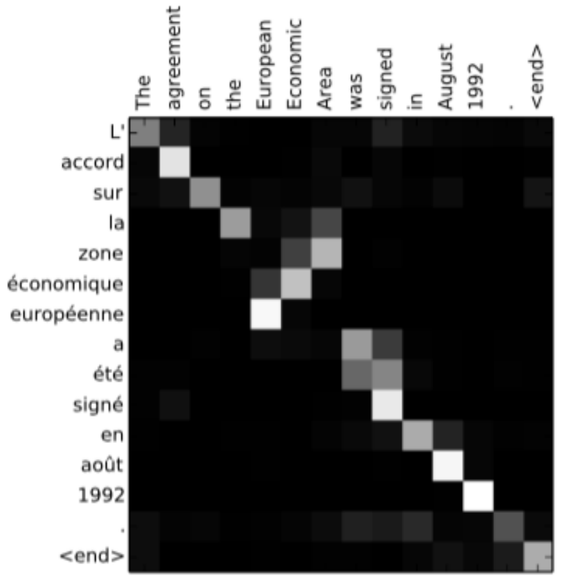


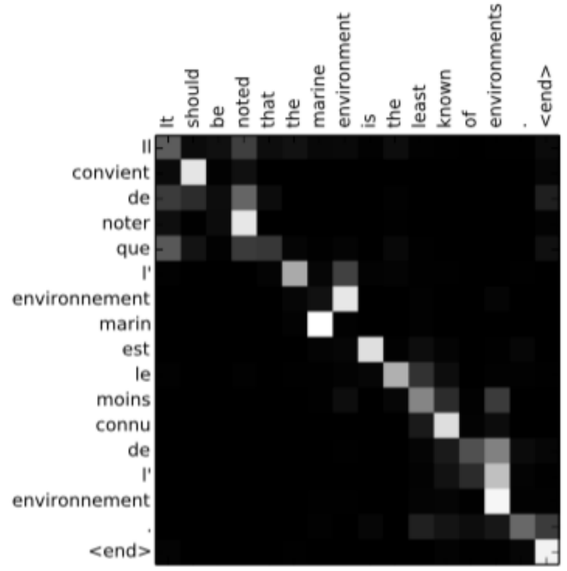

**Bahdanau, Cho and Bengio (2014)**

# Attention and Interpretability

# Attention and Interpretability

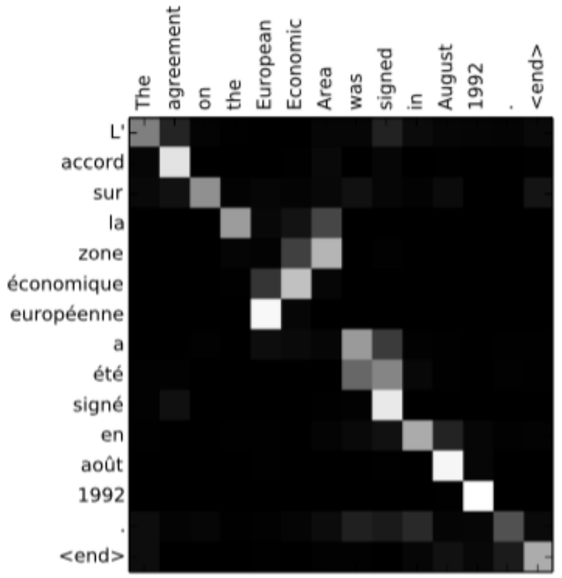
- The model learns to “align” source and target representations

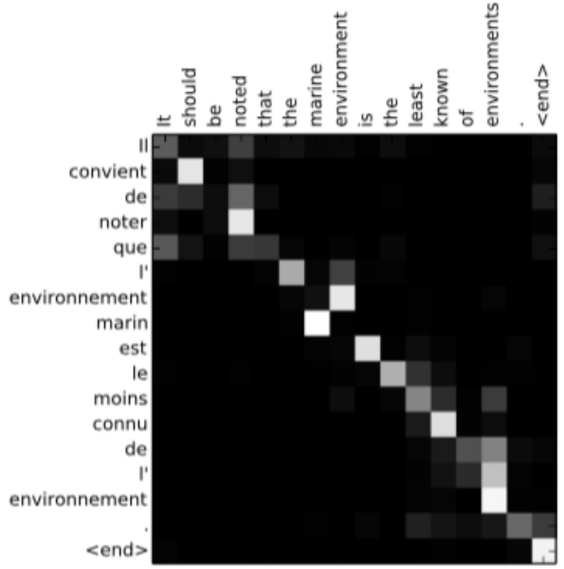


# Attention and Interpretability

- The model learns to “align” source and target representations
- No explicit alignment supervision was given!



# Attention and Interpretability

- The model learns to “align” source and target representations
- No explicit alignment supervision was given!
- But this isn’t always perfect…

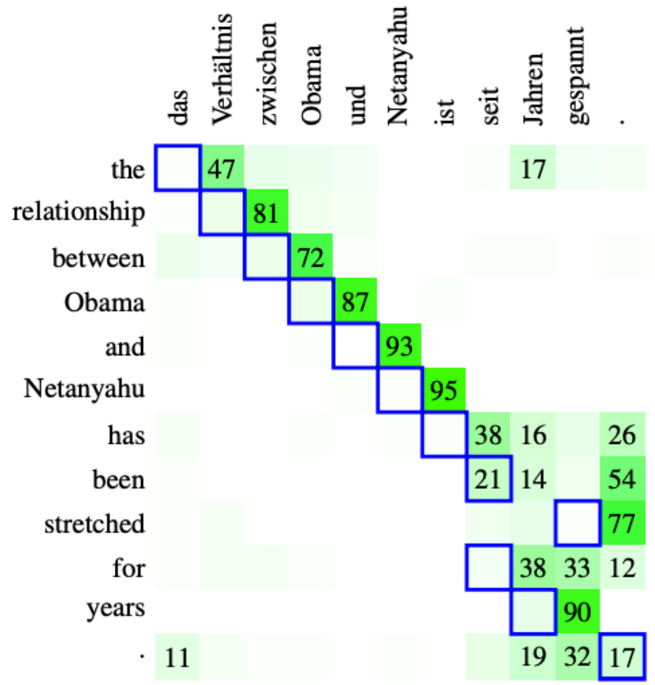



Figure 9: Mismatch between attention states and desired word alignments (German–English).

