Deep learning for text processing

STAT 4710

November 29, 2022

Acknowledgement: Material drawn from Stanford's CS 224n class (http://web.stanford.edu/class/cs224n/index.html)

Where we are

- **Unit 1:** R for data mining
- **Unit 2:** Prediction fundamentals
- **Unit 3:** Regression-based methods
- **Unit 4:** Tree-based methods
- **Unit 5:** Deep learning

Lecture 1: Deep learning preliminaries

Lecture 2: Neural networks

Lecture 3: Deep learning for images

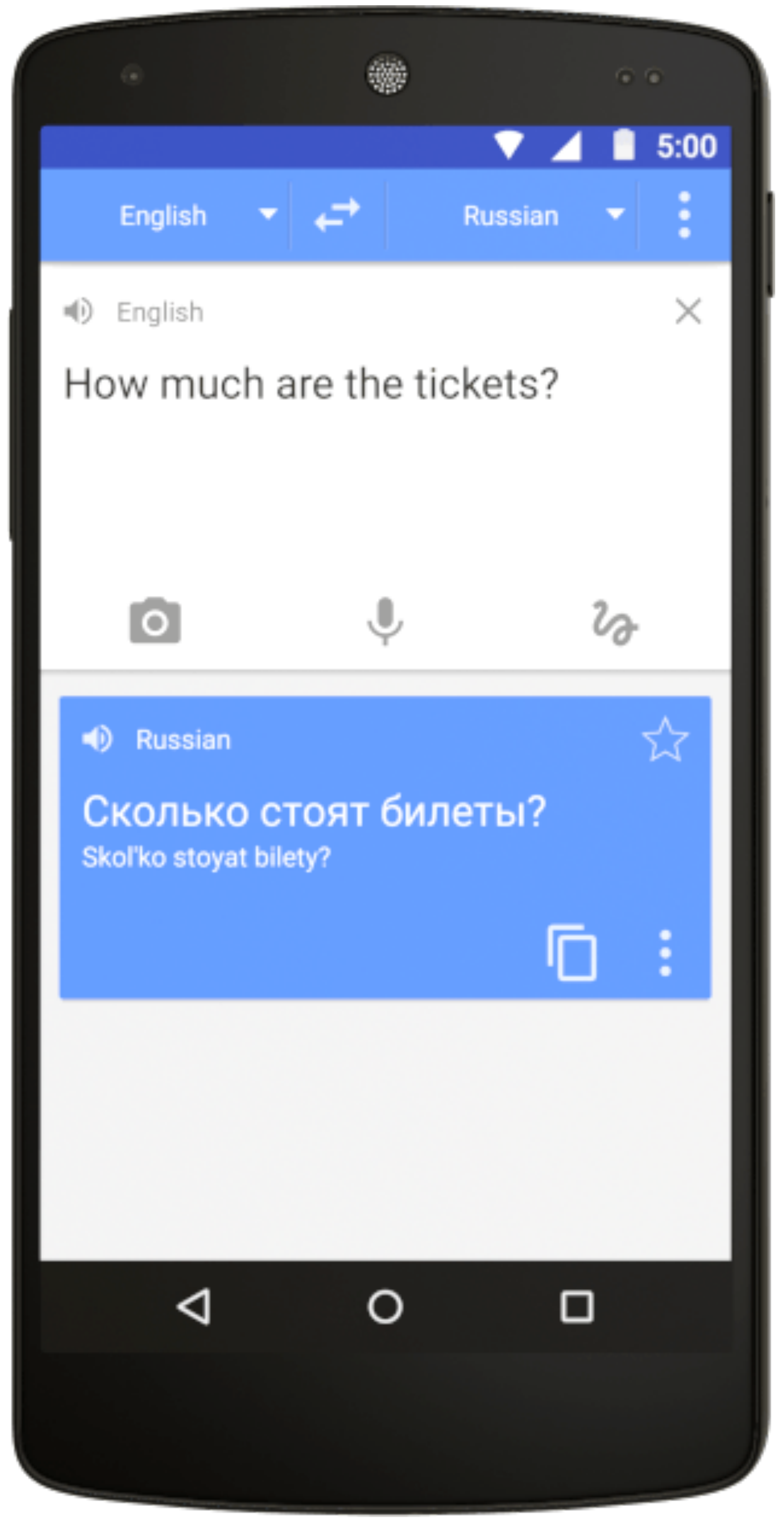
Lecture 4: Deep learning for text

Lecture 5: Unit review and quiz in class

Applications of natural language processing

Applications of natural language processing

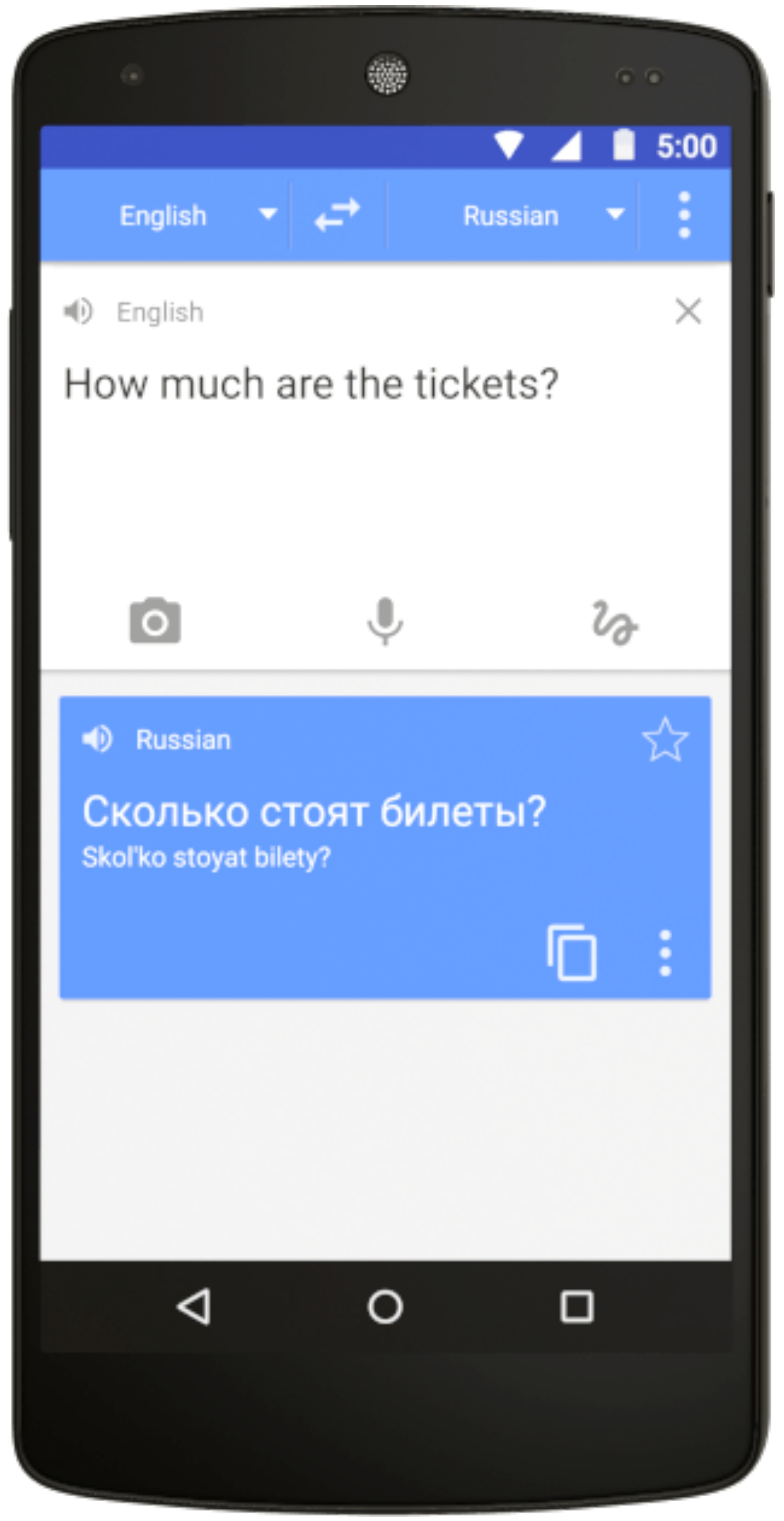
Machine translation



https://translate.google.com/intl/en/about/

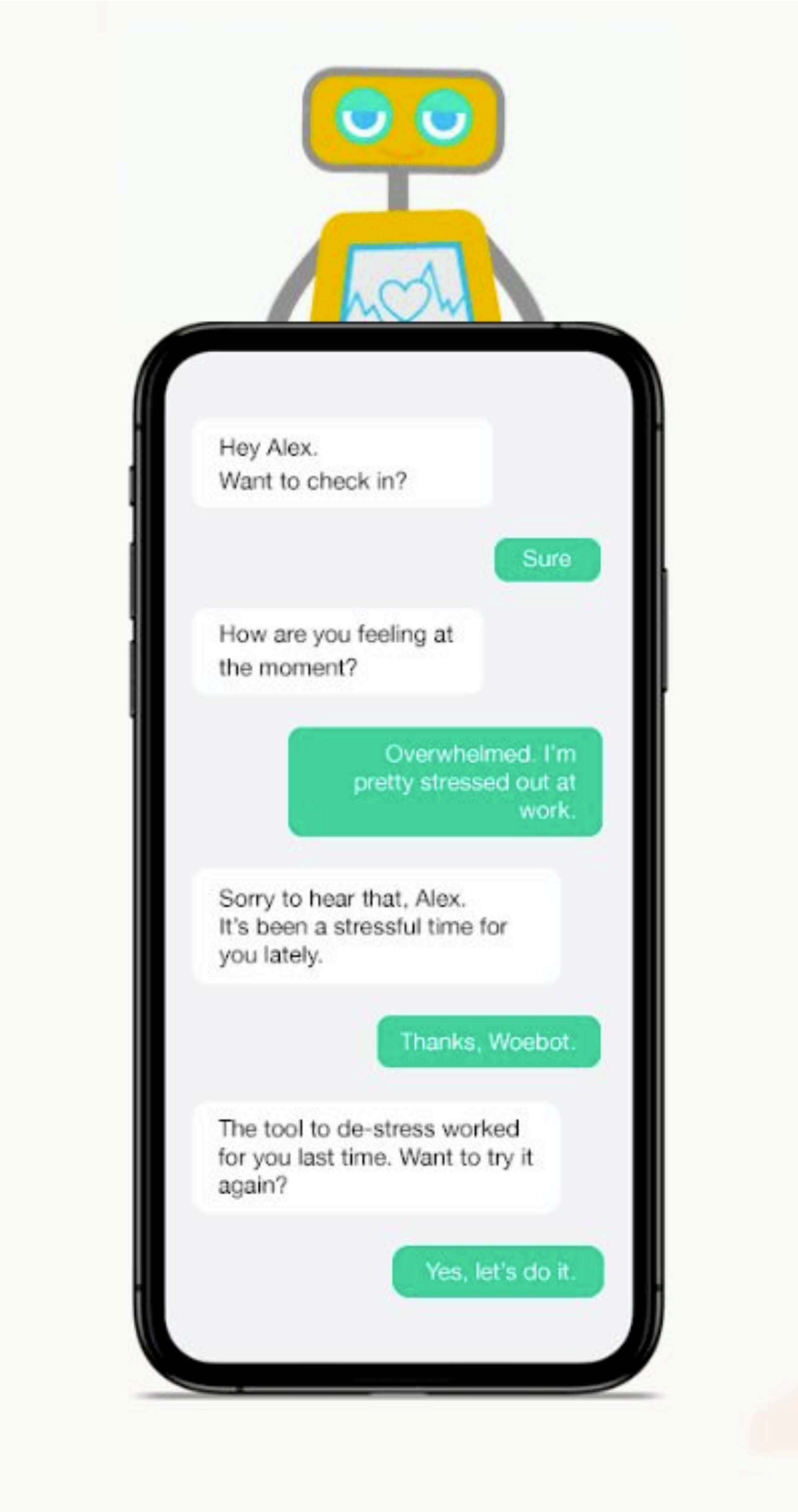
Applications of natural language processing

Machine translation



https://translate.google.com/intl/en/about/

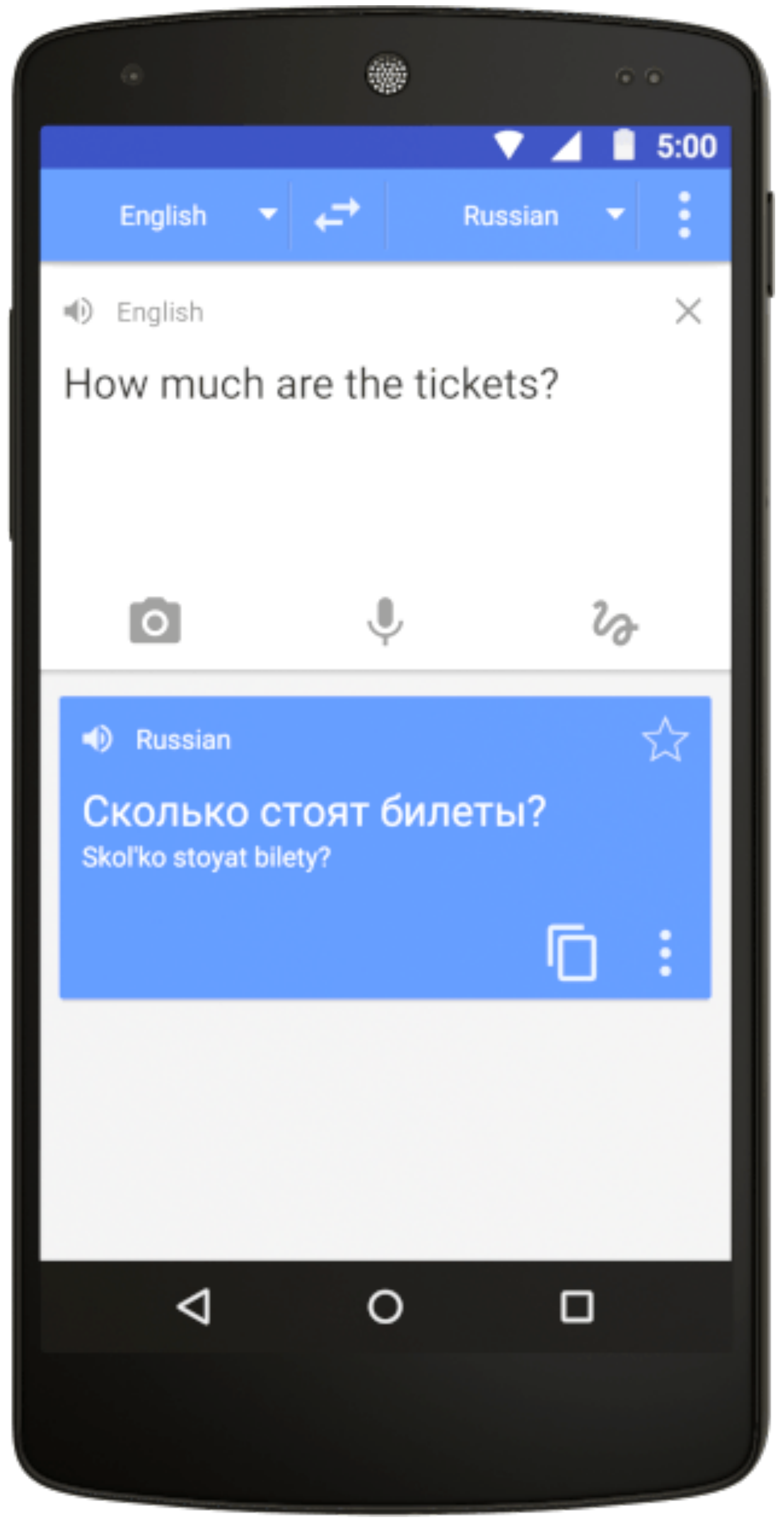
Chatbots



https://www.trendhunter.com/trends/woebot

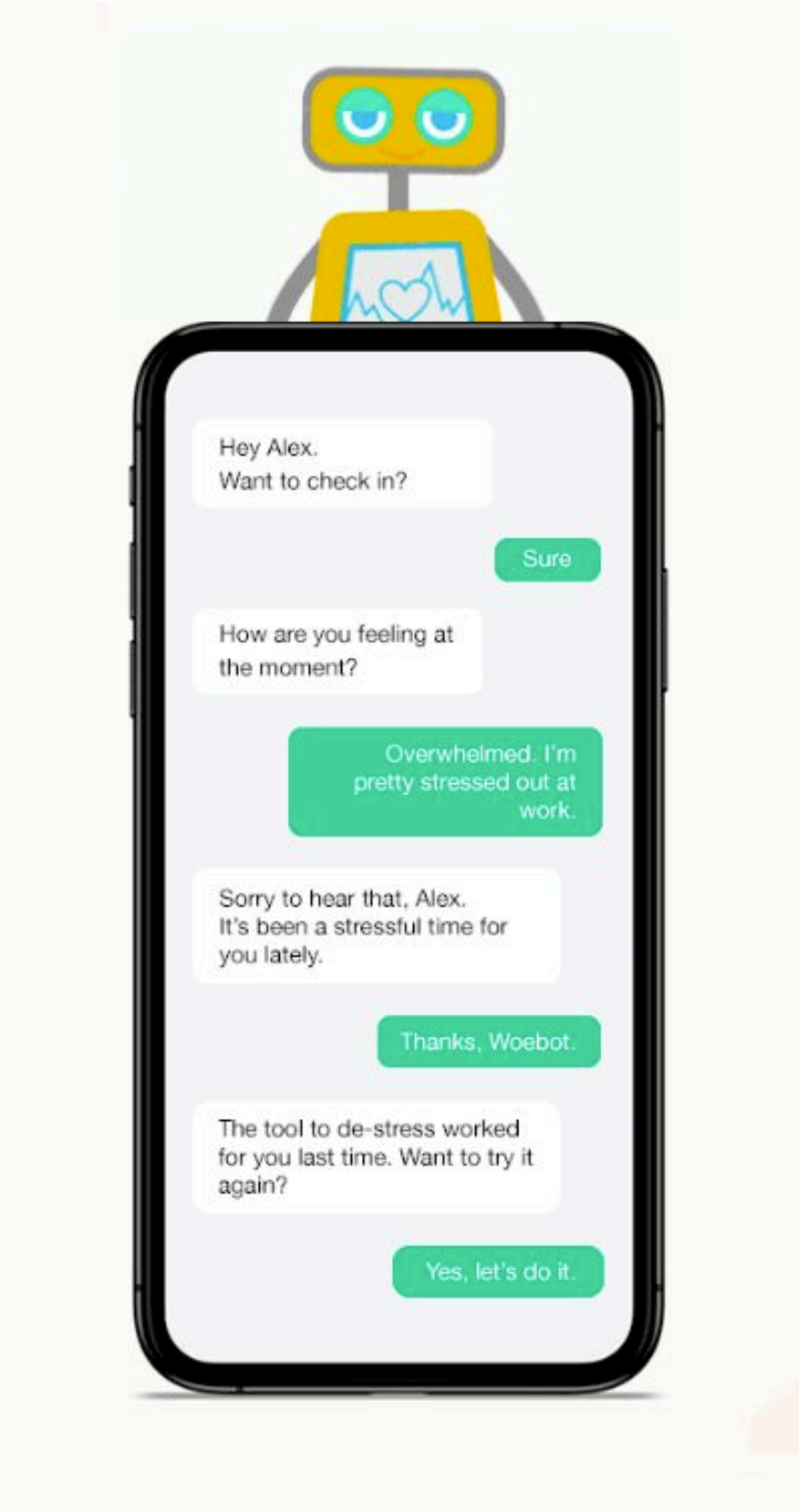
Applications of natural language processing

Machine translation



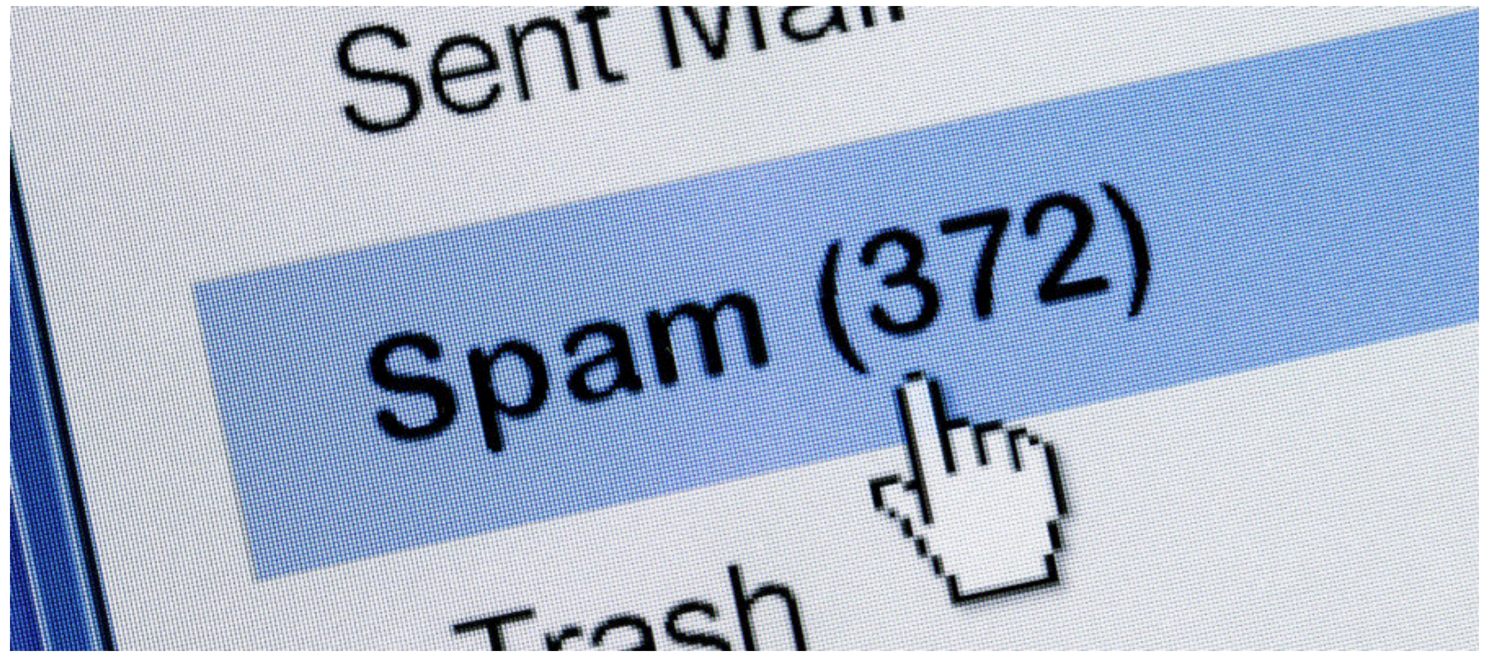
https://translate.google.com/intl/en/about/

