Neural Network Language Model

Lecture # 4

Neural Network Language Model

Language Model

You shall know a word by the company it keeps

—Firth, J. R. 1957:11

Language Model

Fill in the blank:

... a ________ ...

car	cars	water	cat

Language Model

Fill in the blank:

... a __________ ...

car and *cat* both work

car	cars	water	cat

Language Model

Fill in the blank:

... a ________ ...

car and *cat* both work

car | cars | water | cat

John is driving a ________ ...

Language Model

Fill in the blank:

... a ________ ...

car and *cat* both work

car	cars	water	cat

John is driving a ________ ...

only *car* works here

Language Model

Fill in the blank:

... a ________ ...

car and *cat* both work

car	cars	water	cat

John is driving a ________ ...

Similarly, machines use the context to predict the next words

Language Model

You chose “driving a car” because you’ve seen that phrase more frequently

“driving a cat” is not a common phrase

Language Model

Fill in the blank:

This _______ is going at 100 km/hours

car

bicycle

Car at 100km/hours is more probable than a bicycle

Language Model

Language model defines
"how probable a sentence is"

Language Model

Let’s look at the example again

How probable is:

John is driving a car vs. John is driving a cat

In other words, what is the probability to predict cat or car given the context “John is driving a”

Language Model

Q: Can we see language modeling as a classification problem?

Language Model

Q: Can we see language modeling as a classification problem?

A: Yes! We are just predicting which word ("class") is coming next.

Language Model

Q: Can we see language modeling as a classification problem?

A: Yes! We are just predicting which word ("class") is coming next.

$$p(\texttt{<s>}\ \text{Dan likes ham}\ \texttt{</s>}) =$$

Language Model

Q: Can we see language modeling as a classification problem?

A: Yes! We are just predicting which word ("class") is coming next.

$$p(\texttt{<s>}\ \text{Dan likes ham}\ \texttt{</s>}) = p(\texttt{Dan}\,|\,\texttt{<s>})$$

Predict **Dan** given **<s>**

Language Model

Q: Can we see language modeling as a classification problem?

A: Yes! We are just predicting which word ("class") is coming next.

$$p(\texttt{<s>}\ \text{Dan likes ham}\ \texttt{</s>}) = p(\texttt{Dan} \mid \texttt{<s>})$$

Predict **Dan** given **<s>**

What is the probability of **Dan** given **<s>**?

Language Model

Q: Can we see language modeling as a classification problem?

A: Yes! We are just predicting which word (“class”) is coming next.

$$p(\texttt{<s>}\ \text{Dan likes ham}\ \texttt{</s>}) = p(\texttt{Dan|<s>}) \cdot p(\texttt{likes|Dan})$$

Predict **likes** given **Dan**

Language Model

Q: Can we see language modeling as a classification problem?

A: Yes! We are just predicting which word ("class") is coming next.

$$p(\texttt{<s>}\ \text{Dan likes ham}\ \texttt{</s>}) = p(\texttt{Dan}\,|\,\texttt{<s>})$$
$$\cdot\, p(\texttt{likes}\,|\,\texttt{Dan})$$
$$\cdot\, p(\texttt{ham}\,|\,\texttt{likes})$$

Predict **ham** given **likes**

Language Model

Q: Can we see language modeling as a classification problem?

A: Yes! We are just predicting which word ("class") is coming next.

$$
\begin{aligned}
p(\texttt{<s>}\ \text{Dan likes ham}\ \texttt{</s>}) = \; & p(\texttt{Dan}\,|\,\texttt{<s>}) \\
& \cdot p(\texttt{likes}\,|\,\texttt{Dan}) \\
& \cdot p(\texttt{ham}\,|\,\texttt{likes}) \\
& \cdot \boxed{p(\texttt{</s>}\,|\,\texttt{ham})}
\end{aligned}
$$