**Koehn and Knowles, 2017**

# Self Attention and The Transformer Architecture

# Self Attention and The Transformer Architecture

- Vaswani et al. (2017)

# Self Attention and The Transformer Architecture

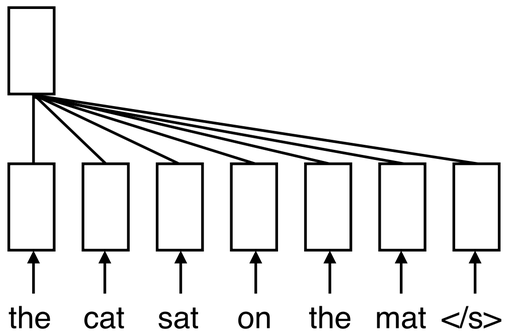
- Vaswani et al. (2017)
  - “Attention is All You Need”

# Self Attention and The Transformer Architecture

- Vaswani et al. (2017)
  - “Attention is All You Need”
- Main idea: replace RNNs with self attention layers

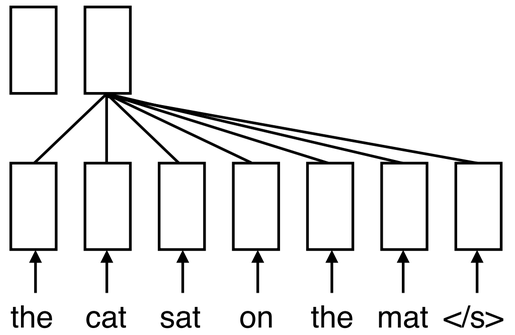
# Self Attention and The Transformer Architecture

- Vaswani et al. (2017)
  - “Attention is All You Need”
- Main idea: replace RNNs with self attention layers



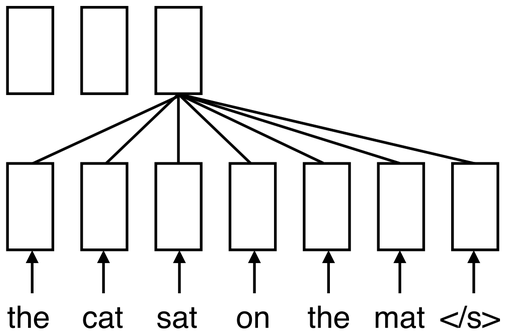
# Self Attention and The Transformer Architecture

- Vaswani et al. (2017)
  - “Attention is All You Need”
- Main idea: replace RNNs with self attention layers



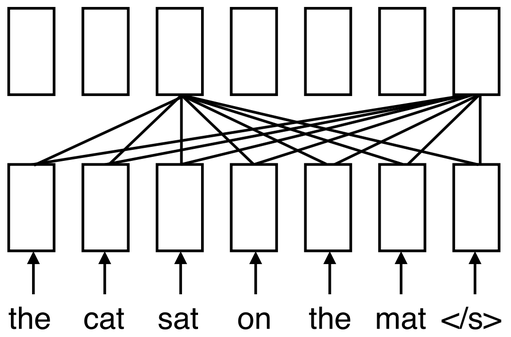
# Self Attention and The Transformer Architecture

- Vaswani et al. (2017)
  - “Attention is All You Need”
- Main idea: replace RNNs with self attention layers