Chatbots



https://www.trendhunter.com/trends/woebot

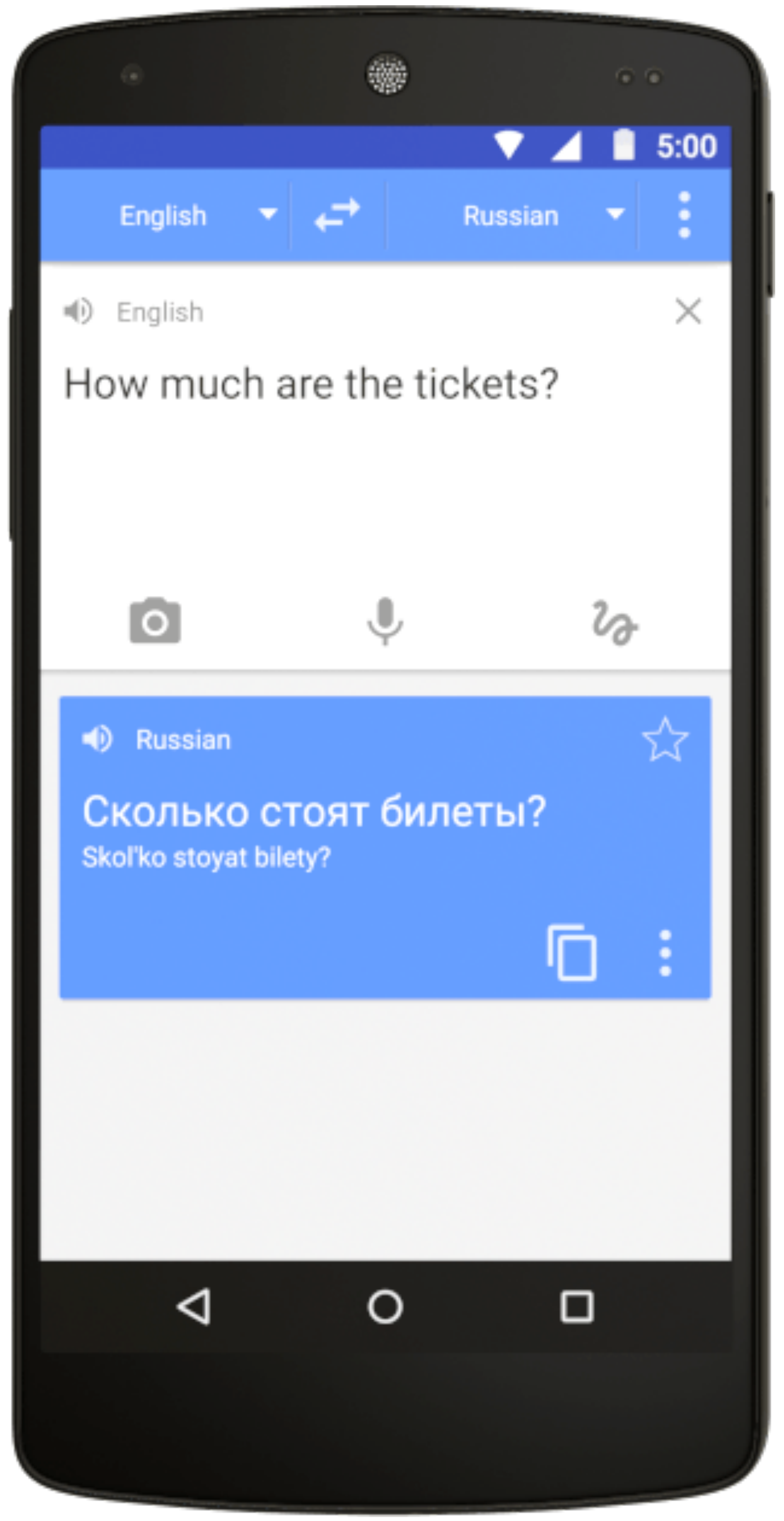
Spam filtering



https://blog.malwarebytes.com/security-world/2017/02/explained-bayesian-spam-filtering/

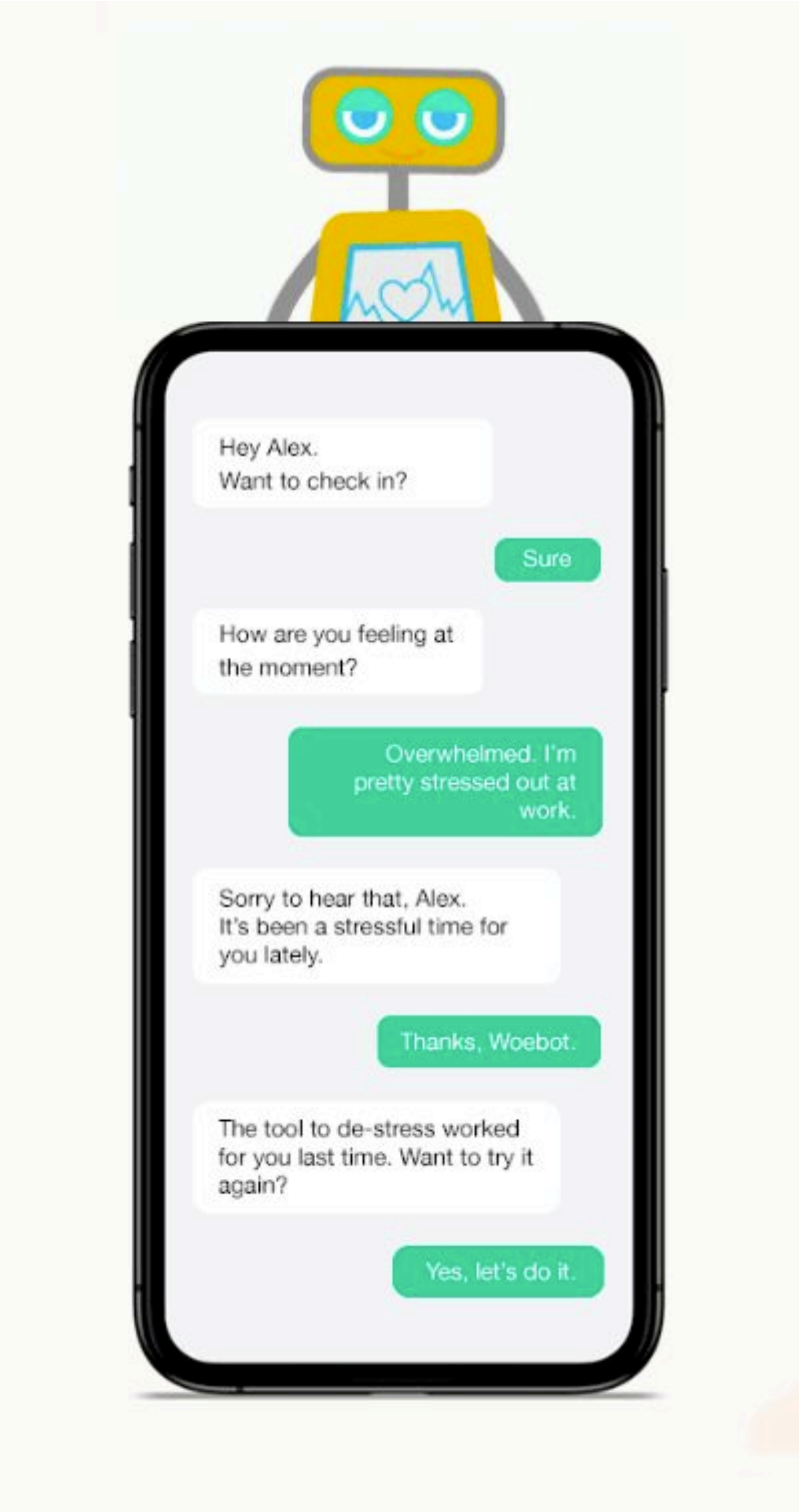
Applications of natural language processing

Machine translation



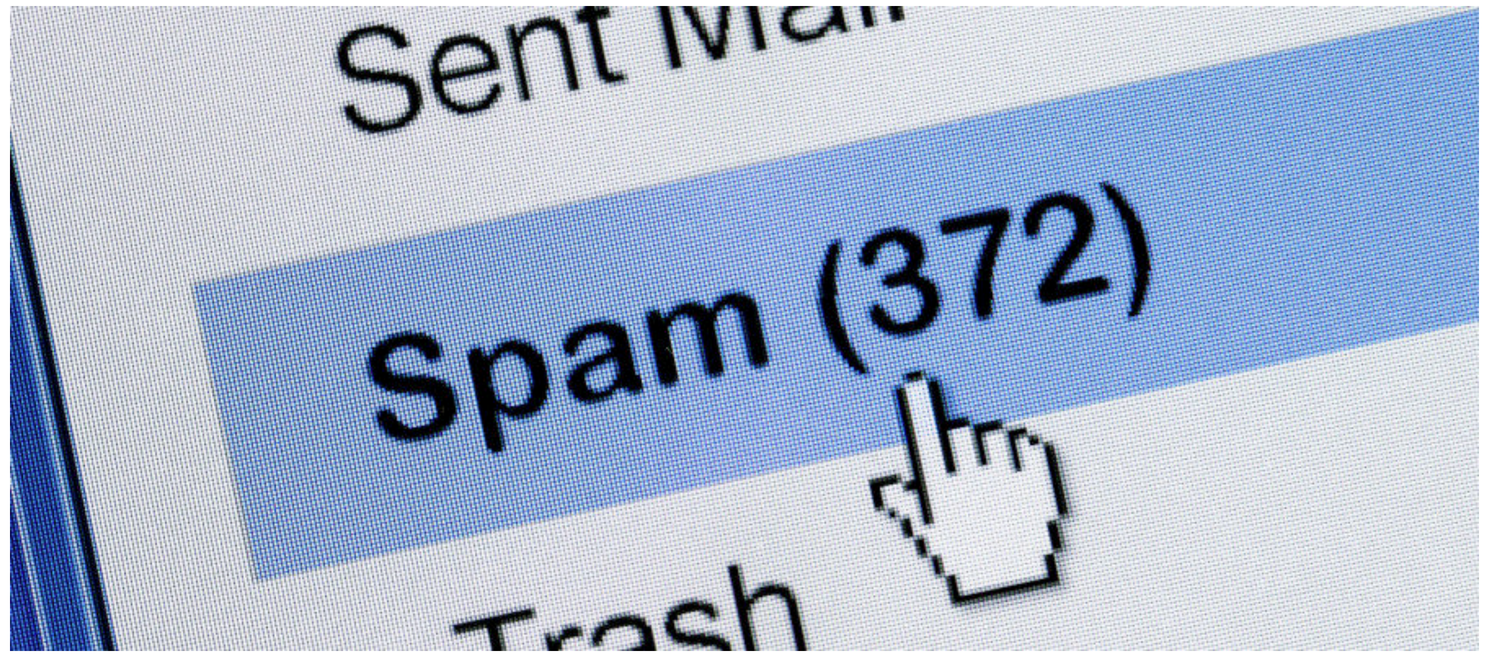
https://translate.google.com/intl/en/about/

Chatbots



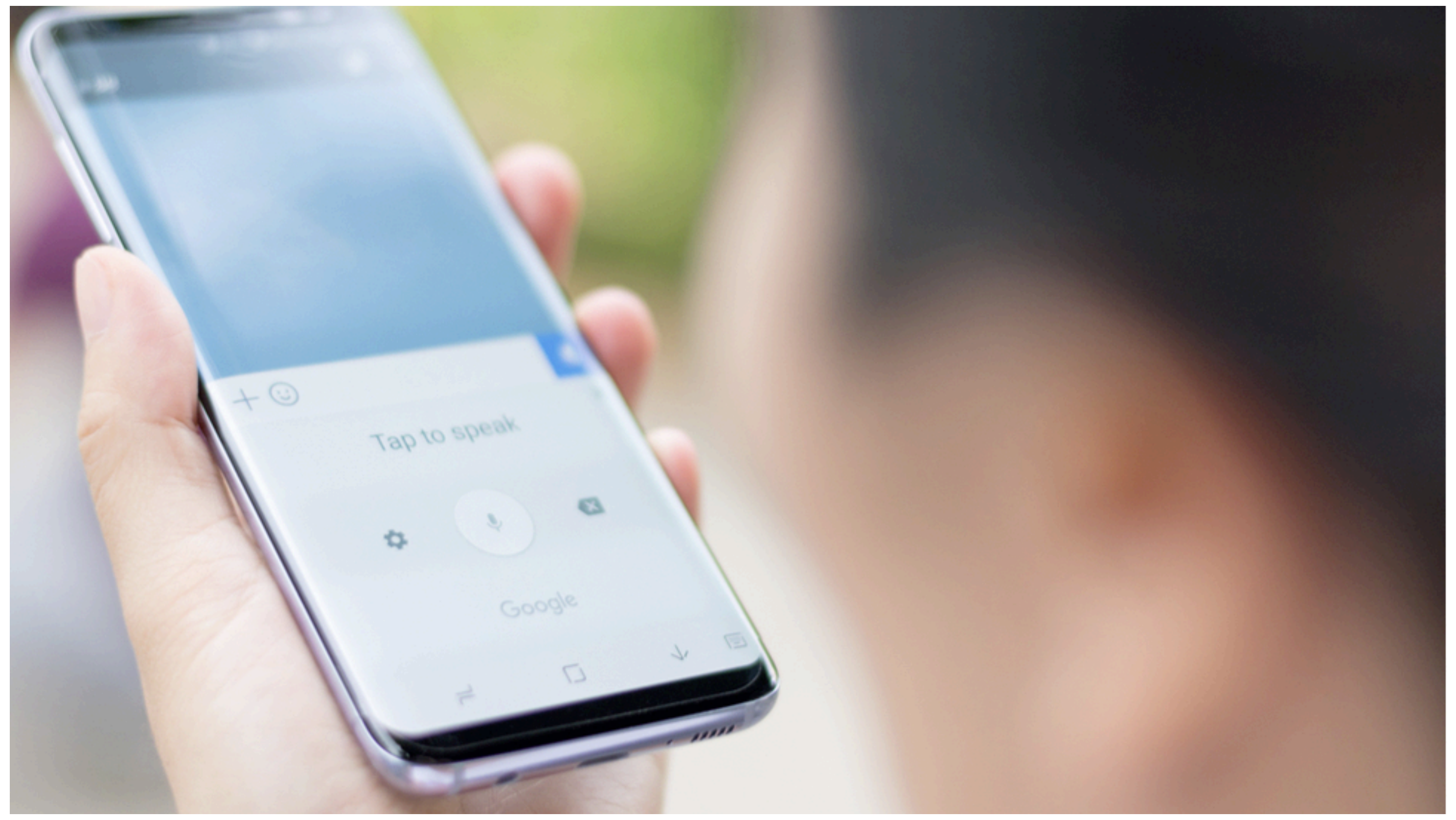
https://www.trendhunter.com/trends/woebot

Spam filtering



https://blog.malwarebytes.com/security-world/2017/02/explained-bayesian-spam-filtering/

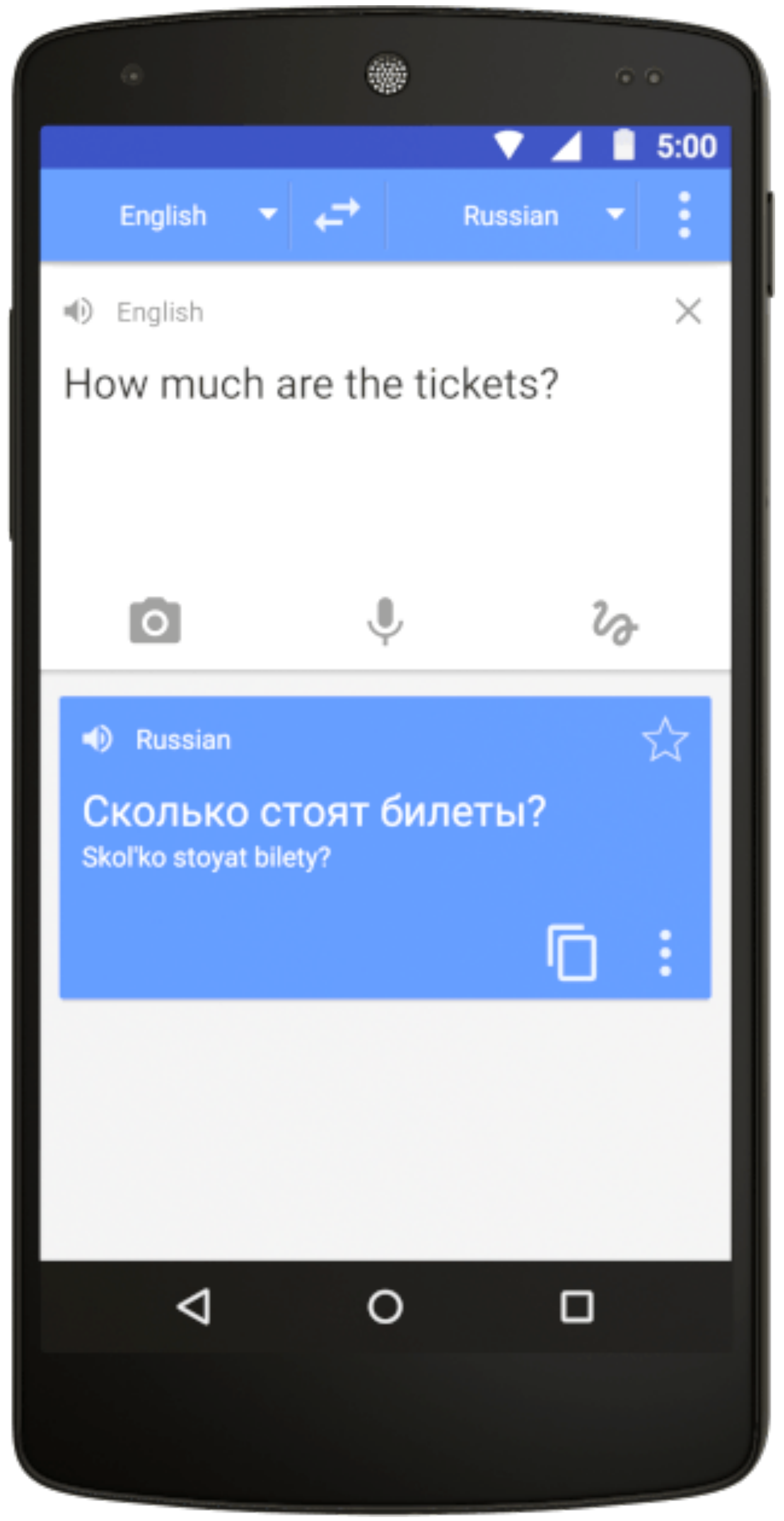
Voice to text



https://www.itproportal.com/guides/how-to-turn-off-ok-google-android-voice-search/