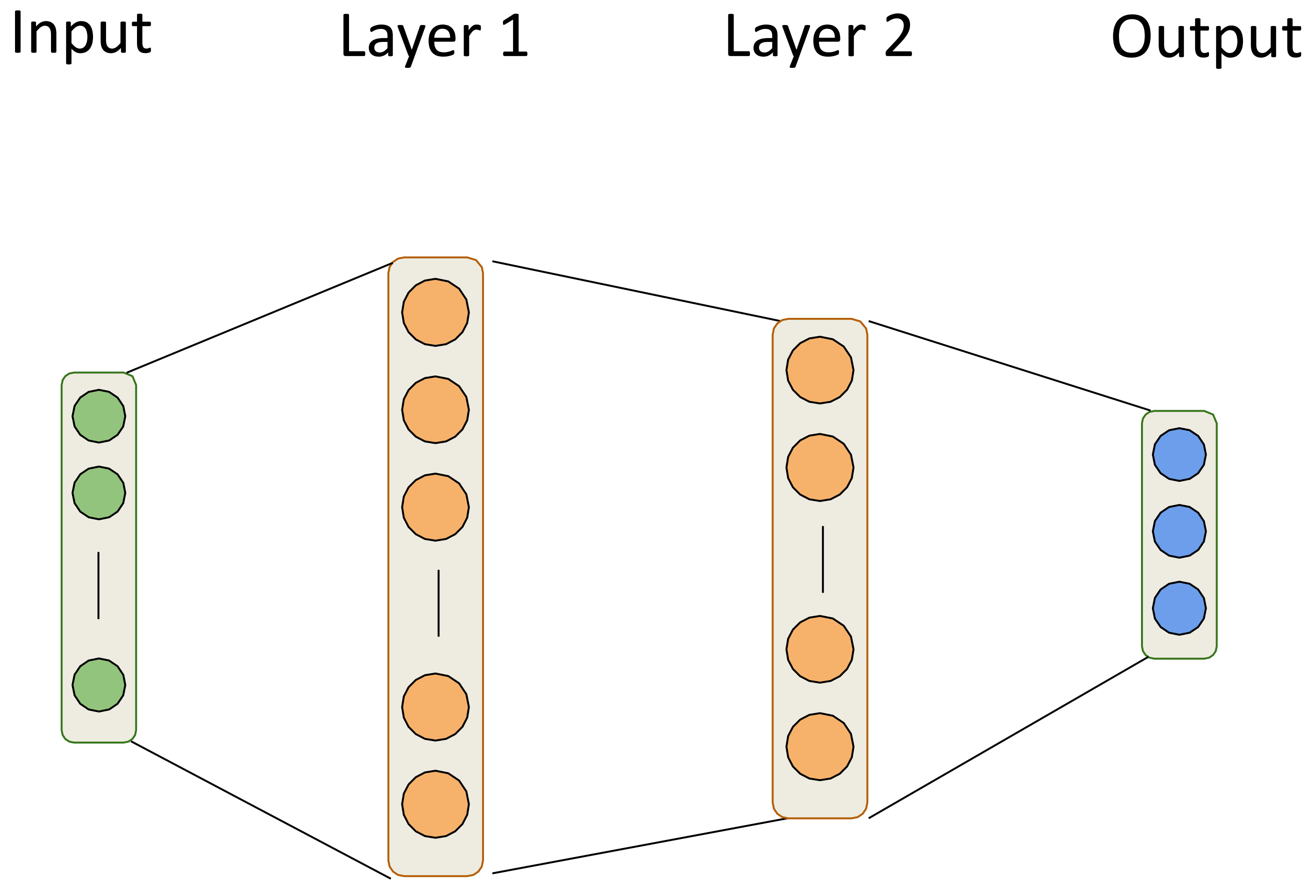
Language Model

Words represent classes that we want to predict!