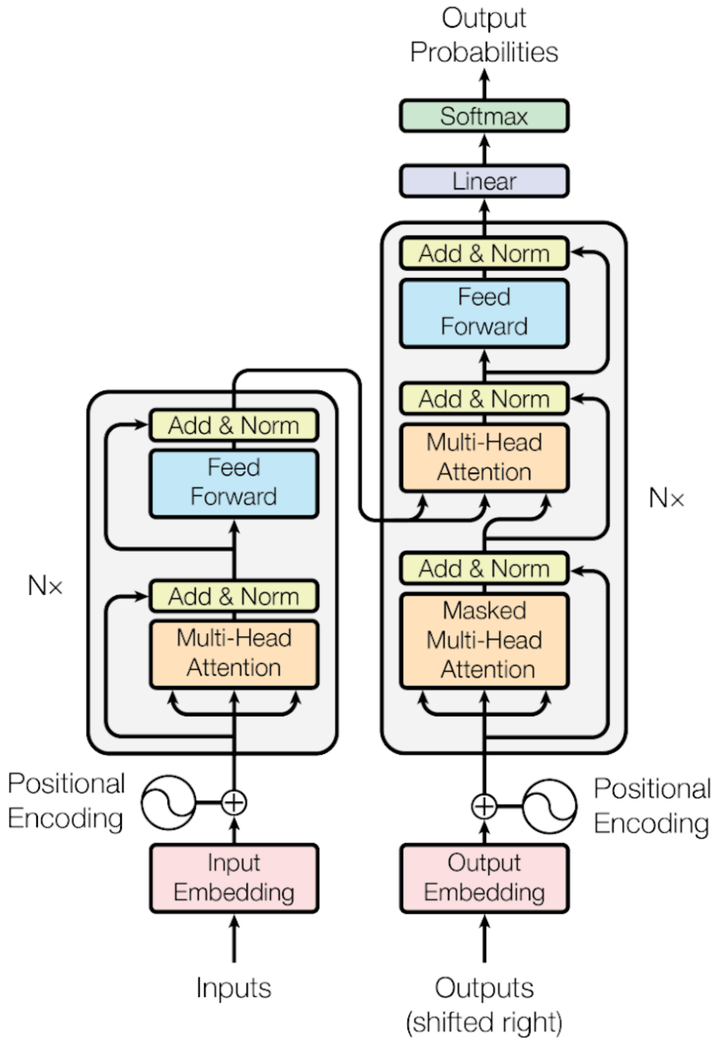


# Self Attention and The Transformer Architecture

- Vaswani et al. (2017)
  - “Attention is All You Need”
- Main idea: replace RNNs with self attention layers
- Can be parallelized at the sequence level - faster training of large networks