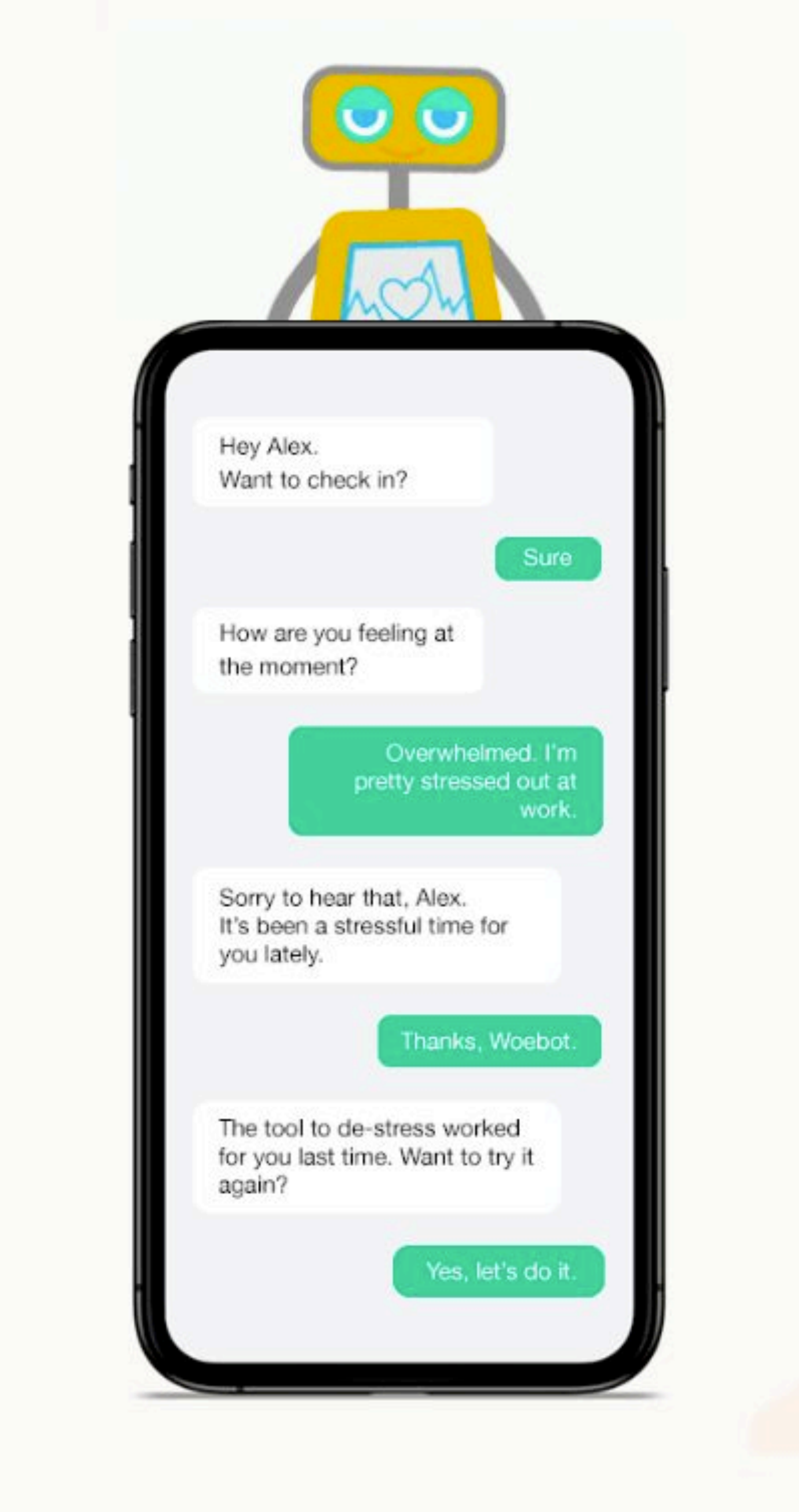
Applications of natural language processing

Machine translation



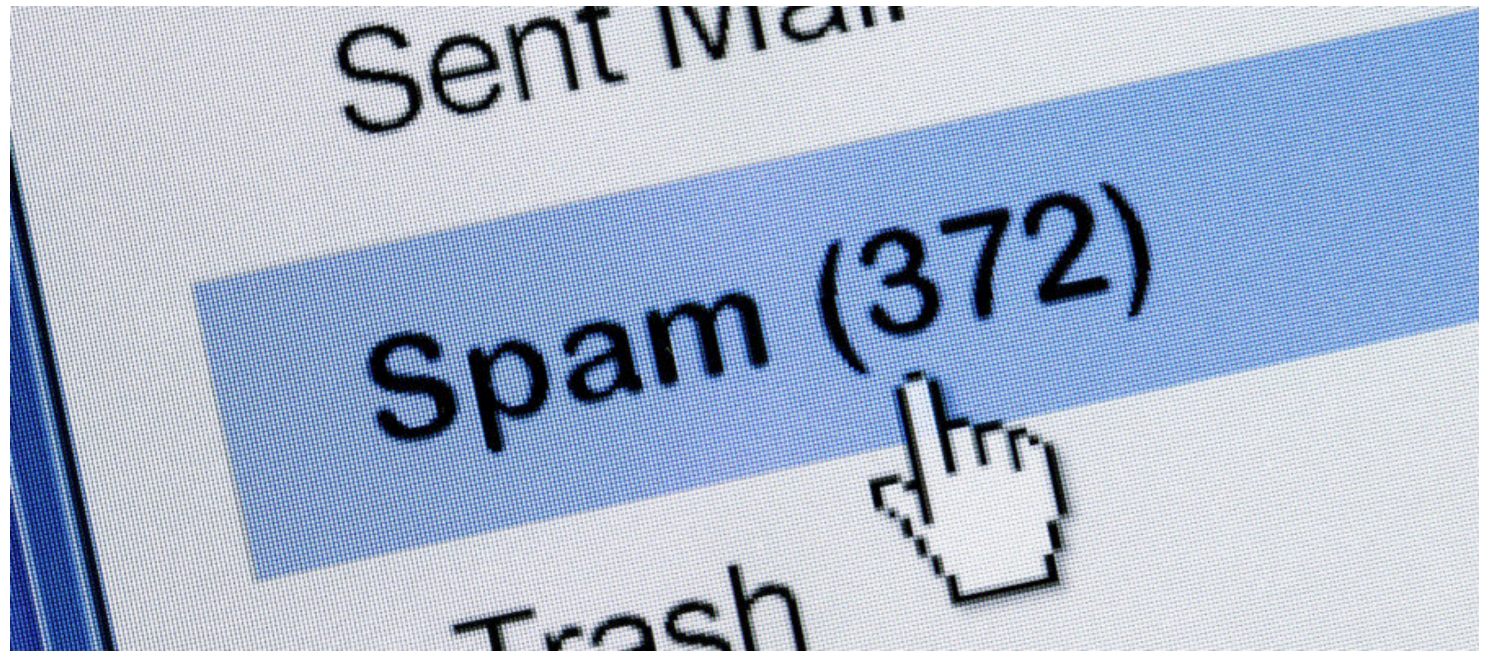
https://translate.google.com/intl/en/about/

Chatbots



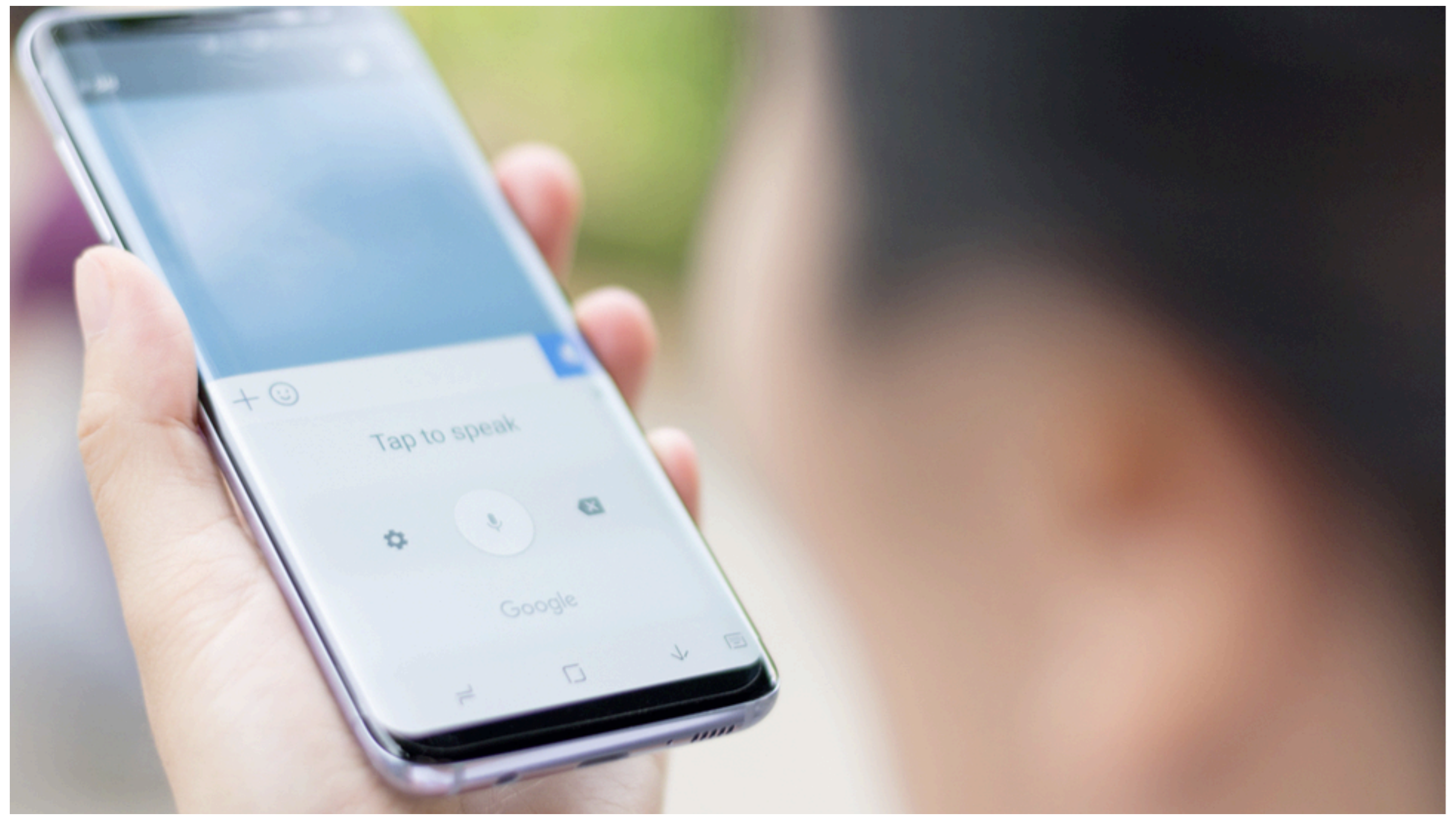
https://www.trendhunter.com/trends/woebot

Spam filtering



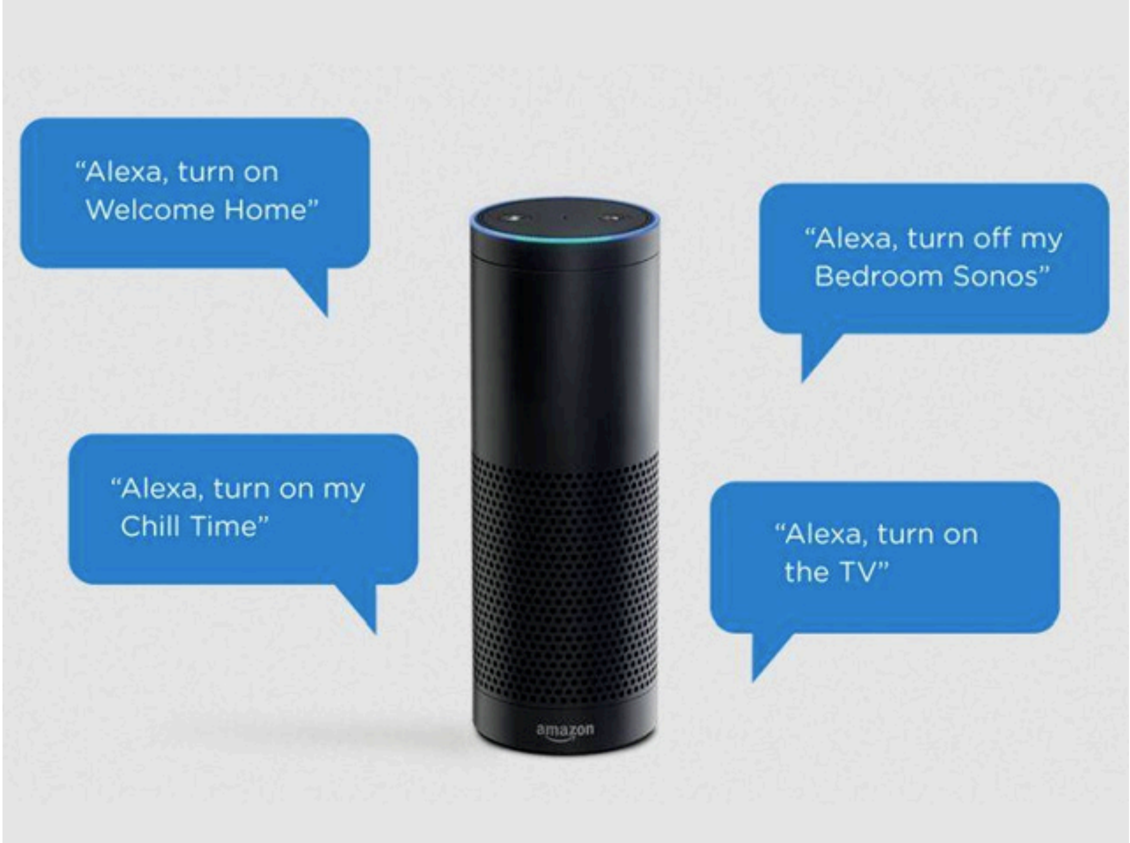
https://blog.malwarebytes.com/security-world/2017/02/explained-bayesian-spam-filtering/

Voice to text



https://www.itproportal.com/guides/how-to-turn-off-ok-google-android-voice-search/

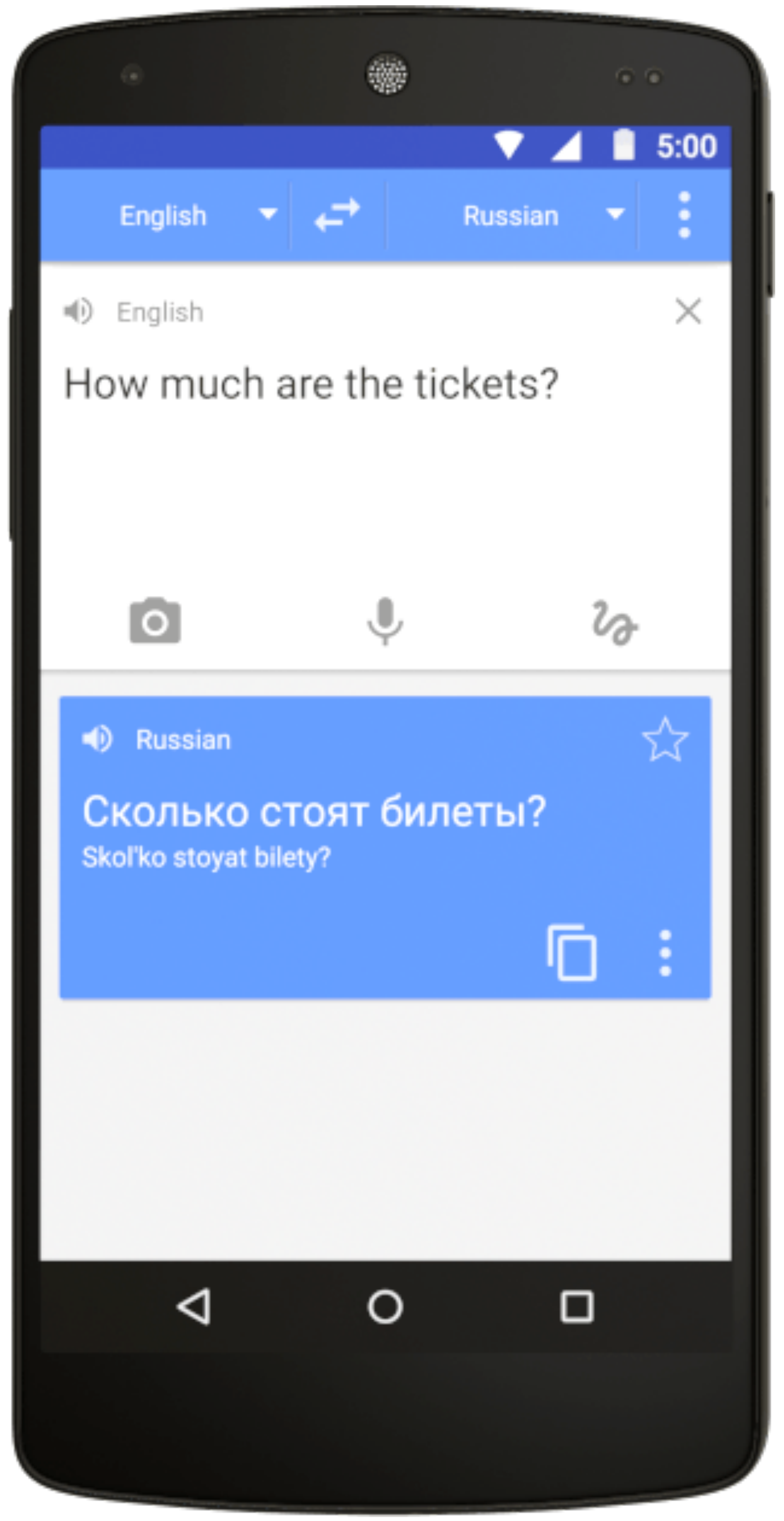
Personal assistant



https://brailleinstitute.org/event/online-introducing-amazon-alexa

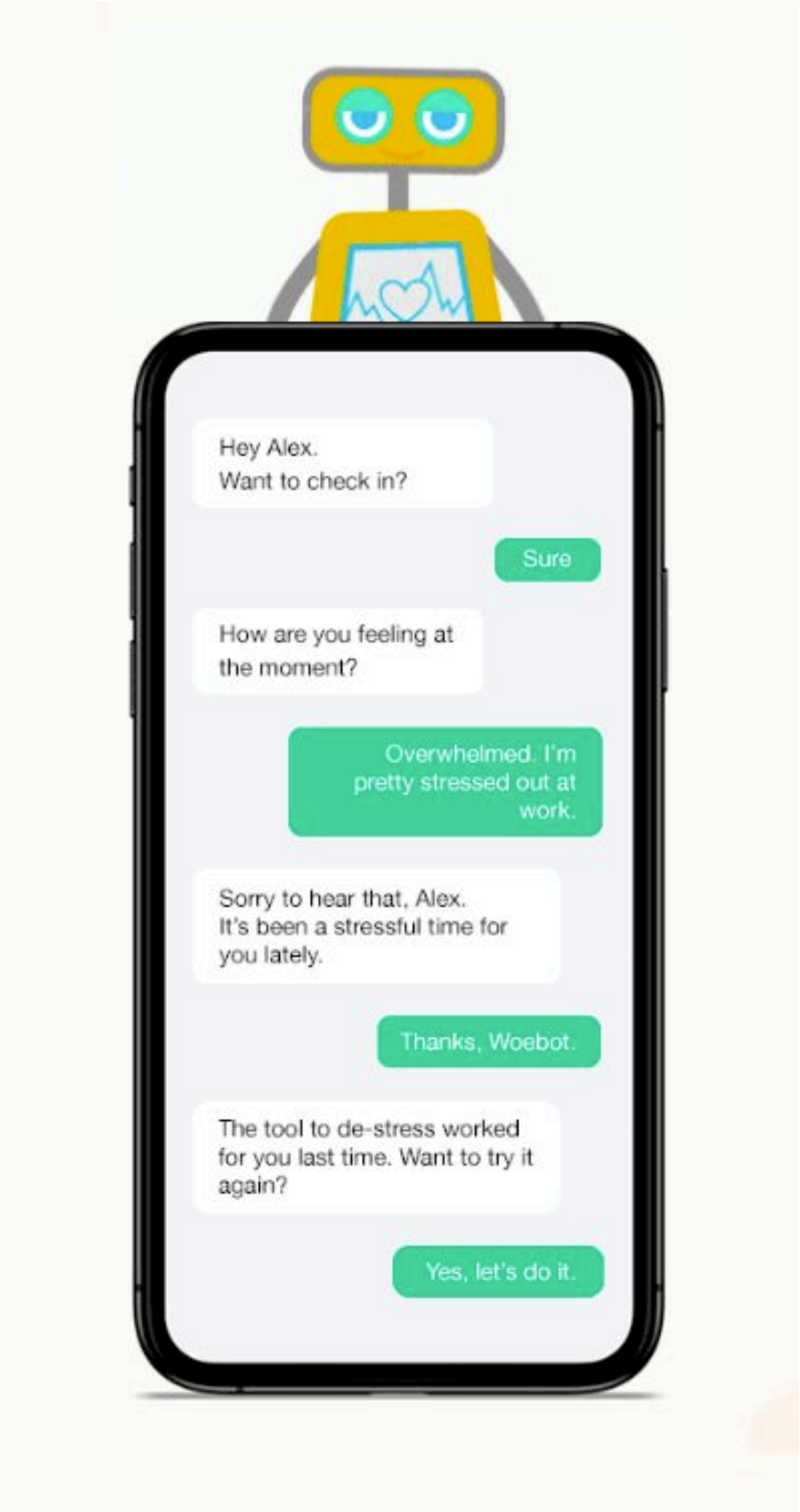
Applications of natural language processing

Machine translation



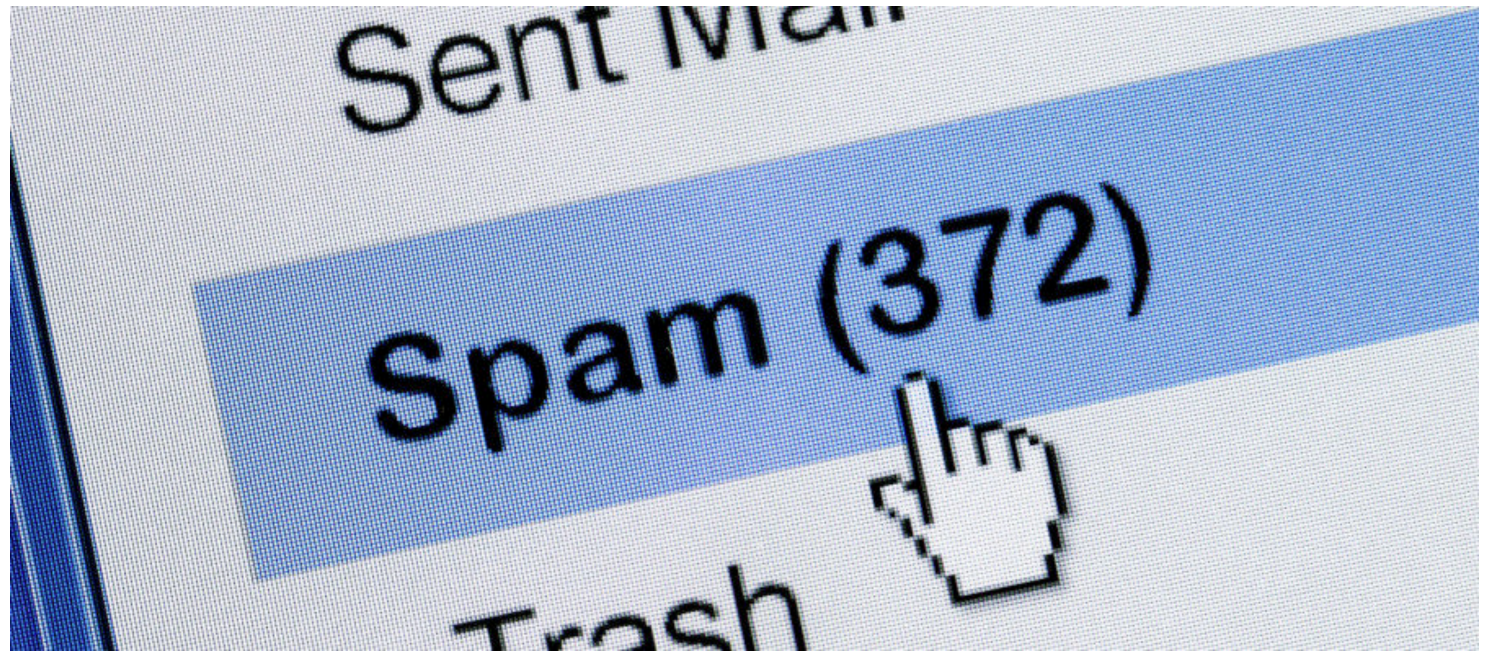
https://translate.google.com/intl/en/about/

Chatbots



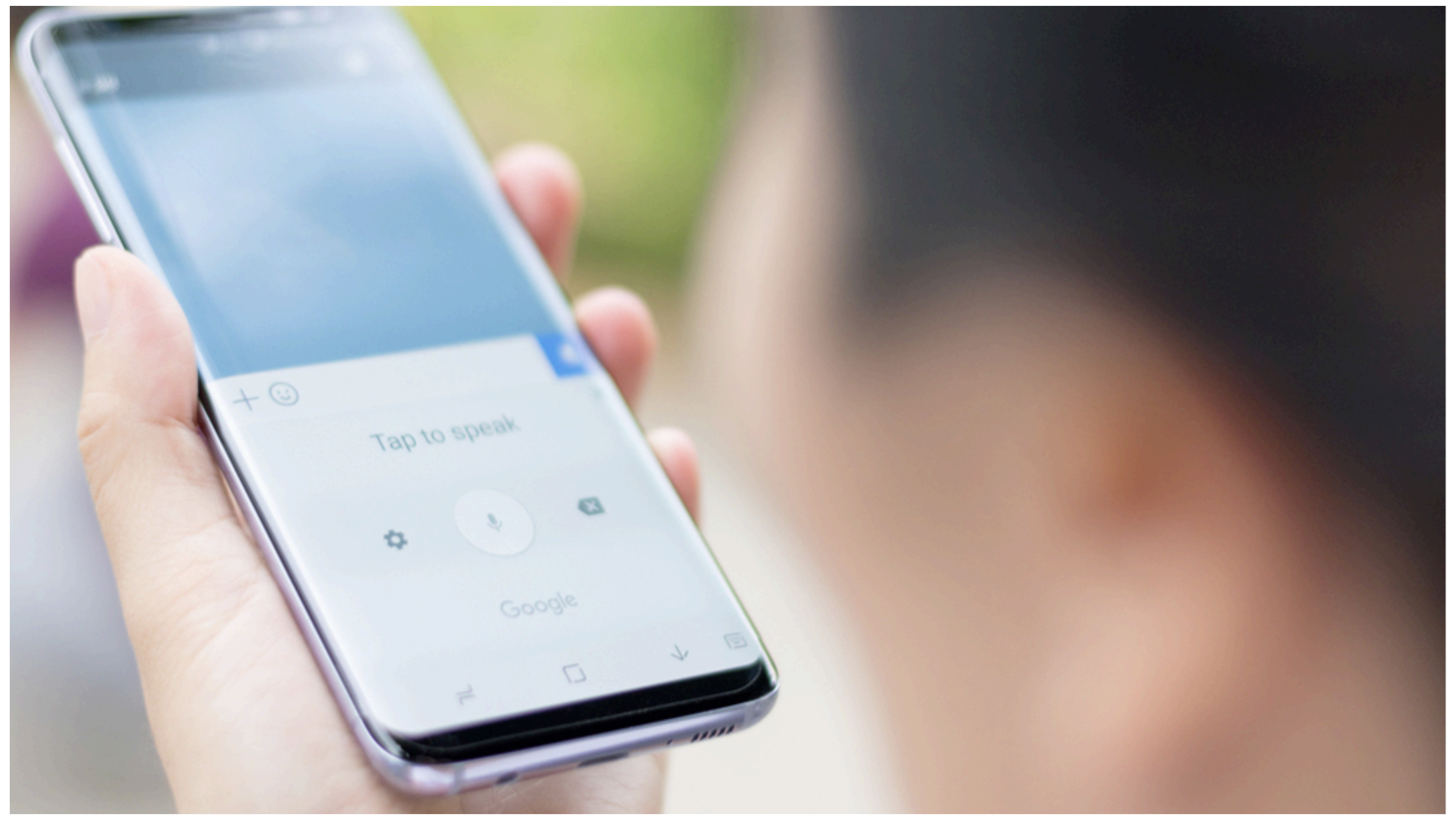
https://www.trendhunter.com/trends/woebot

Spam filtering



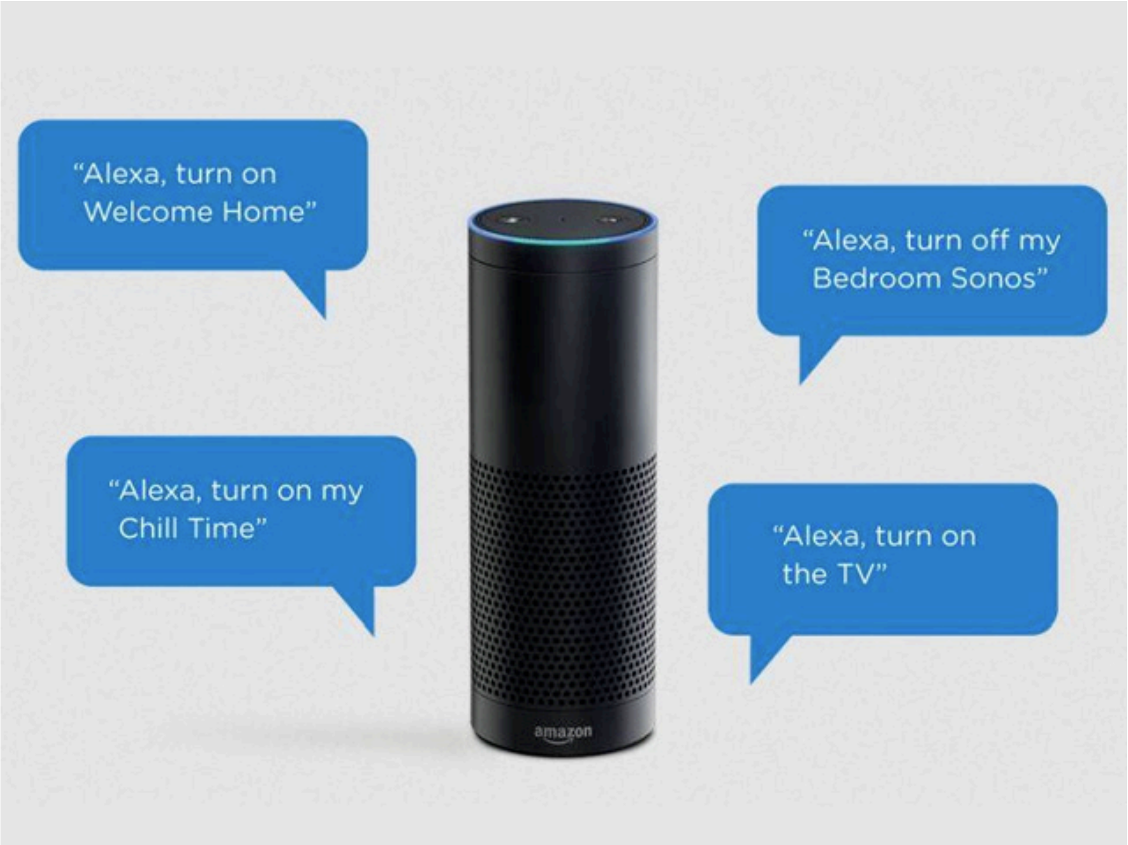
https://blog.malwarebytes.com/security-world/2017/02/explained-bayesian-spam-filtering/

Voice to text



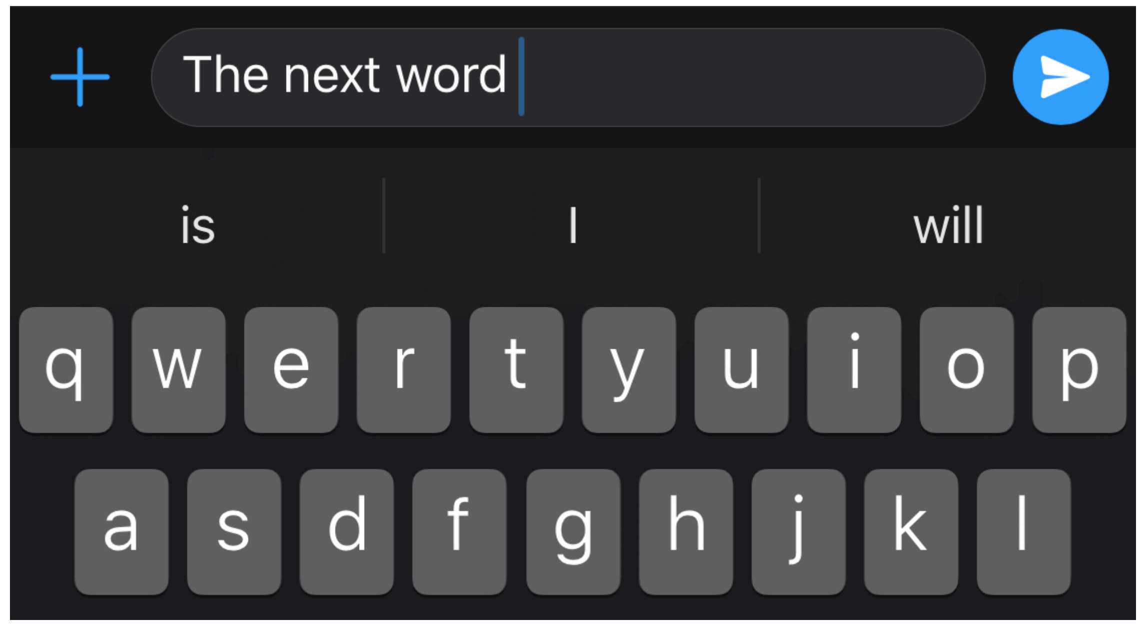
https://www.itproportal.com/guides/how-to-turn-off-ok-google-android-voice-search/

Personal assistant



https://brailleinstitute.org/event/online-introducing-amazon-alexa

Auto-complete



https://towardsdatascience.com/language-modeling-c1cf7b983685

Word vectors

The basic units of images are pixels, which automatically have numeric values associated with them.

On the other hand, the basic units of language are words, which do not come with a pre-packaged vector representation.

Word vectors

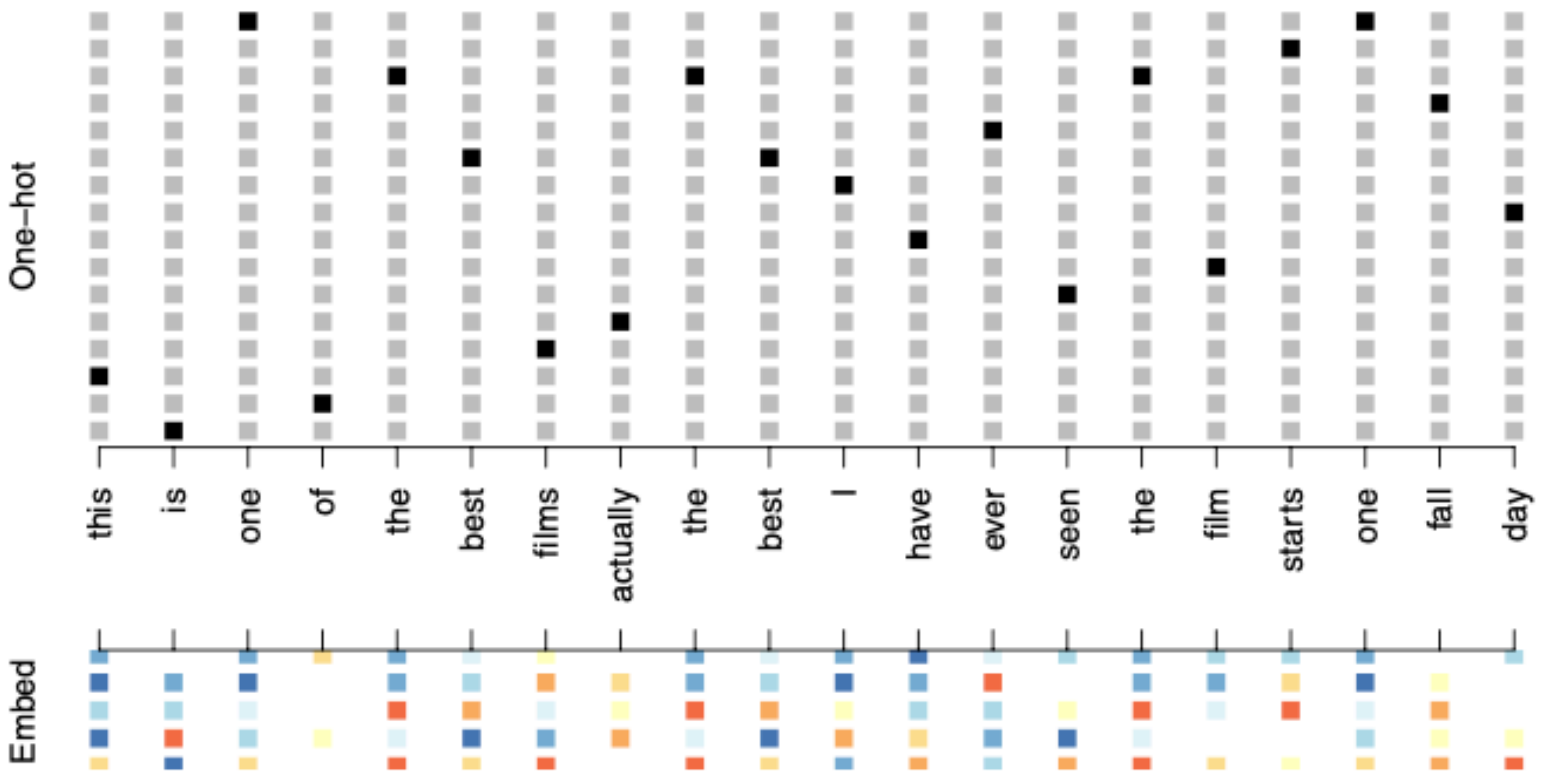
The basic units of images are pixels, which automatically have numeric values associated with them.

On the other hand, the basic units of language are words, which do not come with a pre-packaged vector representation.

A common first step in deep learning for NLP is to encode each word using a word vector, e.g.

$$v_{\text{man}} = (0.2, 1.6, 2.5, \ldots, 4.1)$$

Word vectors are useful if they capture the meaning of the words they represent.



Source: ISLRv2

Constructing word vectors

Example: word2vec

Constructing word vectors

Example: word2vec

Idea: If two words w_1 and w_2 near each other often (e.g. within 5 words of each other), then they should have similar word vectors v_1 and v_2.

Constructing word vectors

Example: word2vec

Idea: If two words w_1 and w_2 near each other often (e.g. within 5 words of each other), then they should have similar word vectors v_1 and v_2.

Suppose for a word w_1, we model the relative frequencies of nearby words w_2 as

$$\mathbb{P}[w_2 \mid w_1] = \frac{\exp(v_1^T v_2)}{\sum_{\text{all word vectors } v} \exp(v_1^T v)}.$$

Constructing word vectors

Example: word2vec

Idea: If two words w_1 and w_2 near each other often (e.g. within 5 words of each other), then they should have similar word vectors v_1 and v_2.

Suppose for a word w_1, we model the relative frequencies of nearby words w_2 as

$$\mathbb{P}[w_2 \mid w_1] = \frac{\exp(v_1^T v_2)}{\sum_{\text{all word vectors } v} \exp(v_1^T v)}.$$

Can then train the word vectors $v_1, v_2, \ldots$ based on a text corpus by maximizing the likelihood of all the word co-occurrence probabilities.

Constructing word vectors

Example: word2vec

Idea: If two words w_1 and w_2 near each other often (e.g. within 5 words of each other), then they should have similar word vectors v_1 and v_2.

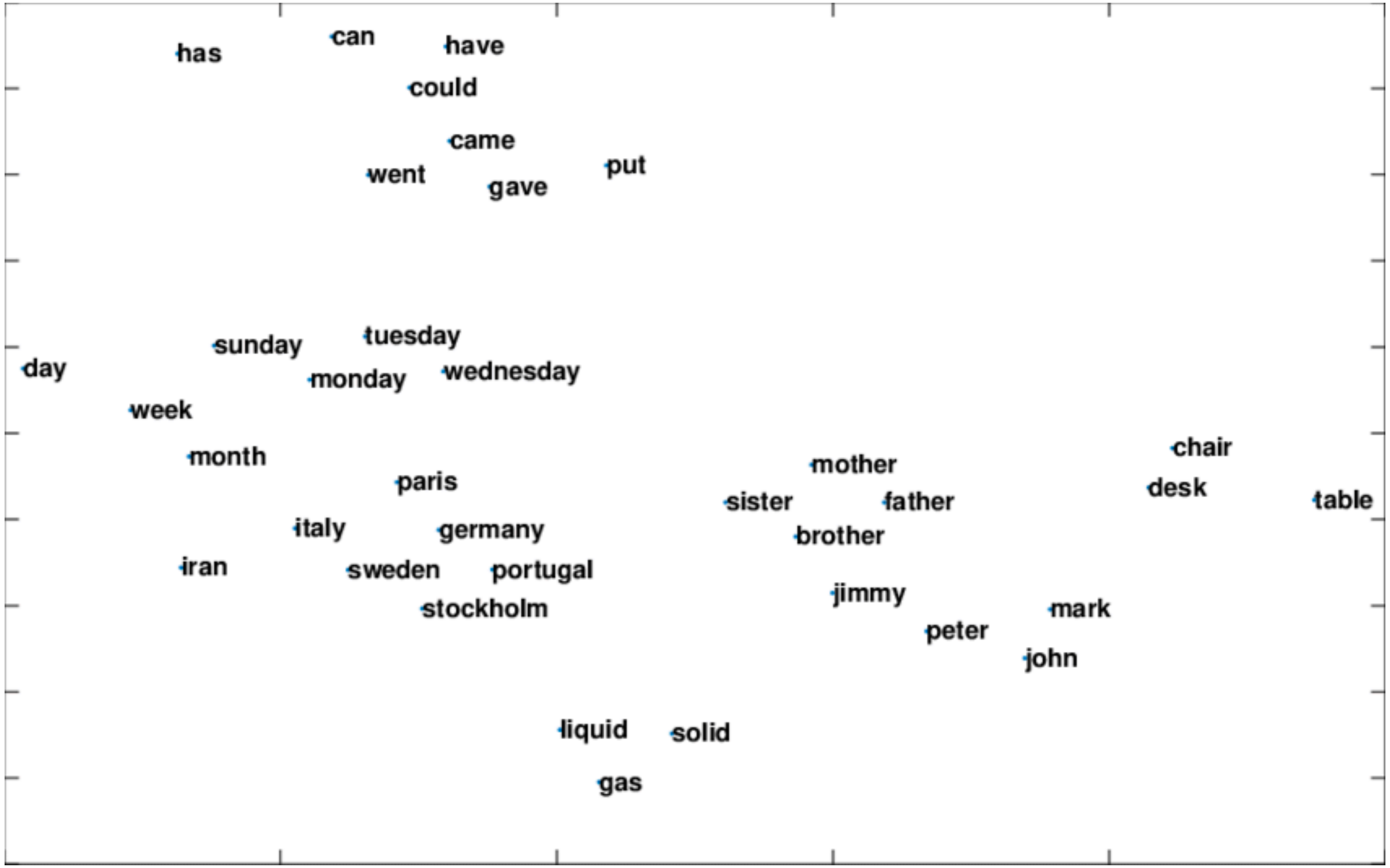
Suppose for a word w_1, we model the relative frequencies of nearby words w_2 as

$$\mathbb{P}[w_2 \mid w_1] = \frac{\exp(v_1^T v_2)}{\sum_{\text{all word vectors } v} \exp(v_1^T v)}.$$

Can then train the word vectors $v_1, v_2, \ldots$ based on a text corpus by maximizing the likelihood of all the word co-occurrence probabilities.