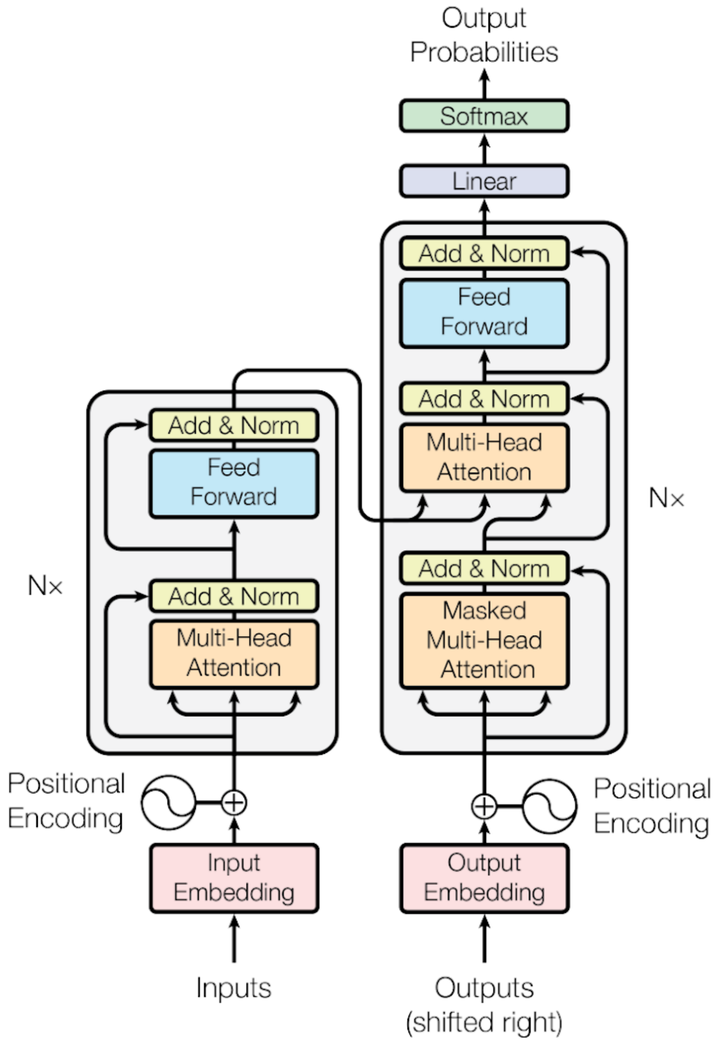


# Other Important Details



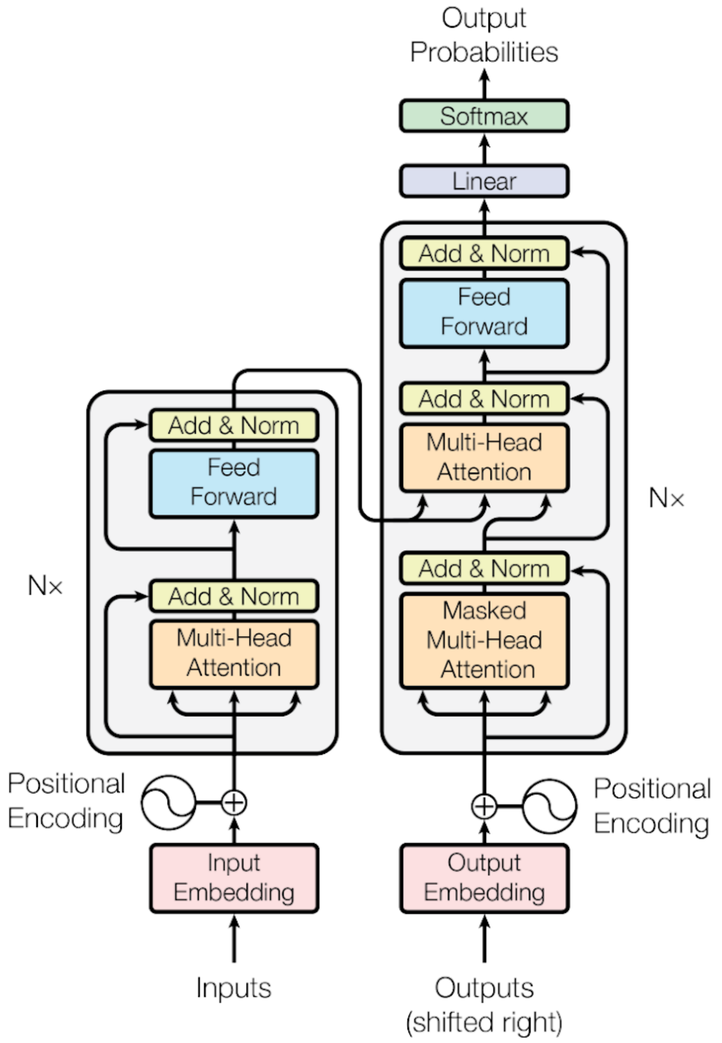
# Other Important Details

- Positional encodings



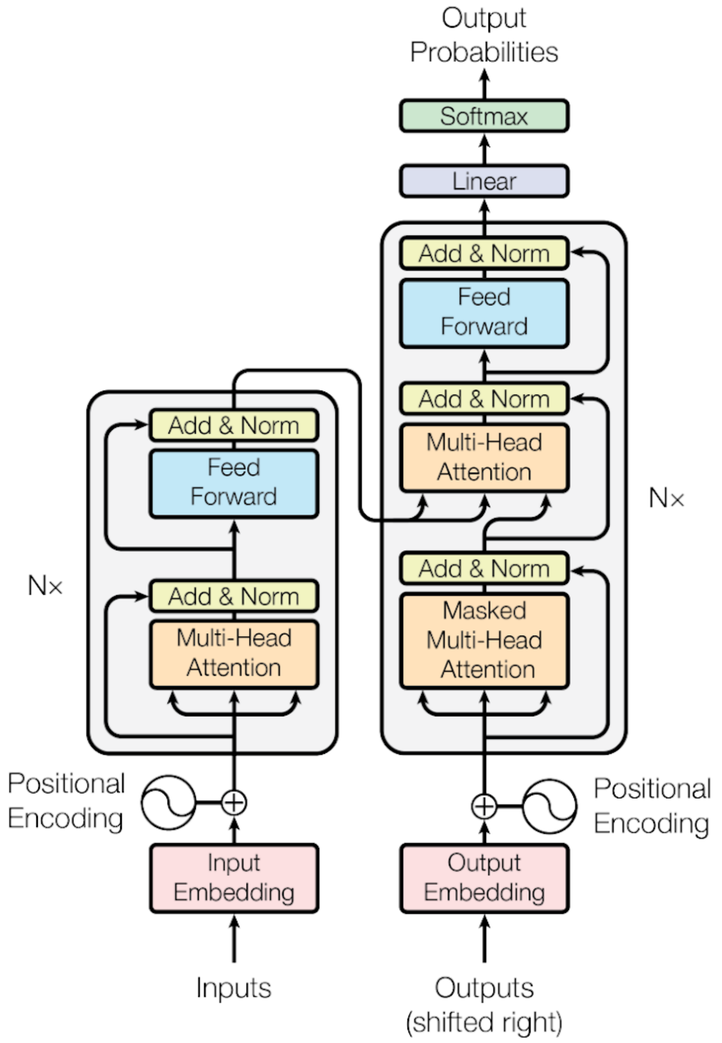
# Other Important Details

- Positional encodings
- Multi-head attention



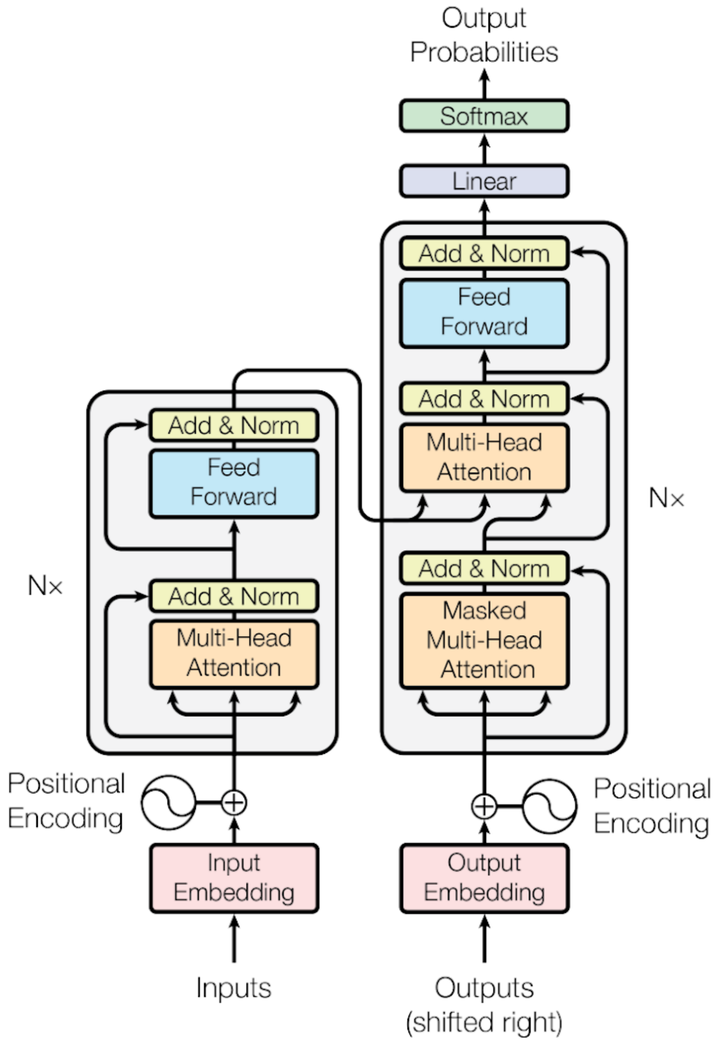
# Other Important Details

- Positional encodings
- Multi-head attention
- Layer normalization



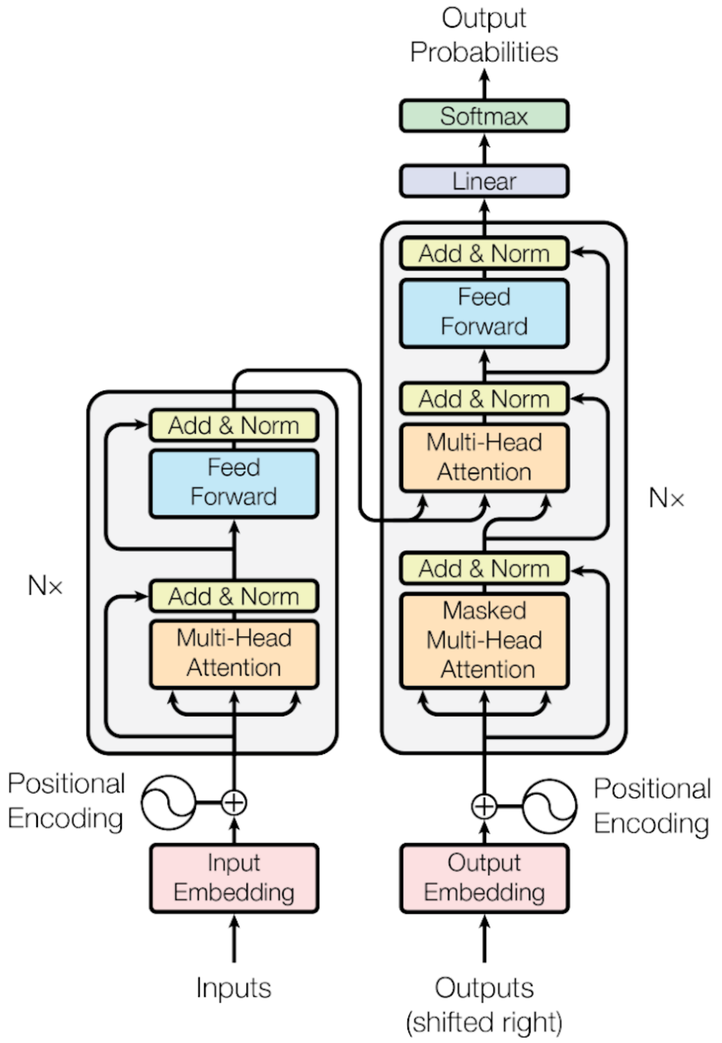
# Other Important Details

- Positional encodings
- Multi-head attention
- Layer normalization
- Decoder - masked self attention



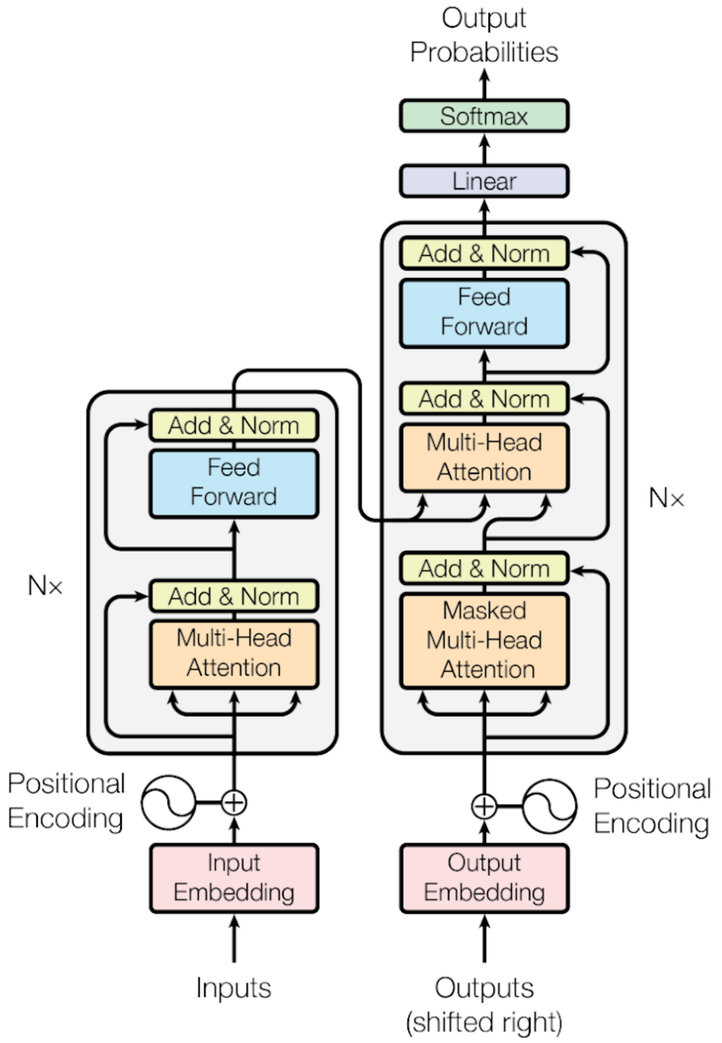
# Other Important Details

- Positional encodings
- Multi-head attention
- Layer normalization
- Decoder - masked self attention
- Unlike LSTM based models-