Word vectors need to be trained only once (for each language), and can be reused thereafter. For example, word2vec vectors for a few dozen languages can be downloaded at http://vectors.nlpl.eu/repository/.

Word vectors capture semantic relationships



https://www.researchgate.net/figure/A-two-dimensional-representation-of-word-embeddings-Words-with-similar-meanings-are_fig1_327074728

Case study 1: Named entity classification

Task: Classify the word in the middle of a window as either a person, location, organization, or other, e.g.

"…museums in Paris are amazing…"

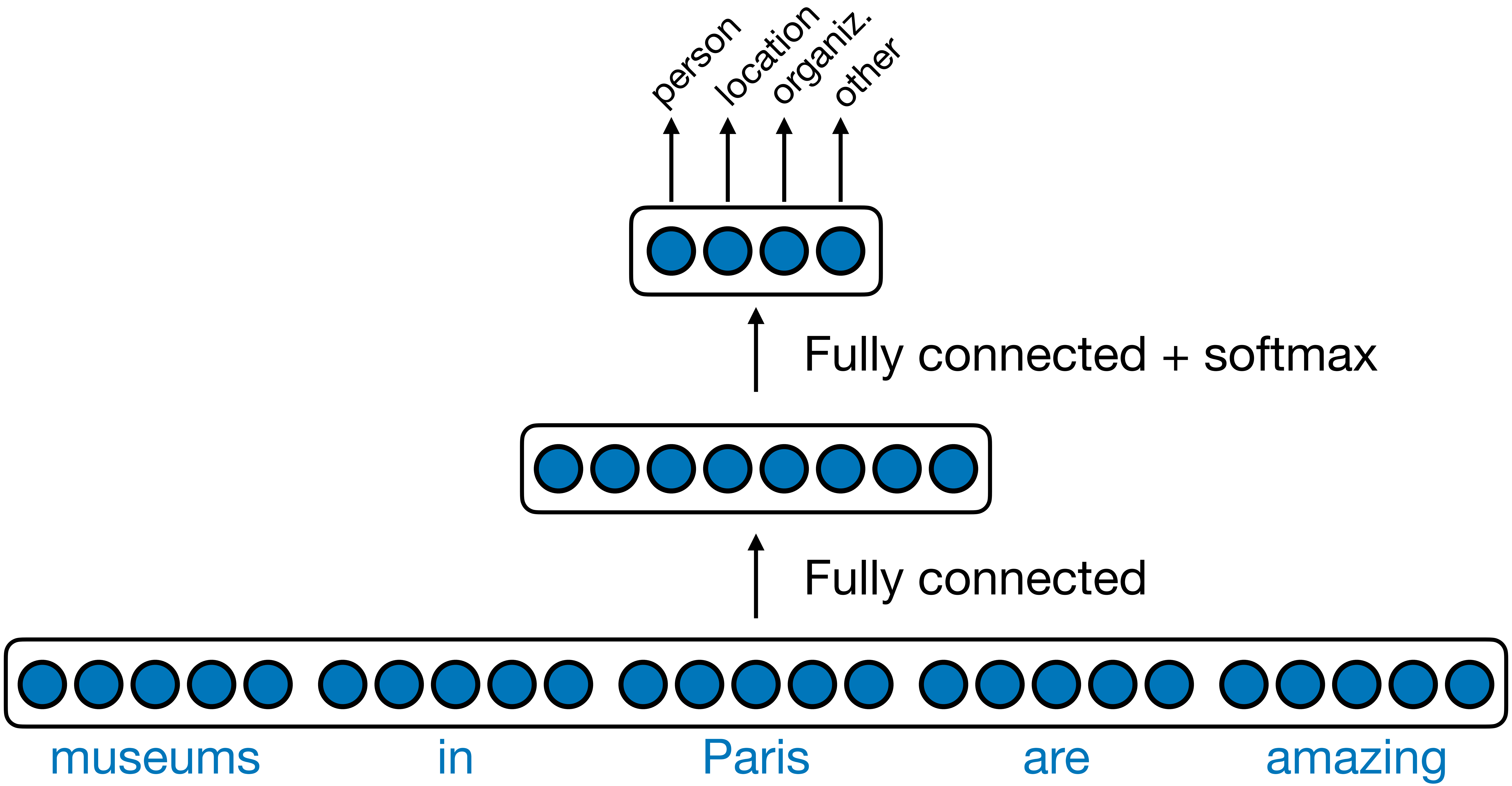
In this context, does Paris refer to Paris Hilton (person) or Paris, France (location)?

This is a kind of word sense disambiguation.

Example from https://www.youtube.com/watch?v=8CWyBNX6eDo&list=PLoROMvodv4rOhcuXMZkNm7j3fVwBBY42z&index=3

Case study 1: Named entity classification

Using word vectors, can create a fully-connected neural network:



Example from https://www.youtube.com/watch?v=8CWyBNX6eDo&list=PLoROMvodv4rOhcuXMZkNm7j3fVwBBY42z&index=3

Fine-tuning word vectors

Fine-tuning word vectors

The word vectors, initialized to their pre-trained values, can either be “frozen” or fine-tuned during training. Which is better?

Fine-tuning word vectors

The word vectors, initialized to their pre-trained values, can either be “frozen” or fine-tuned during training. Which is better?

Advantages of freezing word vectors:

- Faster training and fewer parameters to worry about
- Recommended for small training data sets

Fine-tuning word vectors

The word vectors, initialized to their pre-trained values, can either be “frozen” or fine-tuned during training. Which is better?

Advantages of freezing word vectors:

- Faster training and fewer parameters to worry about
- Recommended for small training data sets

Advantages of fine-tuning word vectors:

- Can give better predictions if there is enough data
- Recommended for large training data sets

From fixed length to variable length

While images are usually a fixed size (e.g. 128x128 pixels), natural language chunks like sentences can have variable length. Using a fixed window size forces you to artificially chop your input. Outputs might also have variable length.

From fixed length to variable length

While images are usually a fixed size (e.g. 128x128 pixels), natural language chunks like sentences can have variable length. Using a fixed window size forces you to artificially chop your input. Outputs might also have variable length.

Input length	Output length	Example
Fixed	Fixed	Image classification
Fixed	Variable	Image captioning
Variable	Fixed	Sentiment analysis
Variable	Variable	Machine translation

From fixed length to variable length

While images are usually a fixed size (e.g. 128x128 pixels), natural language chunks like sentences can have variable length. Using a fixed window size forces you to artificially chop your input. Outputs might also have variable length.

Input length	**Output length**	**Example**
Fixed	Fixed	Image classification
Fixed	Variable	Image captioning
Variable	Fixed	Sentiment analysis
Variable	Variable	Machine translation

We need a neural network architecture to handle variable-length inputs and outputs.

Recurrent neural networks (RNNs)

Recurrent neural networks (RNNs)

How to have variable number of inputs/outputs with a fixed number of parameters?

Recurrent neural networks (RNNs)

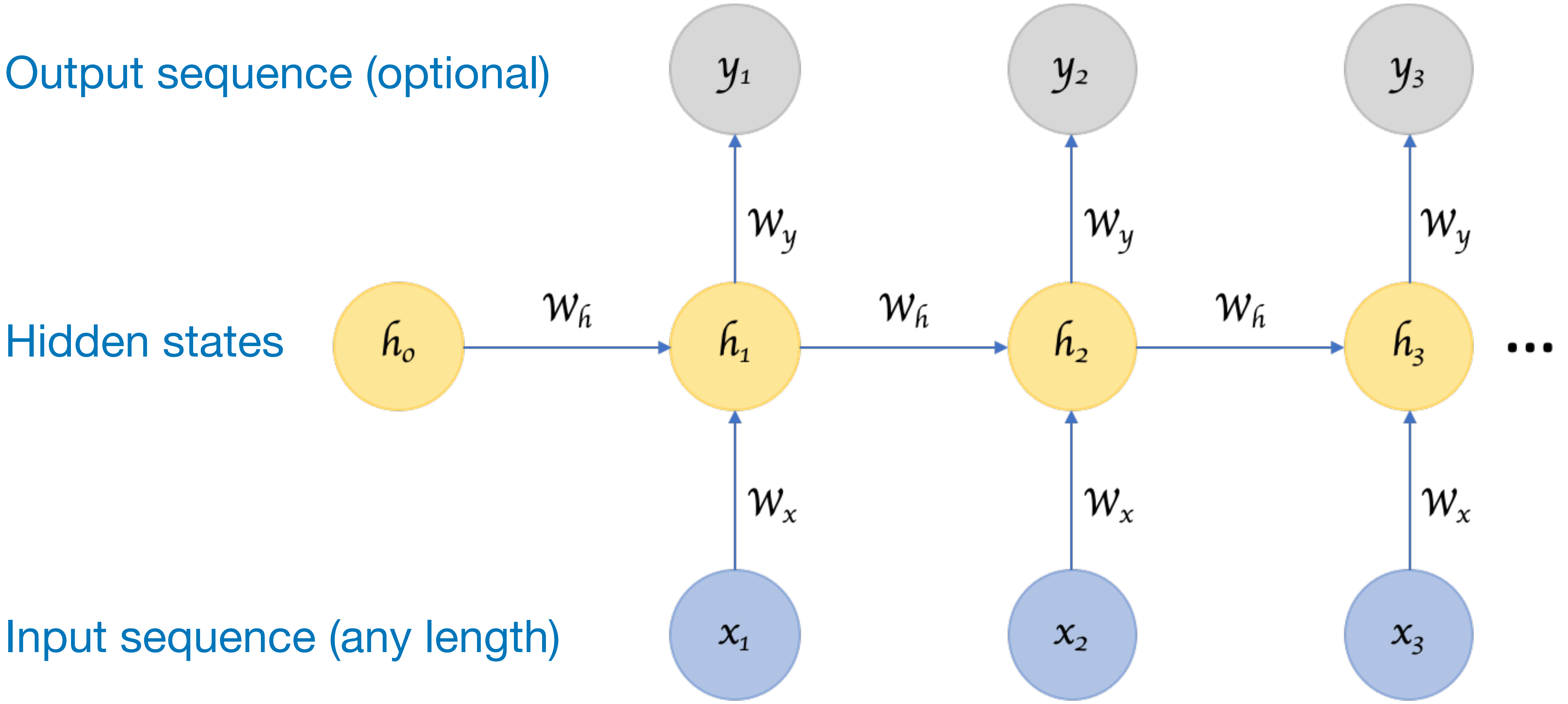
How to have variable number of inputs/outputs with a fixed number of parameters?

Idea: Use a recurrent structure where the same weights are used to update the model at each time step.

Recurrent neural networks (RNNs)

How to have variable number of inputs/outputs with a fixed number of parameters?

Idea: Use a recurrent structure where the same weights are used to update the model at each time step.



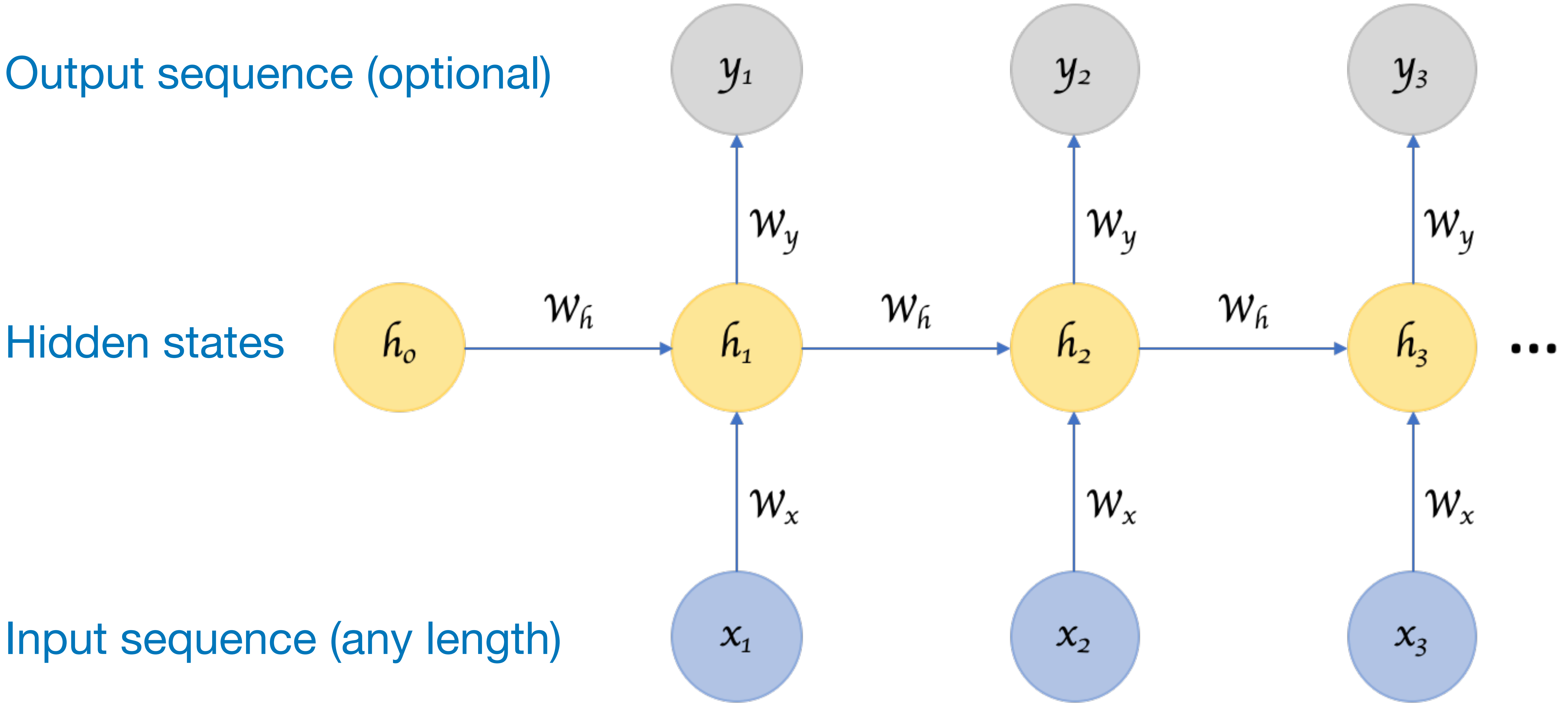
https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce

Recurrent neural networks (RNNs)

How to have variable number of inputs/outputs with a fixed number of parameters?

Idea: Use a recurrent structure where the same weights are used to update the model at each time step.

Hidden state h_t captures available info at time t.



https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce

Recurrent neural networks (RNNs)

How to have variable number of inputs/outputs with a fixed number of parameters?

Idea: Use a recurrent structure where the same weights are used to update the model at each time step.

Hidden state h_t captures available info at time t.

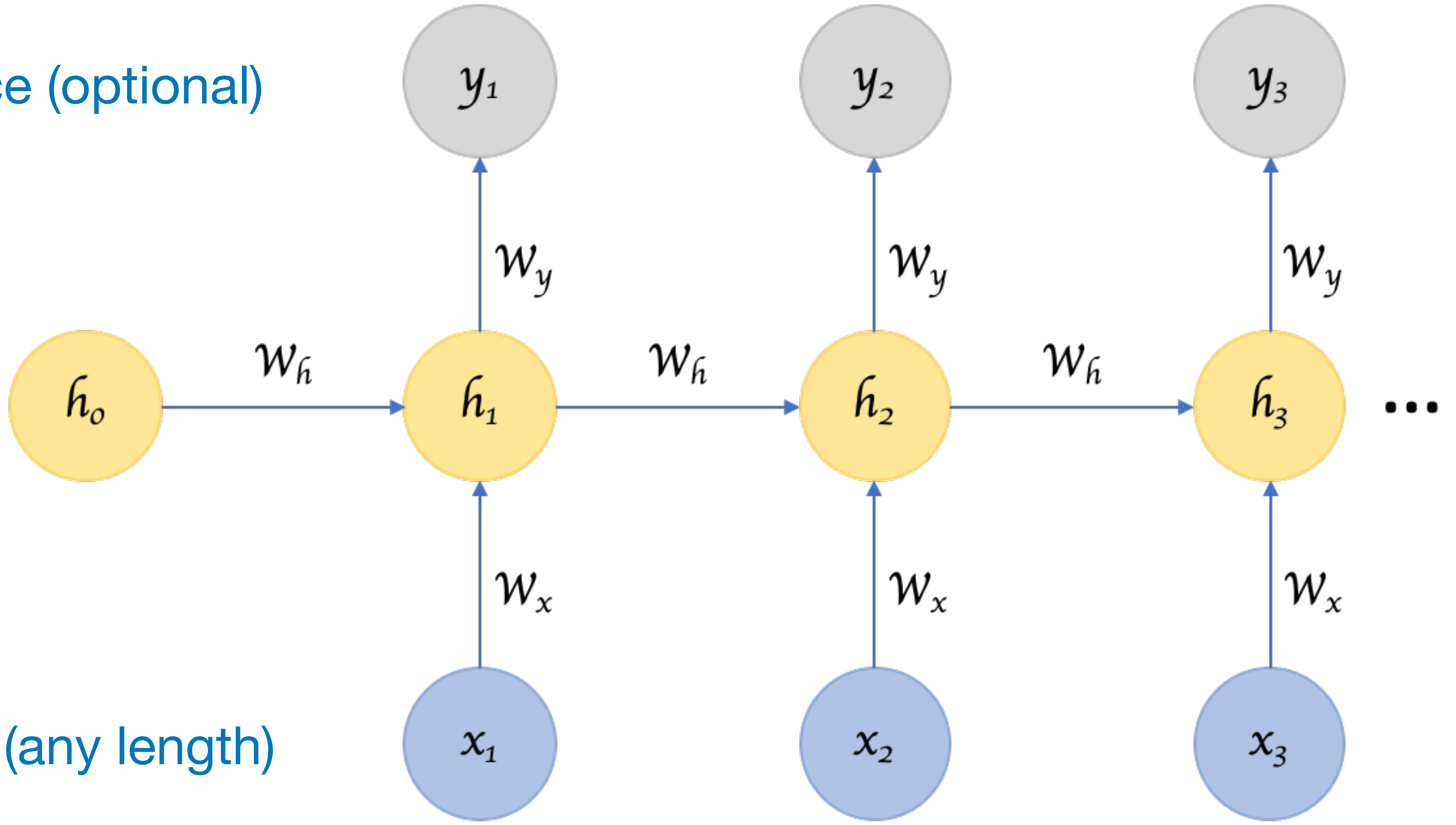
Hidden state evolves based on previous hidden state and new input x_t:

$$h_t = \sigma(W_h h_{t-1} + W_x x_t).$$

Output sequence (optional)

Hidden states

Input sequence (any length)



https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce

Recurrent neural networks (RNNs)

How to have variable number of inputs/outputs with a fixed number of parameters?

Idea: Use a recurrent structure where the same weights are used to update the model at each time step.

Hidden state h_t captures available info at time t.