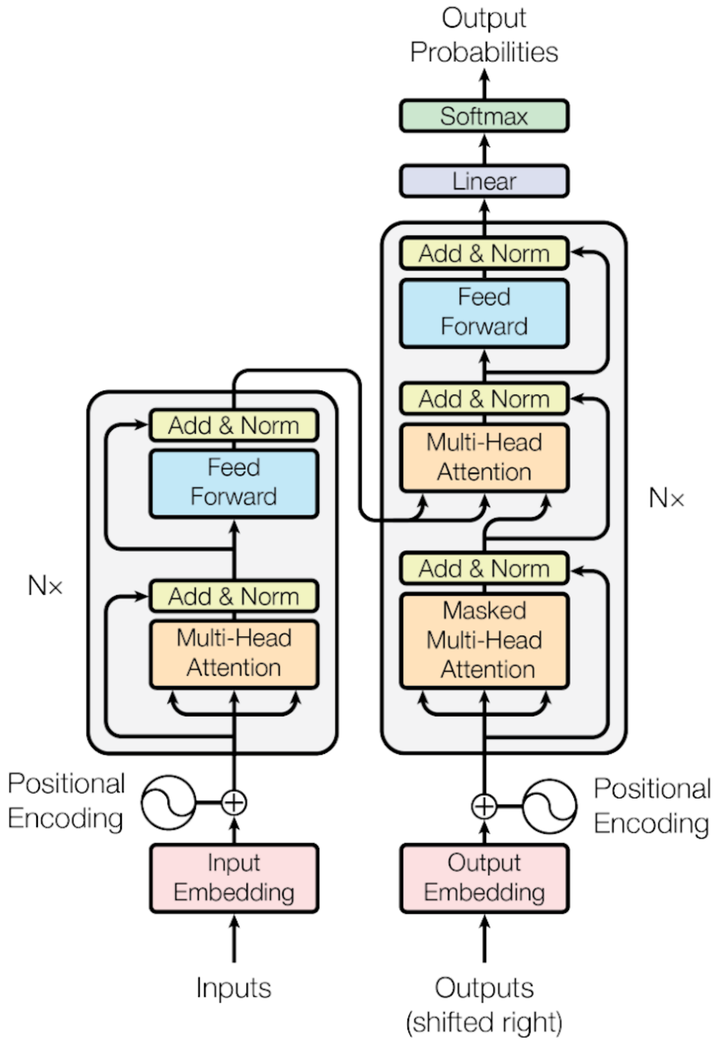
# Other Important Details

- Positional encodings
- Multi-head attention
- Layer normalization
- Decoder - masked self attention
- Unlike LSTM based models-
  - encoder-decoder-attention in each layer!



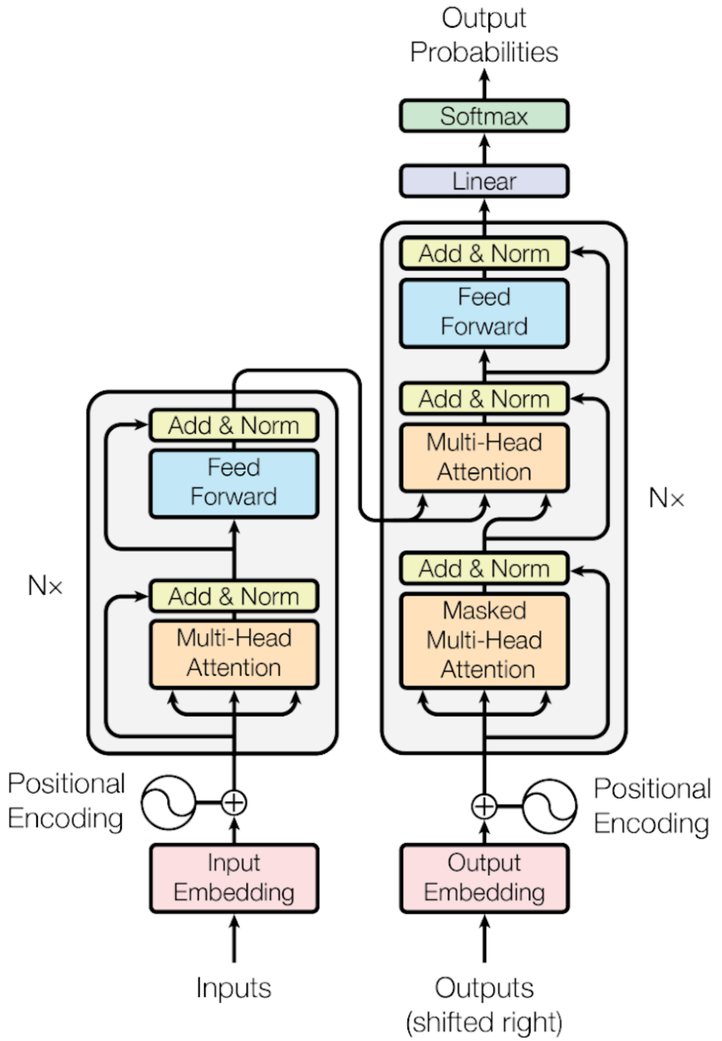
# Other Important Details

- Positional encodings
- Multi-head attention
- Layer normalization
- Decoder - masked self attention
- Unlike LSTM based models-
  - encoder-decoder-attention in each layer!
  - Less interpretable

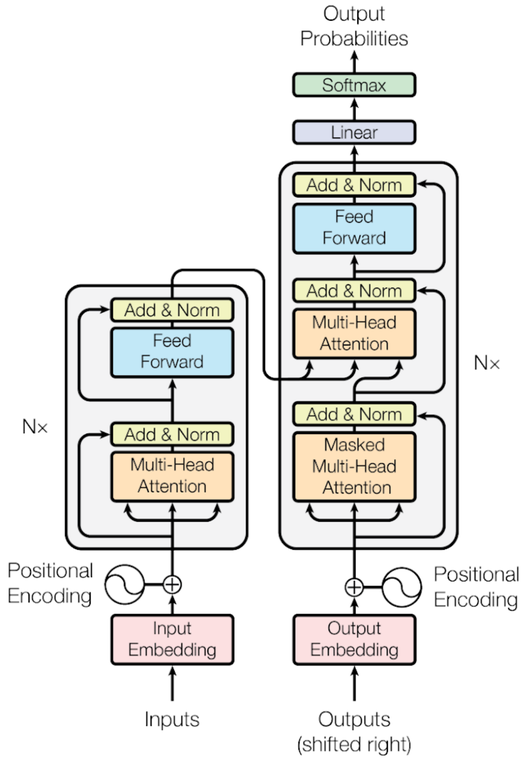


# Other Important Details

- Positional encodings
- Multi-head attention
- Layer normalization
- Decoder - masked self attention
- Unlike LSTM based models-
  - encoder-decoder-attention in each layer!
  - Less interpretable
- Learning rate schedule - more sensitive to hyperparams than LSTM-based models

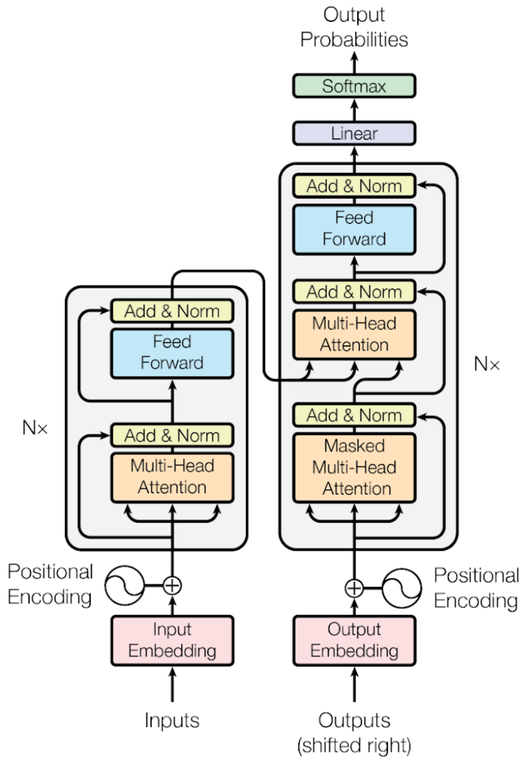


# Summary


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

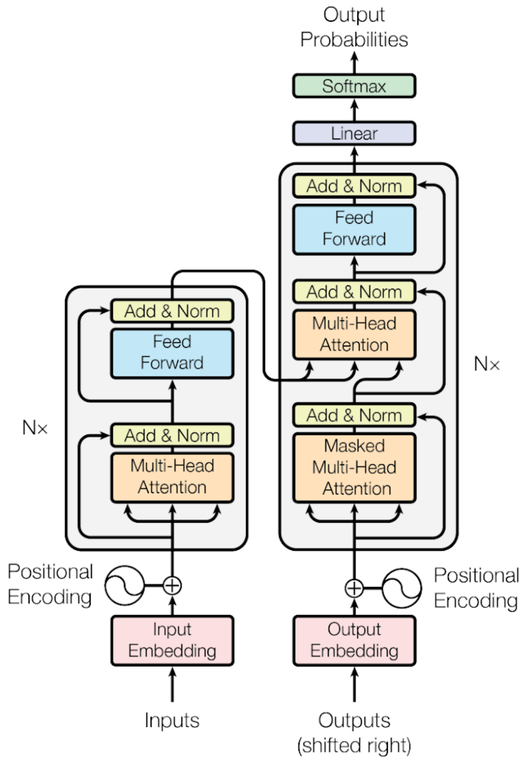
# Summary

- Neural networks are very useful for language modeling



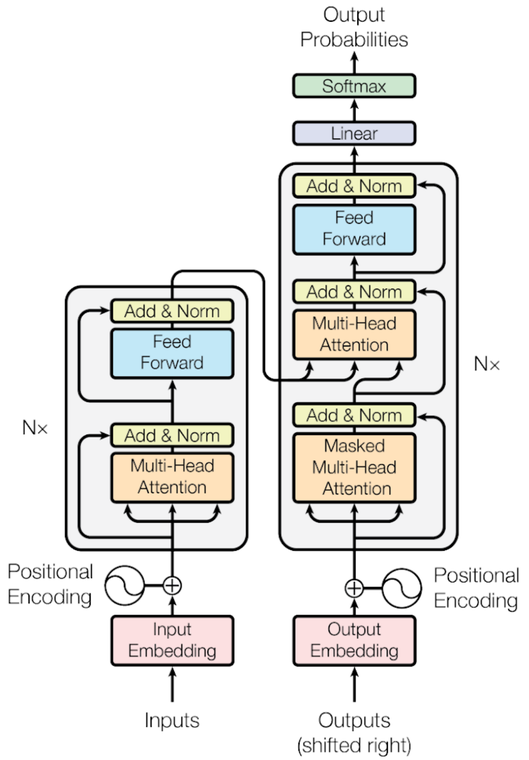
# Summary

- Neural networks are very useful for language modeling
- Better generalization, more context



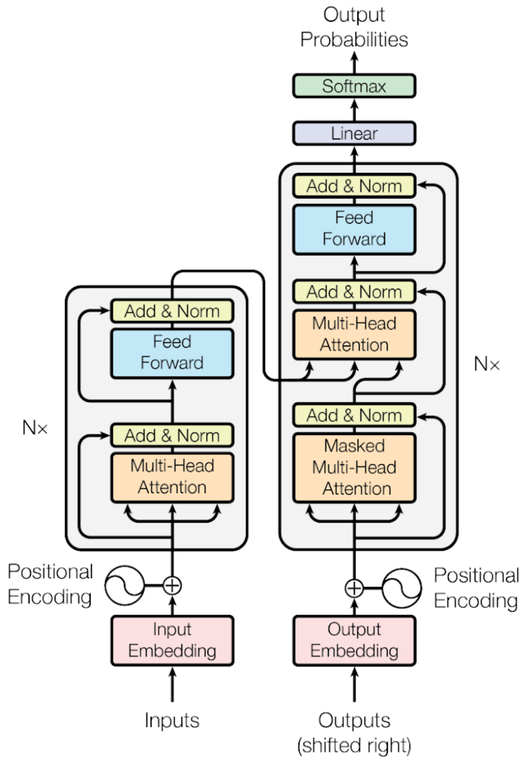
# Summary

- Neural networks are very useful for language modeling
- Better generalization, more context
- Neural language models: from MLPs to RNNs