Hidden state evolves based on previous hidden state and new input x_t:

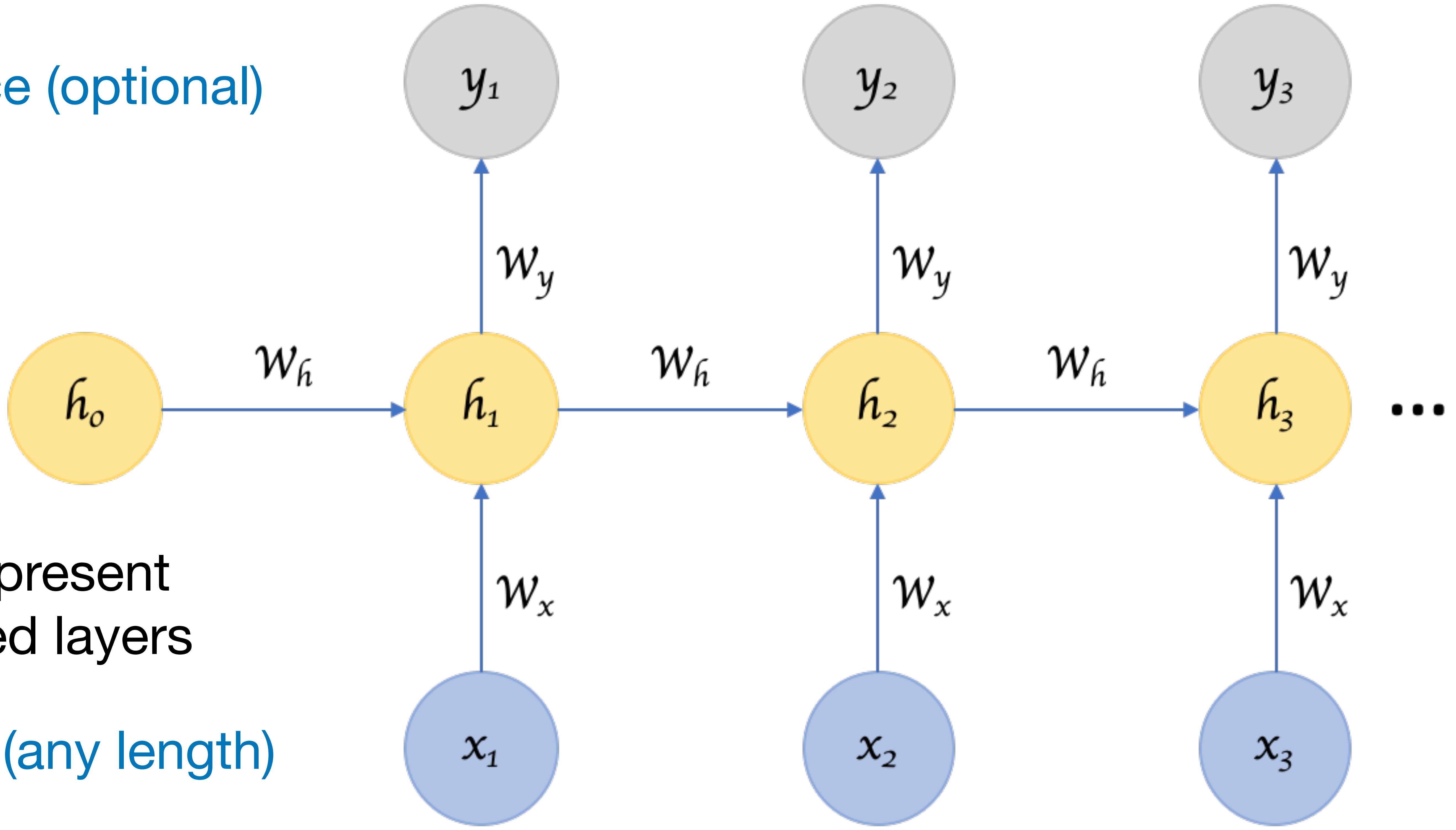
$$h_t = \sigma(W_h h_{t-1} + W_x x_t).$$

Output sequence (optional)

Hidden states

All arrows represent fully connected layers

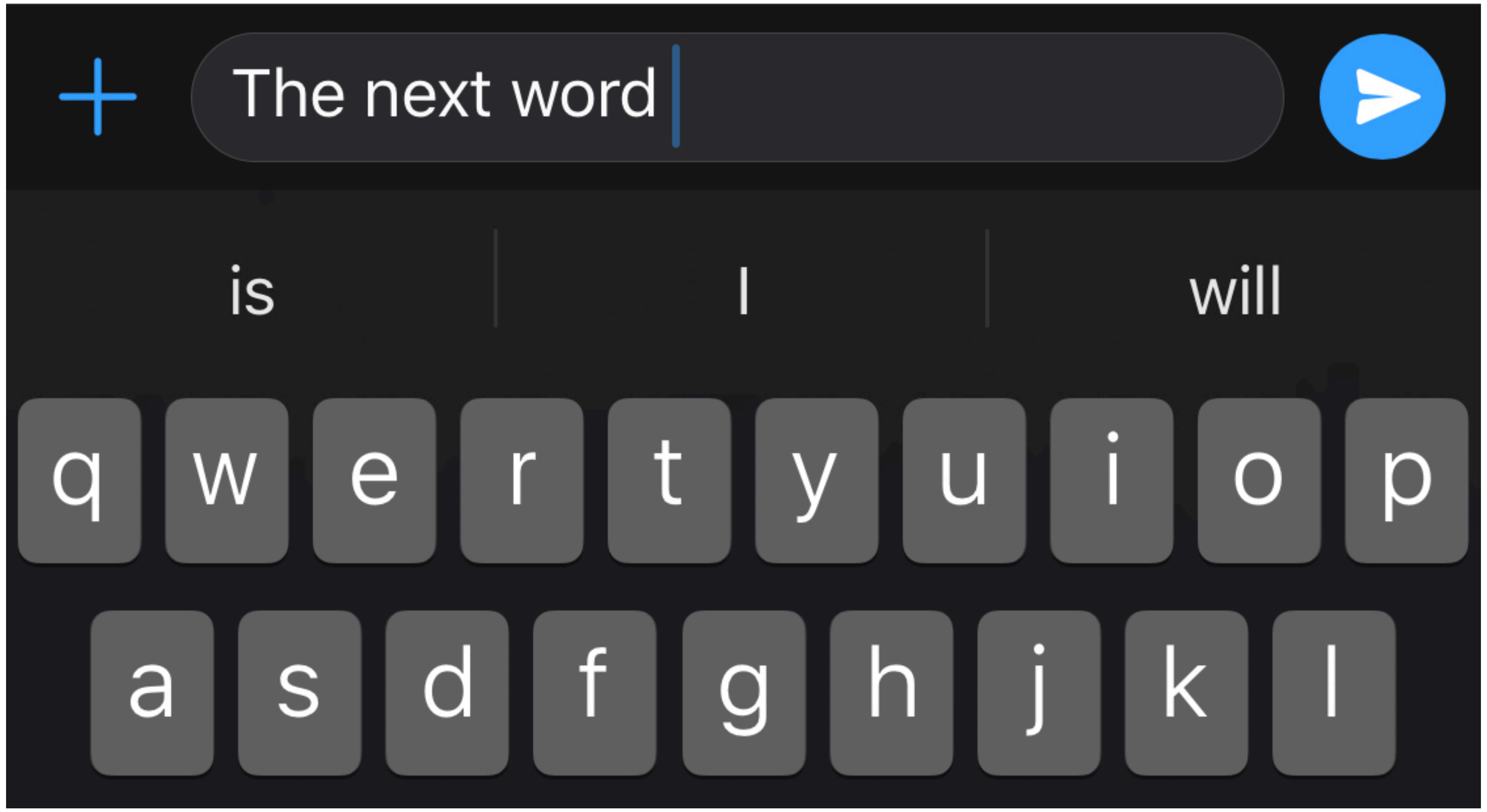
Input sequence (any length)



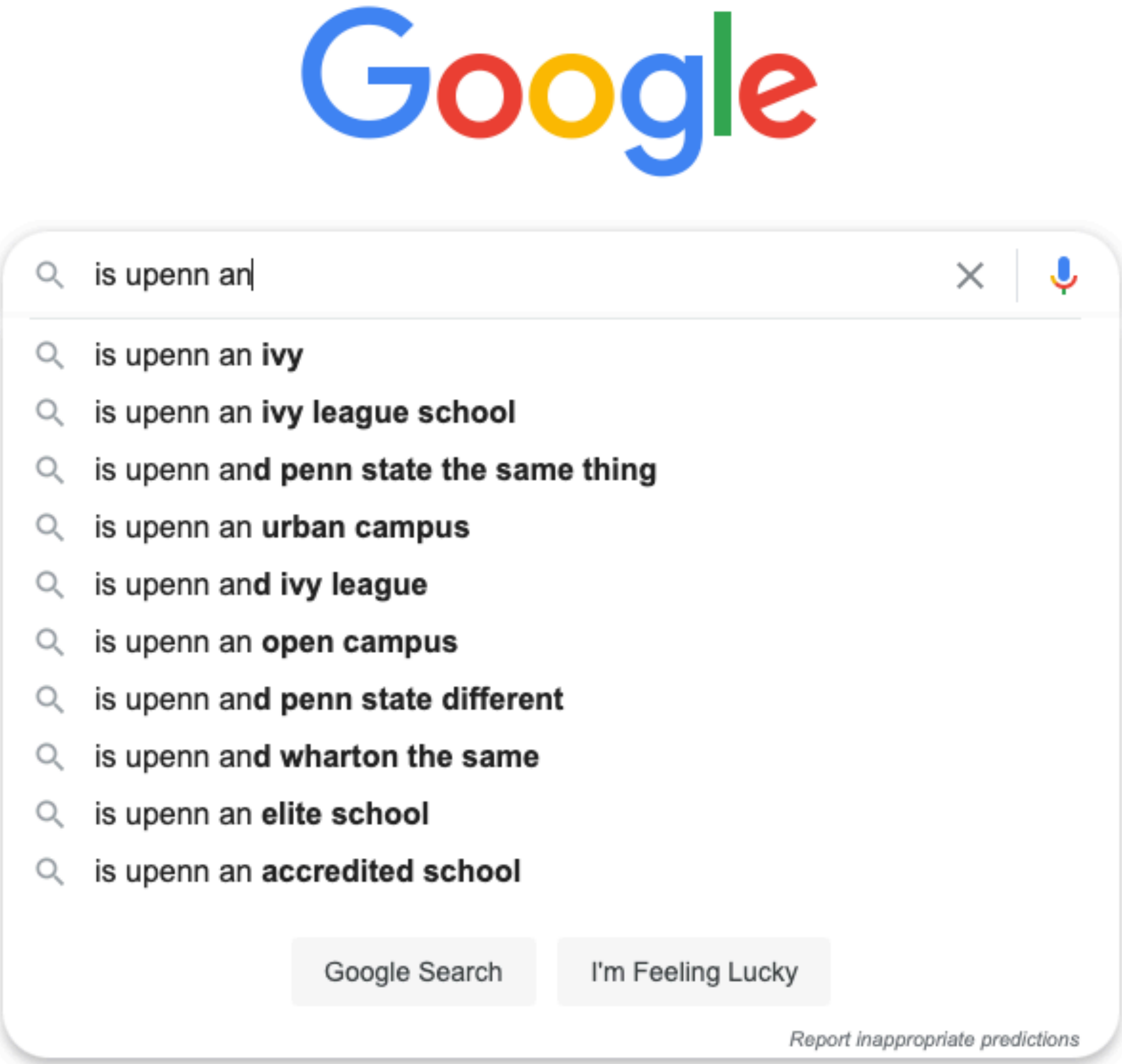
https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce

Case study 2: Language modeling

Given a sequence of words, calculate probabilities of what the next word will be.

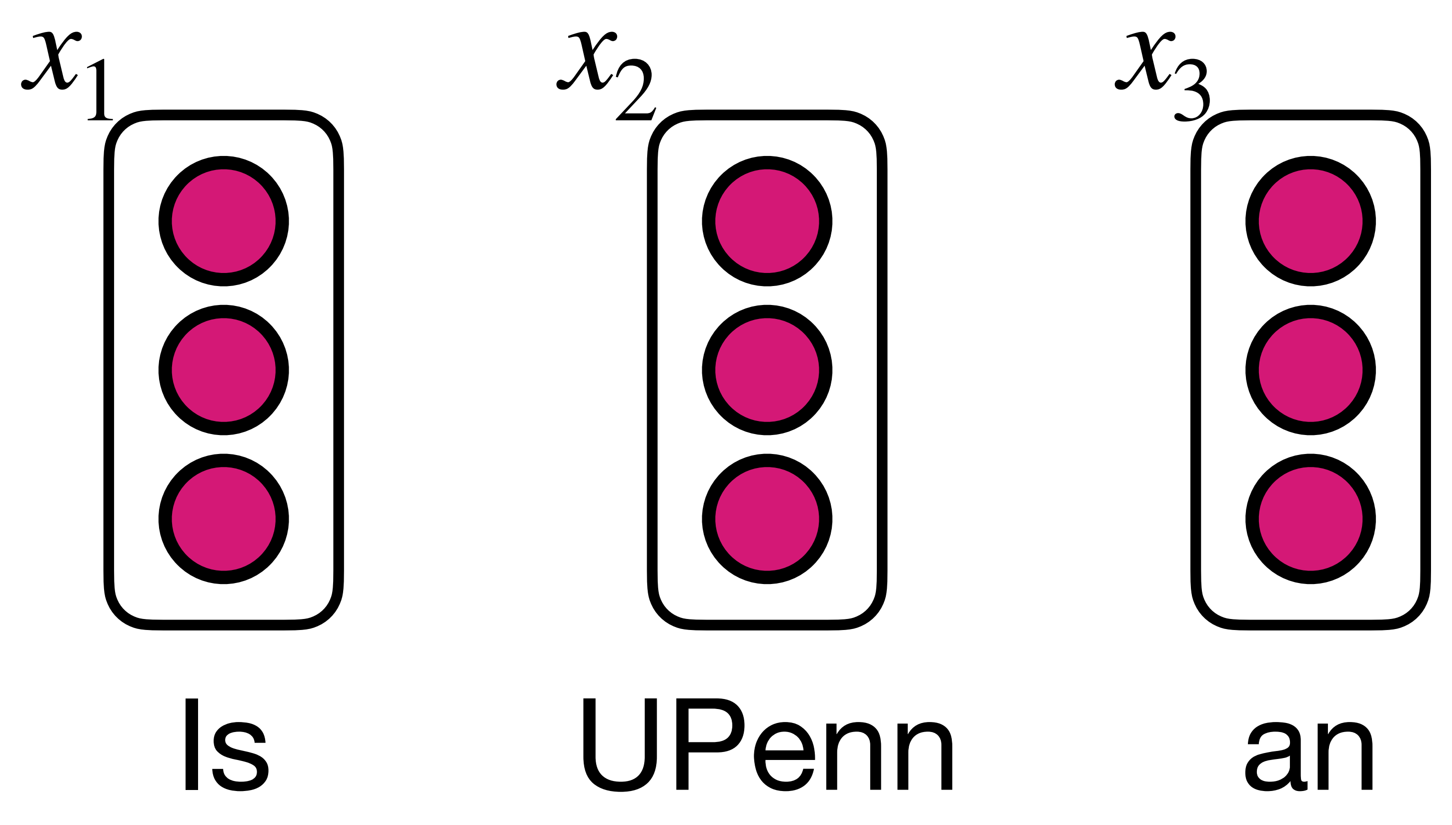


https://towardsdatascience.com/language-modeling-c1cf7b983685



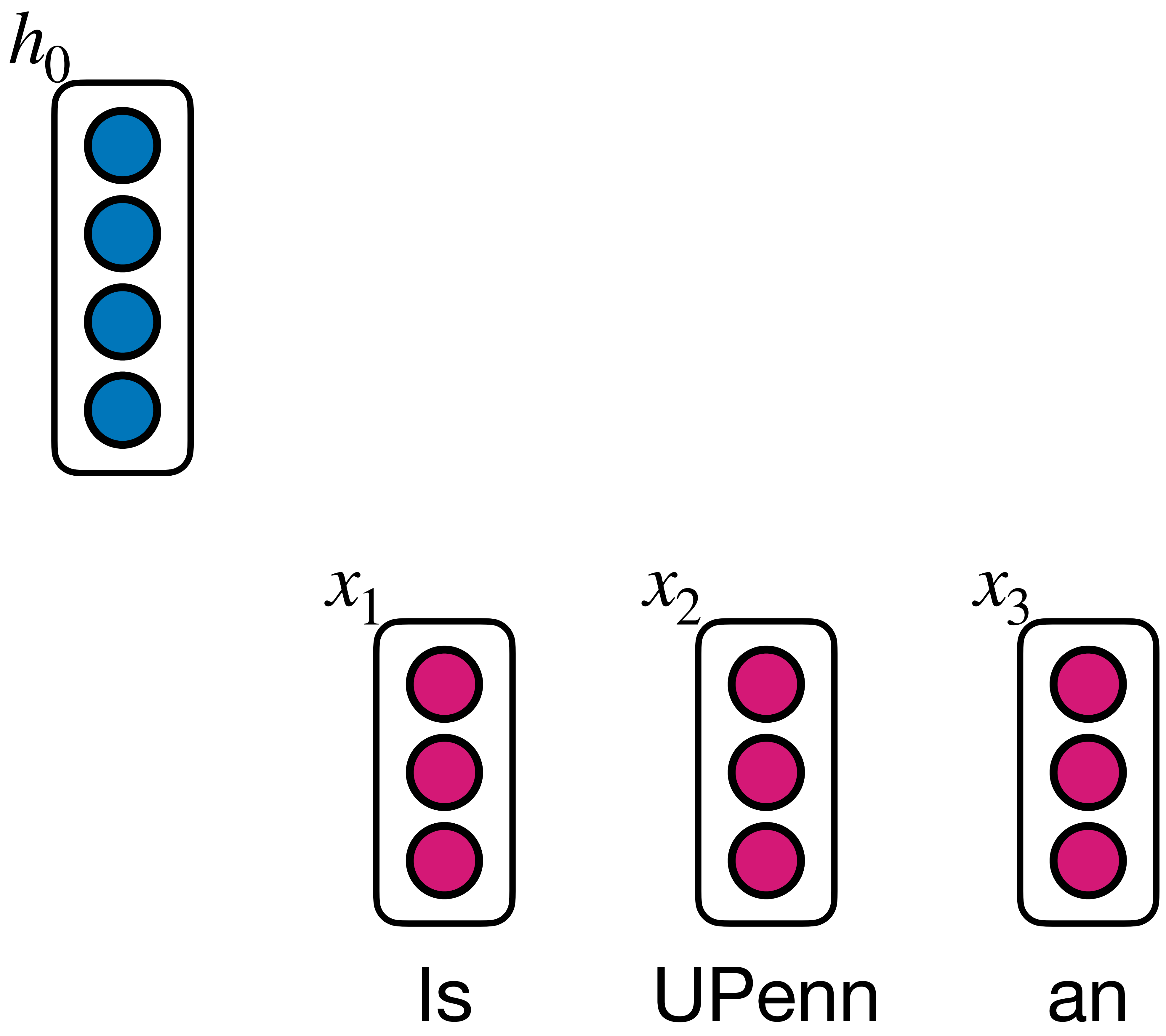
RNN for language modeling

- Initialize h_0 to zero vector.
- Keep updating hidden states using $h_t = \sigma(W_h h_{t-1} + W_x x_t)$.
- Once you get to the end, pass the last hidden state through a fully connected layer plus softmax to get probabilities for next word.



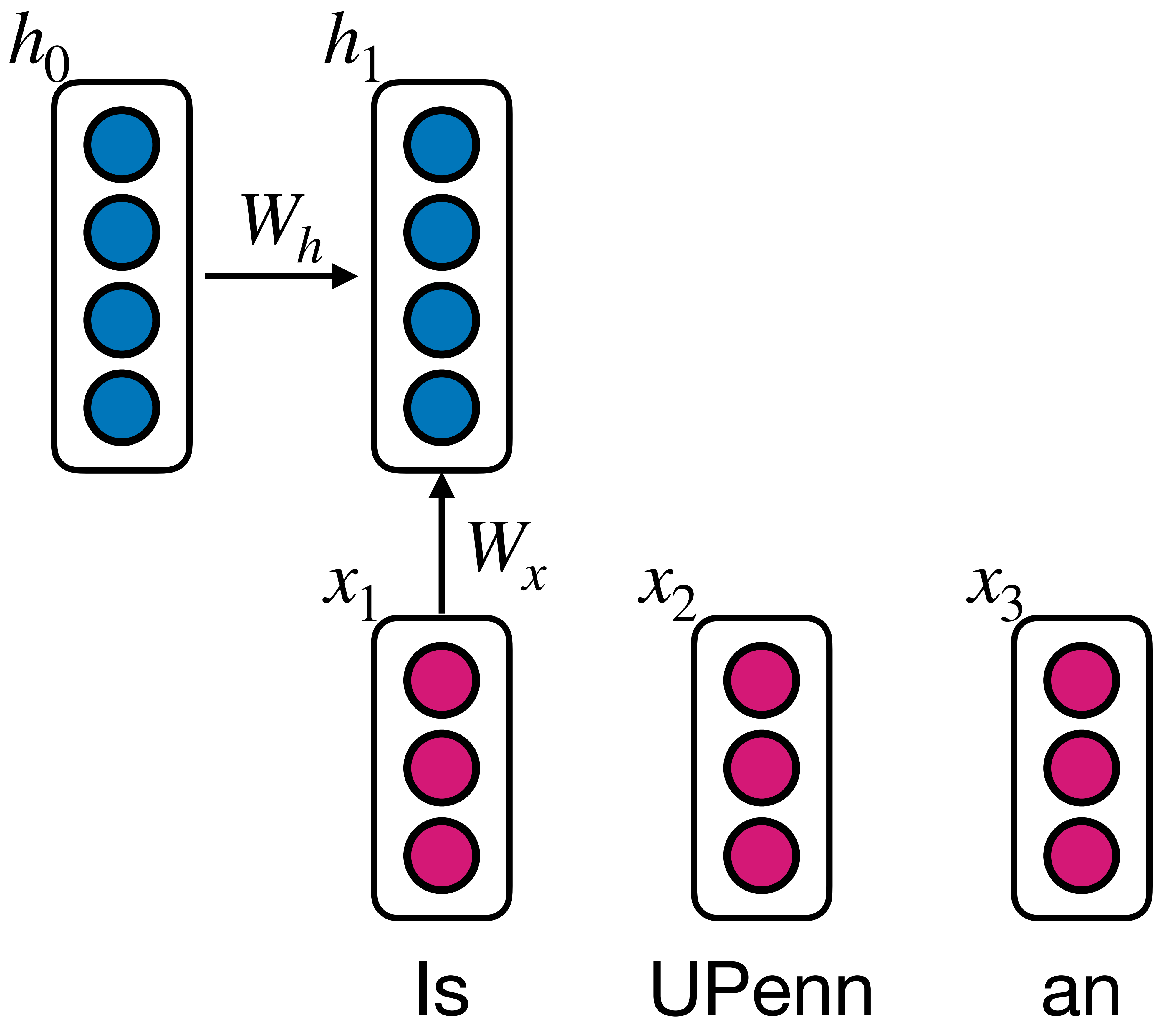
RNN for language modeling

- Initialize h_0 to zero vector.
- Keep updating hidden states using $h_t = \sigma(W_h h_{t-1} + W_x x_t)$.
- Once you get to the end, pass the last hidden state through a fully connected layer plus softmax to get probabilities for next word.



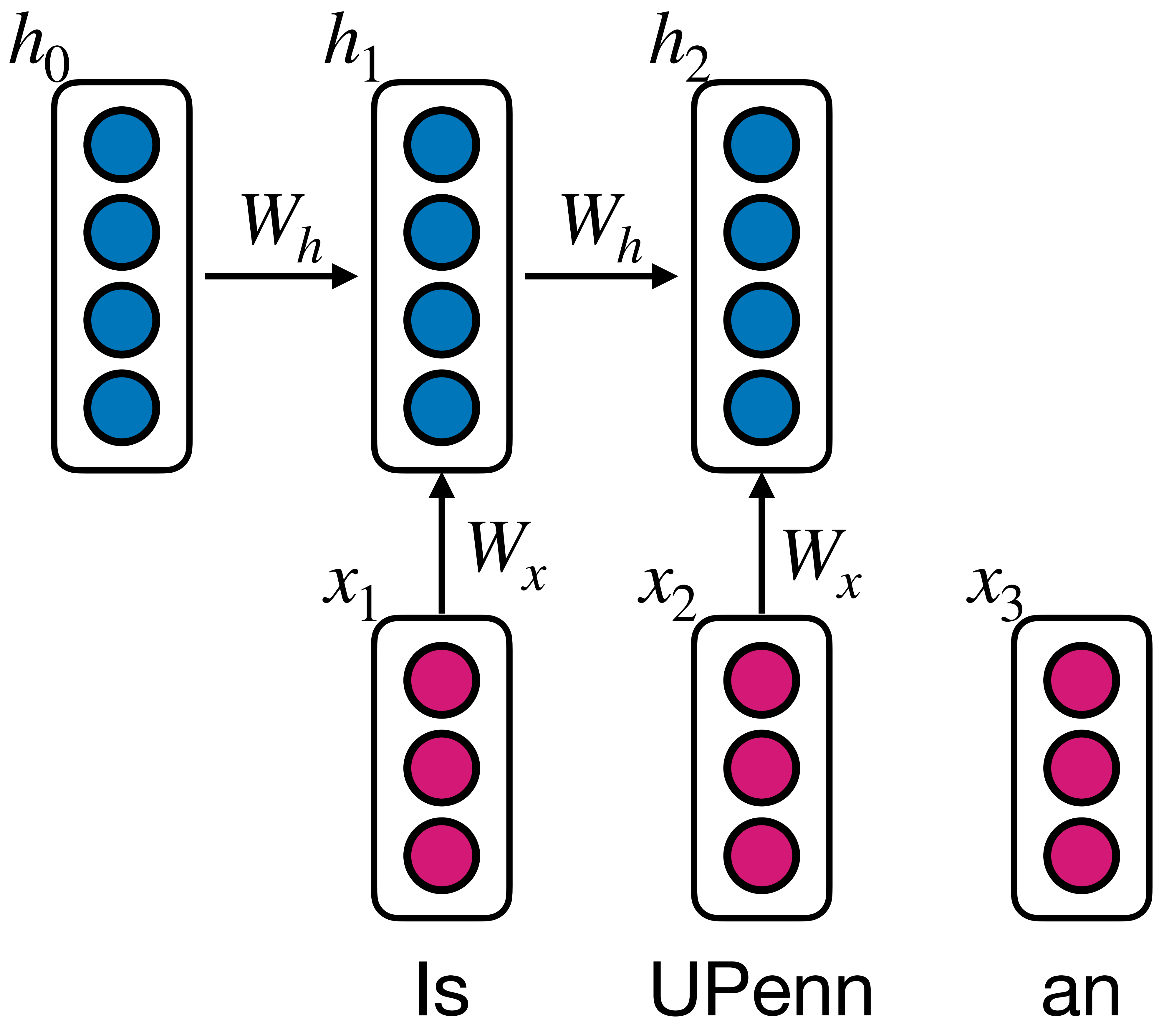
RNN for language modeling

- Initialize h_0 to zero vector.
- Keep updating hidden states using $h_t = \sigma(W_h h_{t-1} + W_x x_t)$.
- Once you get to the end, pass the last hidden state through a fully connected layer plus softmax to get probabilities for next word.



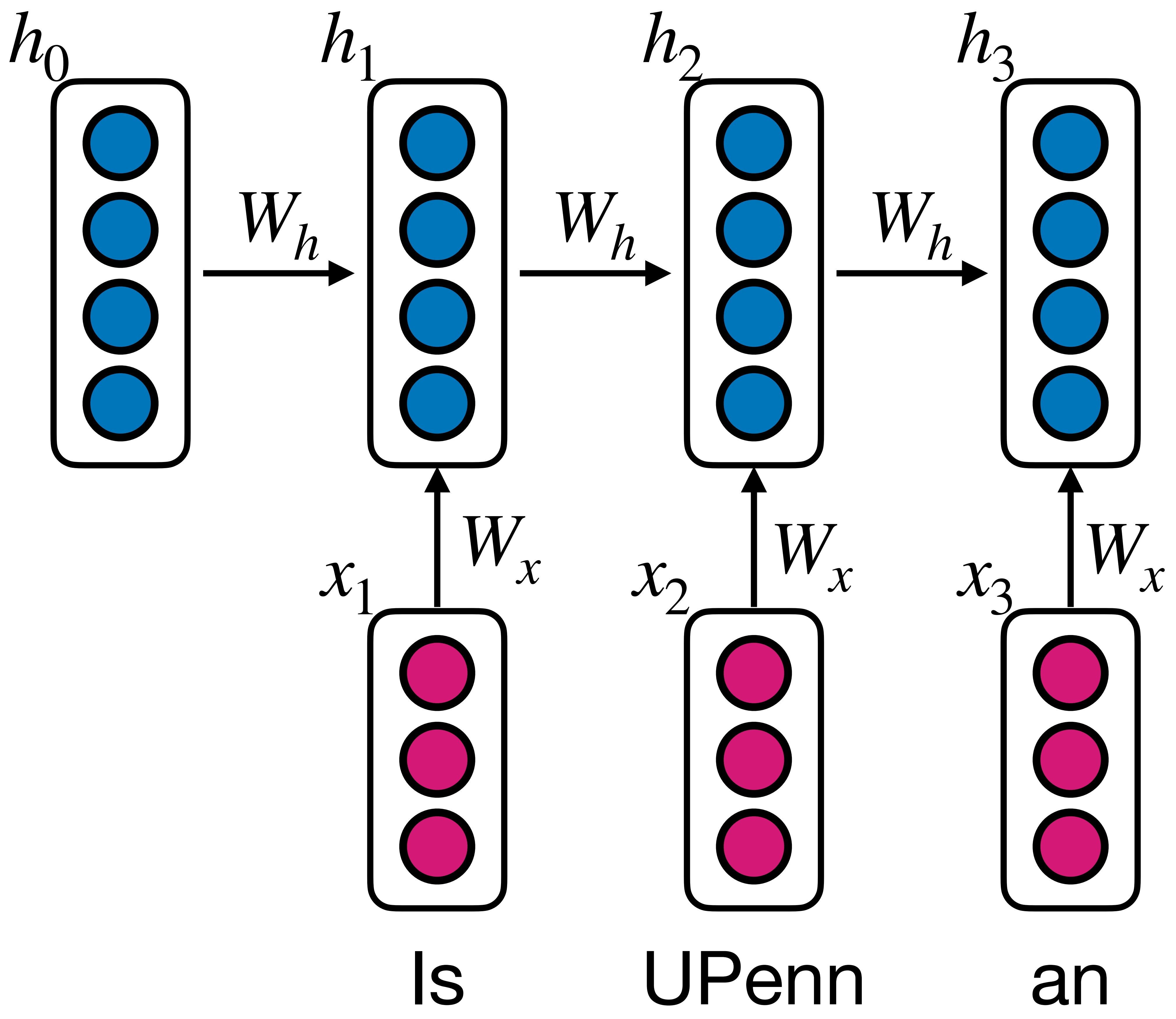
RNN for language modeling

- Initialize h_0 to zero vector.
- Keep updating hidden states using $h_t = \sigma(W_h h_{t-1} + W_x x_t)$.
- Once you get to the end, pass the last hidden state through a fully connected layer plus softmax to get probabilities for next word.



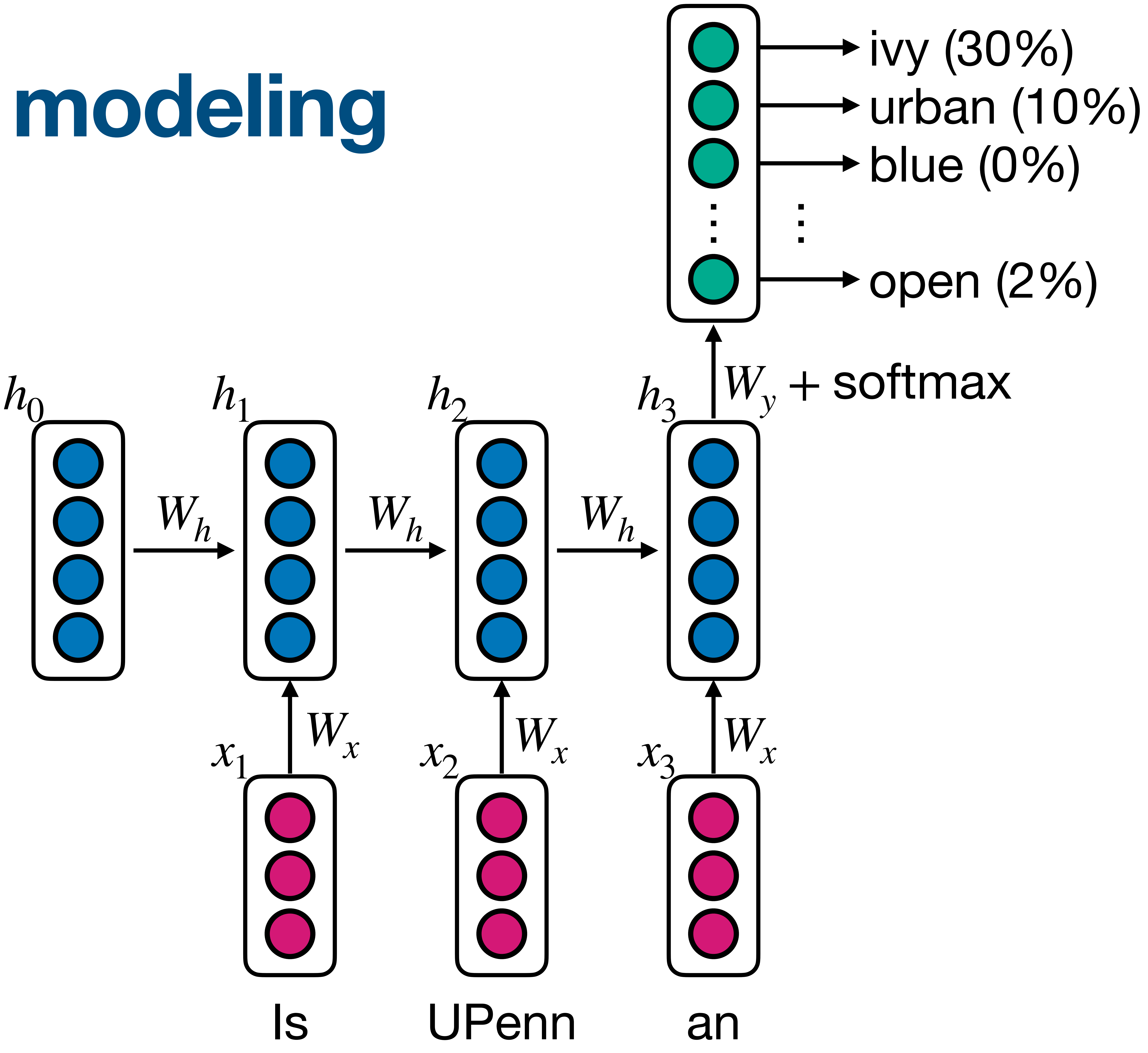
RNN for language modeling

- Initialize h_0 to zero vector.
- Keep updating hidden states using $h_t = \sigma(W_h h_{t-1} + W_x x_t)$.
- Once you get to the end, pass the last hidden state through a fully connected layer plus softmax to get probabilities for next word.



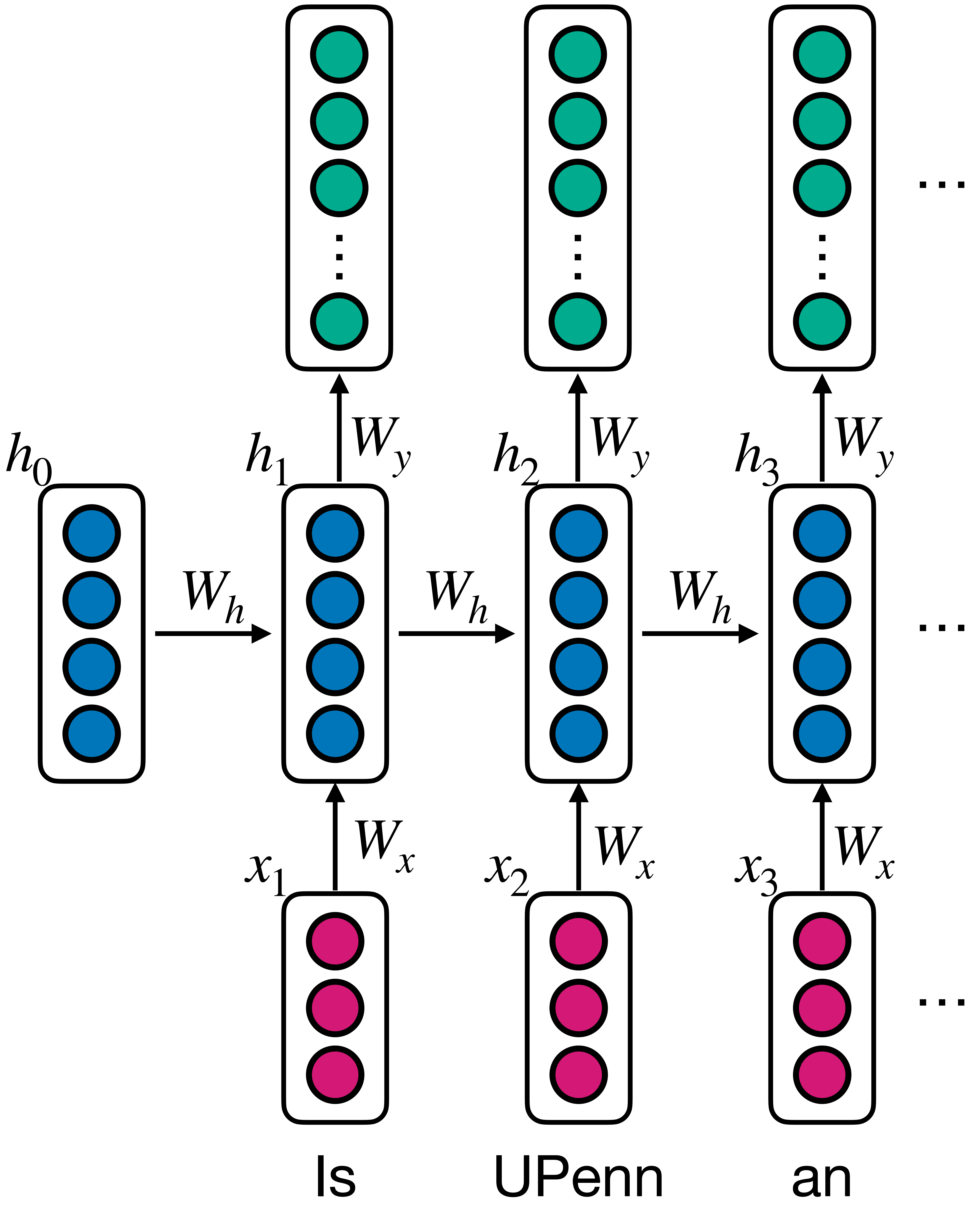
RNN for language modeling

- Initialize h_0 to zero vector.
- Keep updating hidden states using $h_t = \sigma(W_h h_{t-1} + W_x x_t)$.
- Once you get to the end, pass the last hidden state through a fully connected layer plus softmax to get probabilities for next word.



Training an RNN

- Get a big corpus of text, like Wikipedia.
- For any set of parameters, can predict each word based on all previous ones.
- Using the cross-entropy loss function averaged over all words, backpropagate through time to learn W_x, W_h, W_y.



RNNs can also generate text

Example: RNN trained on Barack Obama's speeches

https://medium.com/@samim/obama-rnn-machine-generated-political-speeches-c8abd18a2ea0

RNNs can also generate text

Example: RNN trained on Barack Obama’s speeches

“Good afternoon. God bless you.

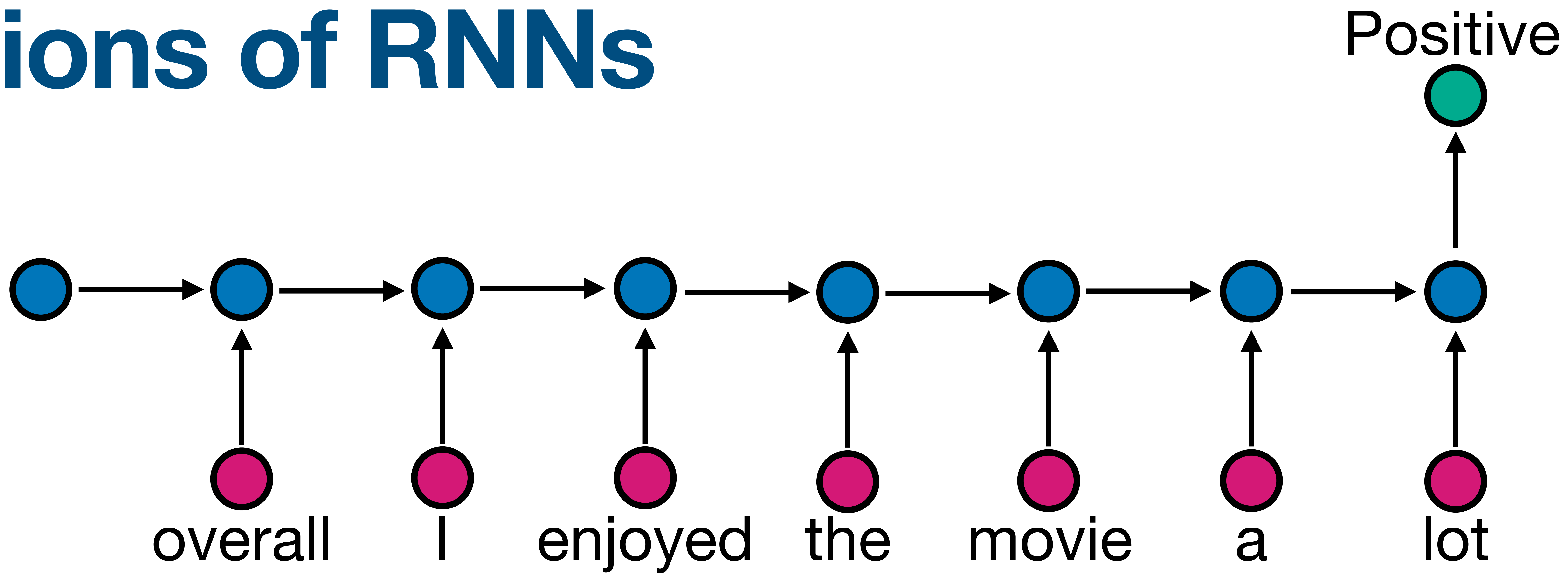
The United States will step up to the cost of a new challenges of the American people that will share the fact that we created the problem. They were attacked and so that they have to say that all the task of the final days of war that I will not be able to get this done. The promise of the men and women who were still going to take out the fact that the American people have fought to make sure that they have to be able to protect our part. It was a chance to stand together to completely look for the commitment to borrow from the American people. And the fact is the men and women in uniform and the millions of our country with the law system that we should be a strong stretch of the forces that we can afford to increase our spirit of the American people and the leadership of our country who are on the Internet of American lives.