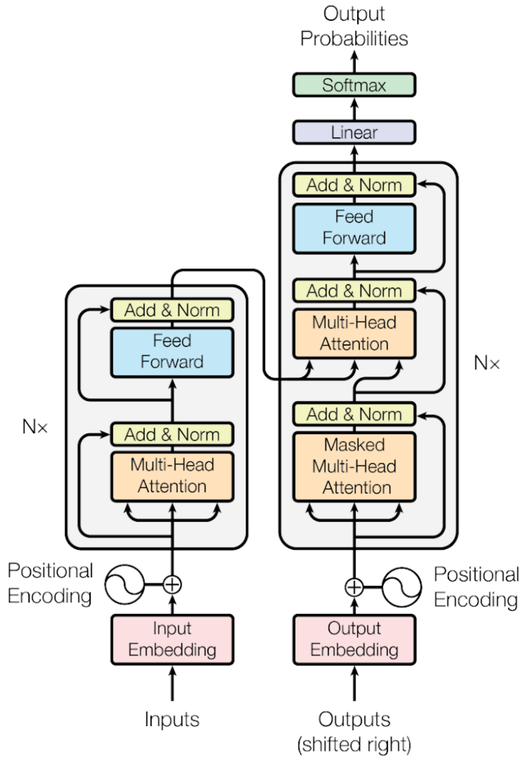
# Summary

- Neural networks are very useful for language modeling
- Better generalization, more context
- Neural language models: from MLPs to RNNs
- Sequence-to-Sequence Learning



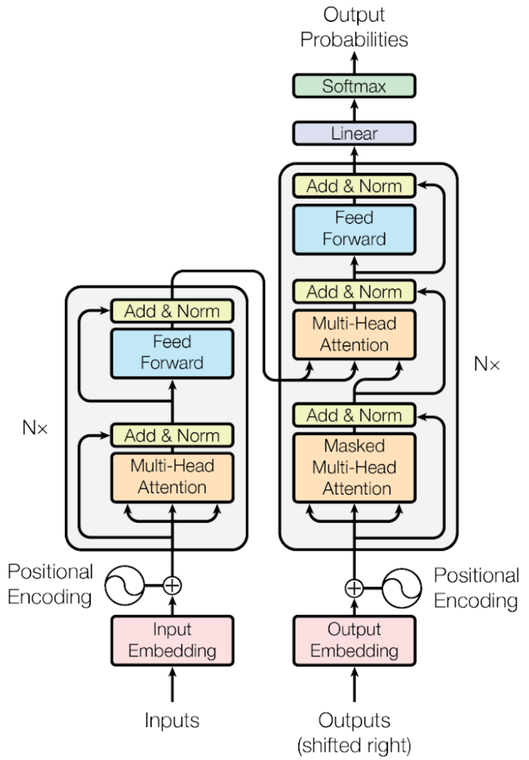
# Summary

- Neural networks are very useful for language modeling
- Better generalization, more context
- Neural language models: from MLPs to RNNs
- Sequence-to-Sequence Learning
- The Attention Mechanism

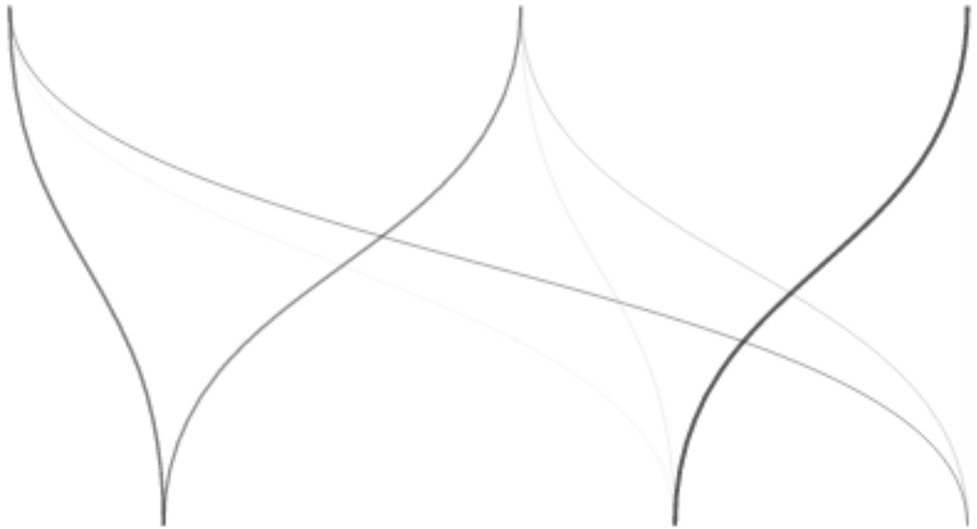


# Summary

- Neural networks are very useful for language modeling
- Better generalization, more context
- Neural language models: from MLPs to RNNs
- Sequence-to-Sequence Learning
- The Attention Mechanism
- The Transformer Architecture



Any Questions ?



Questions diverses ?