Thank you very much. God bless you, and God bless the United States of America.”

https://medium.com/@samim/obama-rnn-machine-generated-political-speeches-c8abd18a2ea0

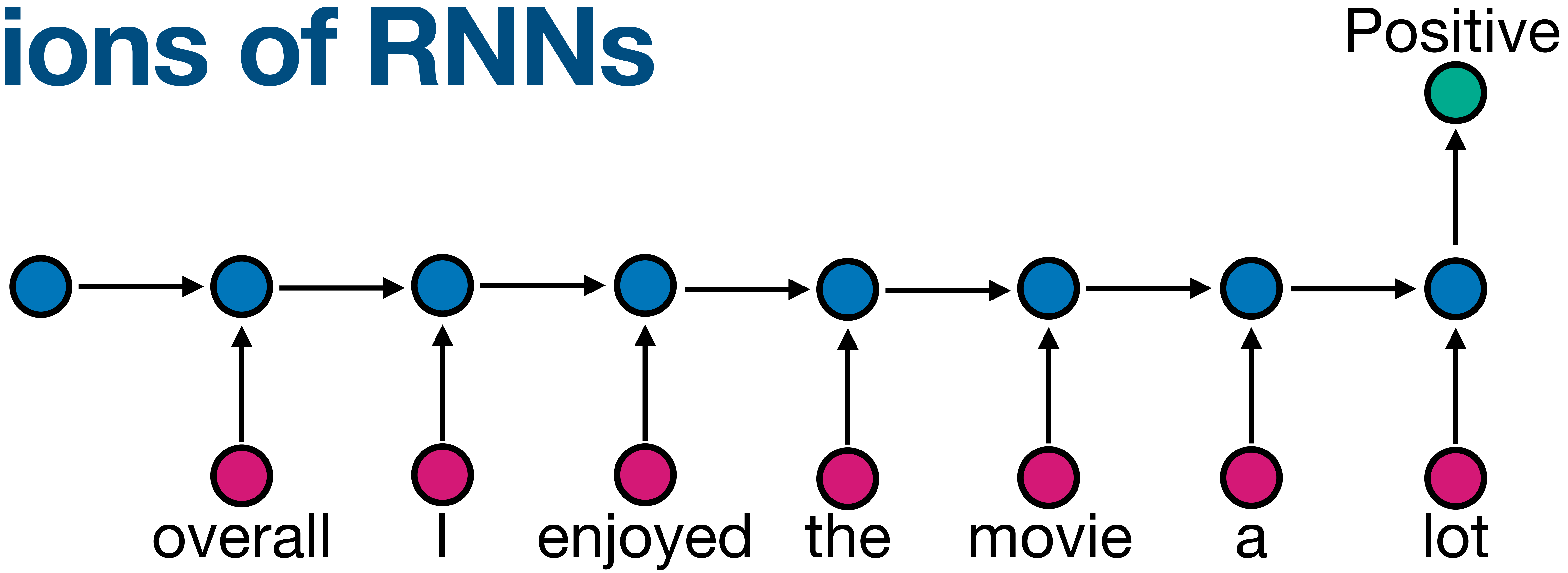
Other applications of RNNs

Sentiment analysis

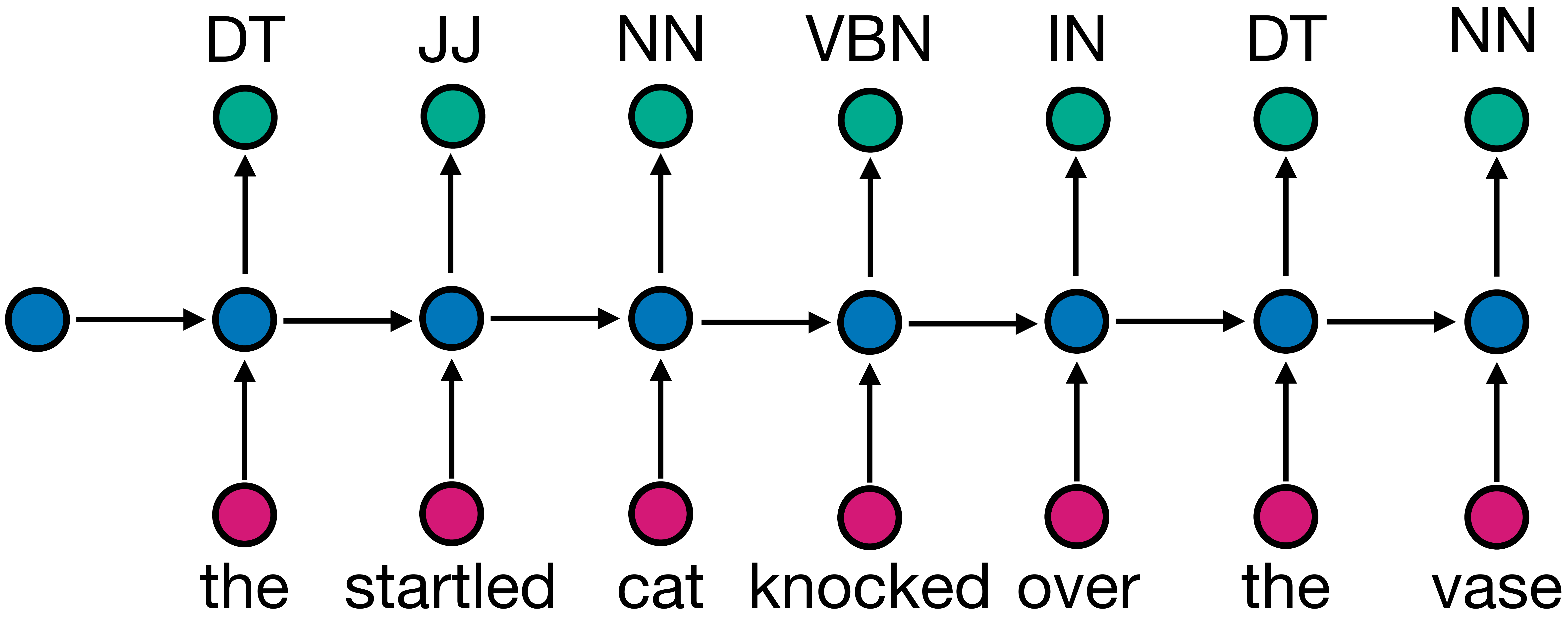


Other applications of RNNs

Sentiment analysis



Part-of-speech tagging



Pre-training and fine-tuning

Pre-training and fine-tuning

Neural networks can learn features
that are useful for multiple tasks.

Pre-training and fine-tuning

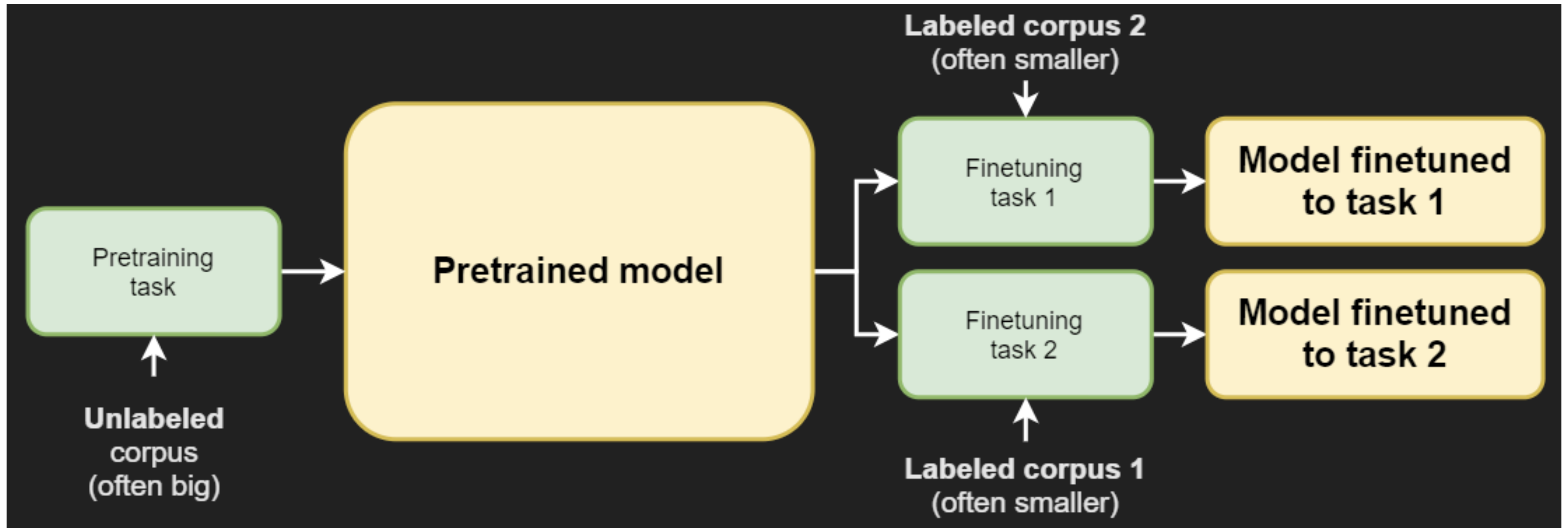
Neural networks can learn features that are useful for multiple tasks.

Paradigm: Unsupervised feature learning (pre-training) followed by fine-tuning to specific tasks.

Pre-training and fine-tuning

Neural networks can learn features that are useful for multiple tasks.

Paradigm: Unsupervised feature learning (pre-training) followed by fine-tuning to specific tasks.



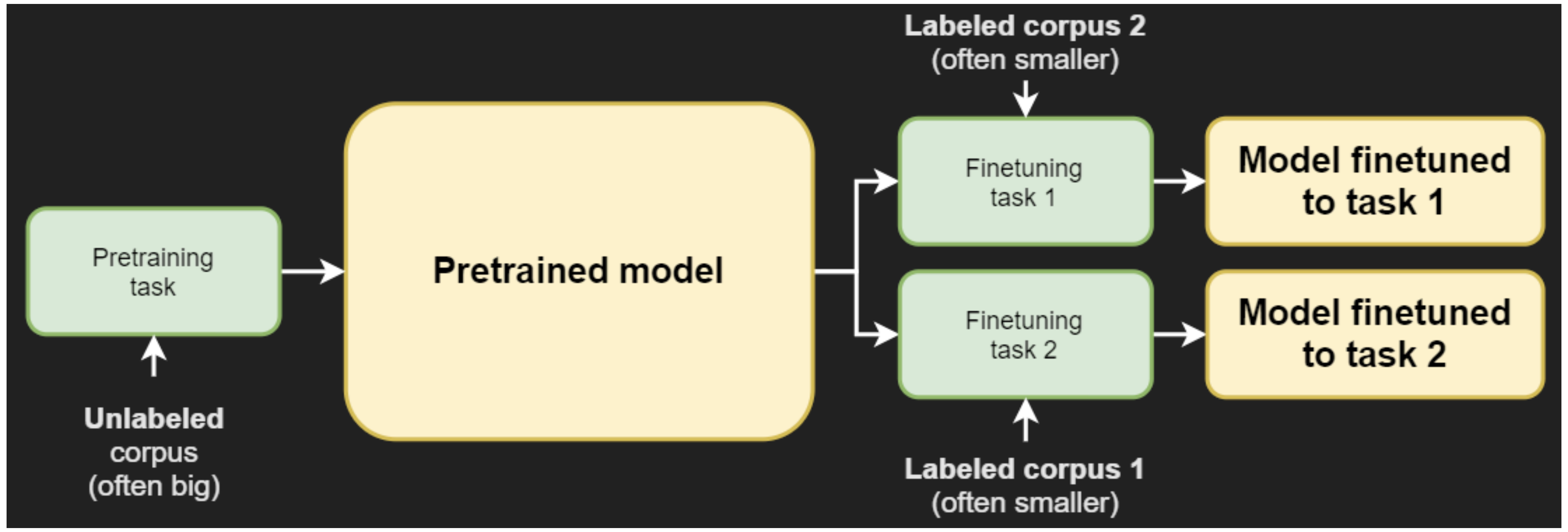
https://www.machinecurve.com/index.php/2021/01/04/intuitive-introduction-to-bert/

Pre-training and fine-tuning

Neural networks can learn features that are useful for multiple tasks.

Paradigm: Unsupervised feature learning (pre-training) followed by fine-tuning to specific tasks.

The final model is obtained by adding a few extra layers on top of pre-trained model and training those.



https://www.machinecurve.com/index.php/2021/01/04/intuitive-introduction-to-bert/

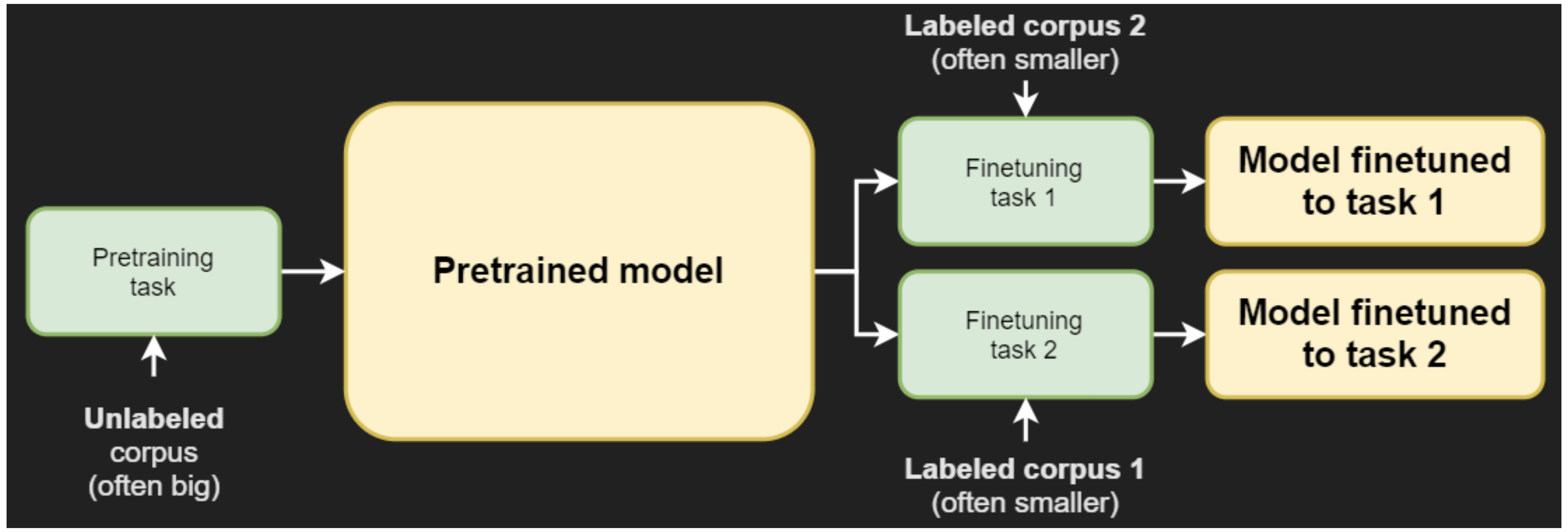
Pre-training and fine-tuning

Neural networks can learn features that are useful for multiple tasks.

Paradigm: Unsupervised feature learning (pre-training) followed by fine-tuning to specific tasks.

The final model is obtained by adding a few extra layers on top of pre-trained model and training those.

BERT (trained on the entirety of Wikipedia) can be easily fine-tuned for many NLP tasks.



https://www.machinecurve.com/index.php/2021/01/04/intuitive-introduction-to-bert/

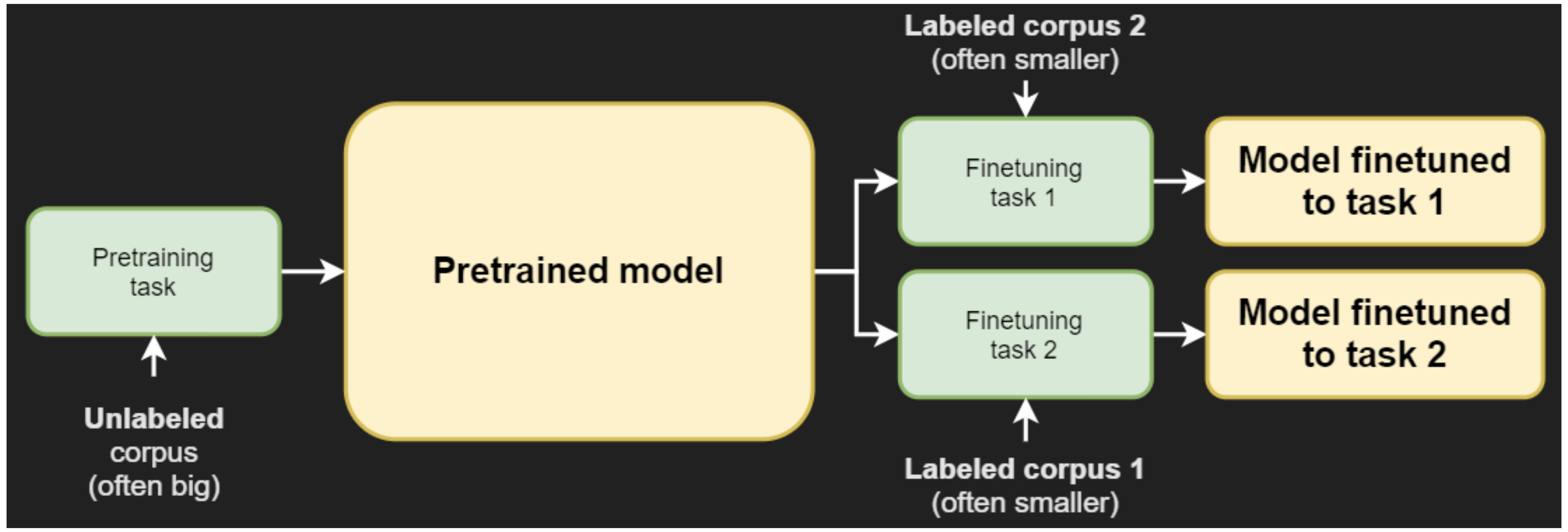
Pre-training and fine-tuning

Neural networks can learn features that are useful for multiple tasks.

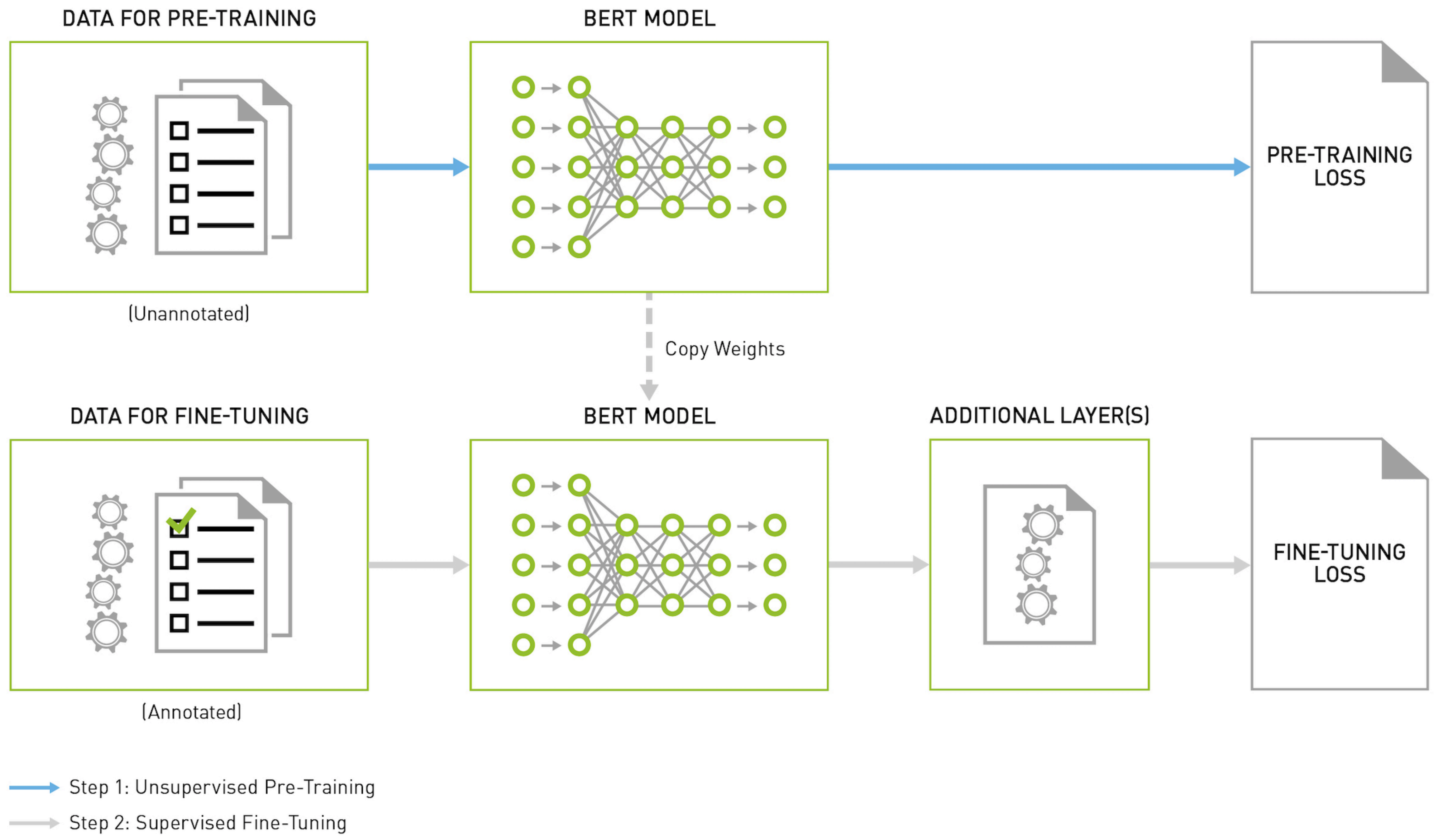
Paradigm: Unsupervised feature learning (pre-training) followed by fine-tuning to specific tasks.

The final model is obtained by adding a few extra layers on top of pre-trained model and training those.

BERT (trained on the entirety of Wikipedia) can be easily fine-tuned for many NLP tasks.



https://www.machinecurve.com/index.php/2021/01/04/intuitive-introduction-to-bert/



https://medium.com/future-vision/bert-meets-gpus-403d3fbed848

Summary

- Word vectors used to translate words into numbers for predictive modeling.
- Usual fully connected architecture can be paired with word vectors for window-based tasks like named entity classification.
- New architectures, such as recurrent neural networks, needed for variable-length inputs and outputs.
- RNNs work by processing the input sequence one word at a time, updating a hidden representation of the input using a fixed set of weights.
- RNNs can be applied to a diverse range of predictive tasks, like language modeling or sentiment analysis.
- The hard work of training deep learning models can be recycled through the pre-training and fine-tuning paradigm (applies not just to NLP).

Quiz Practice