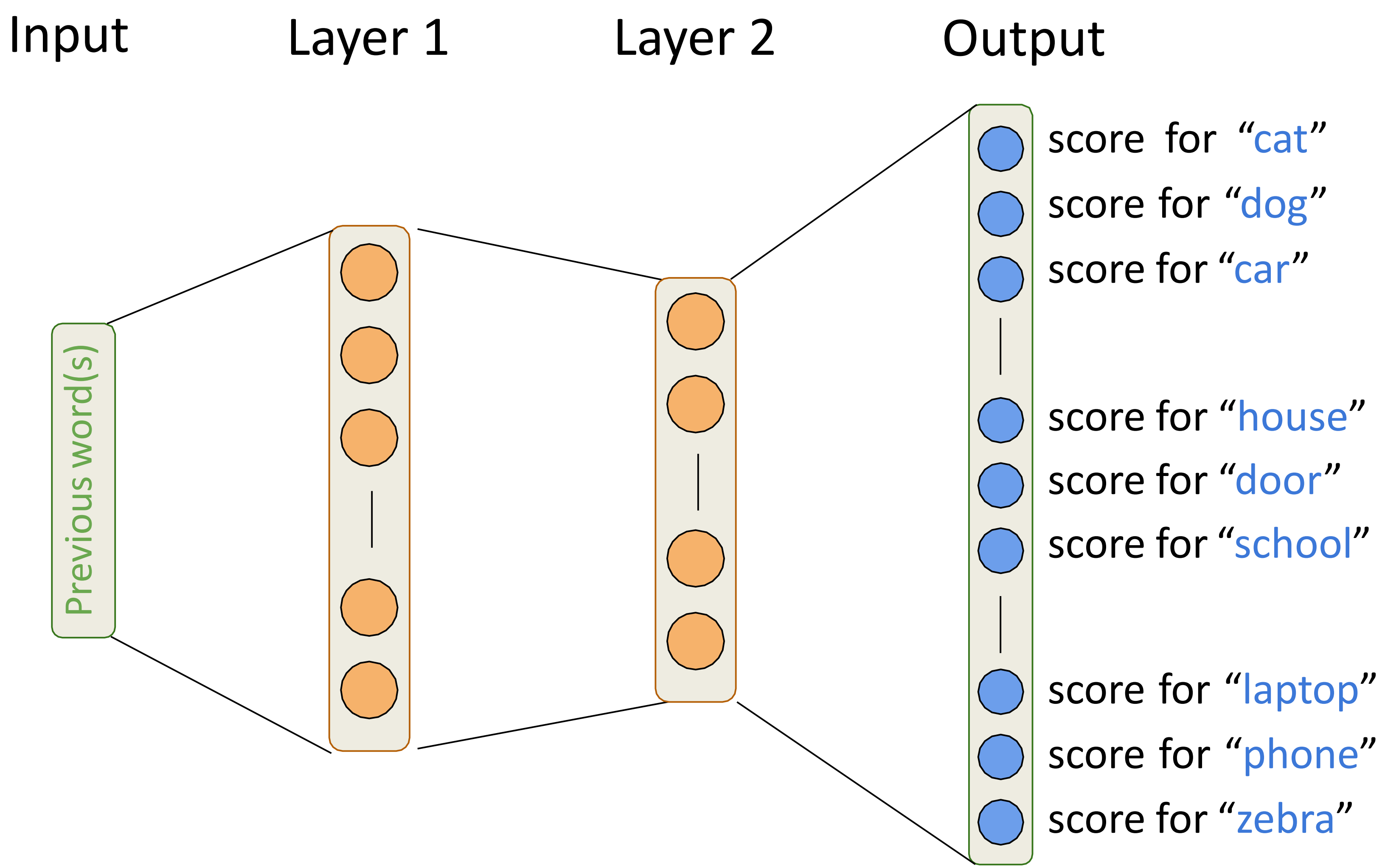
Input to the classifier: previous words i.e. context

Output: probability distribution over all possible words, i.e. our vocabulary

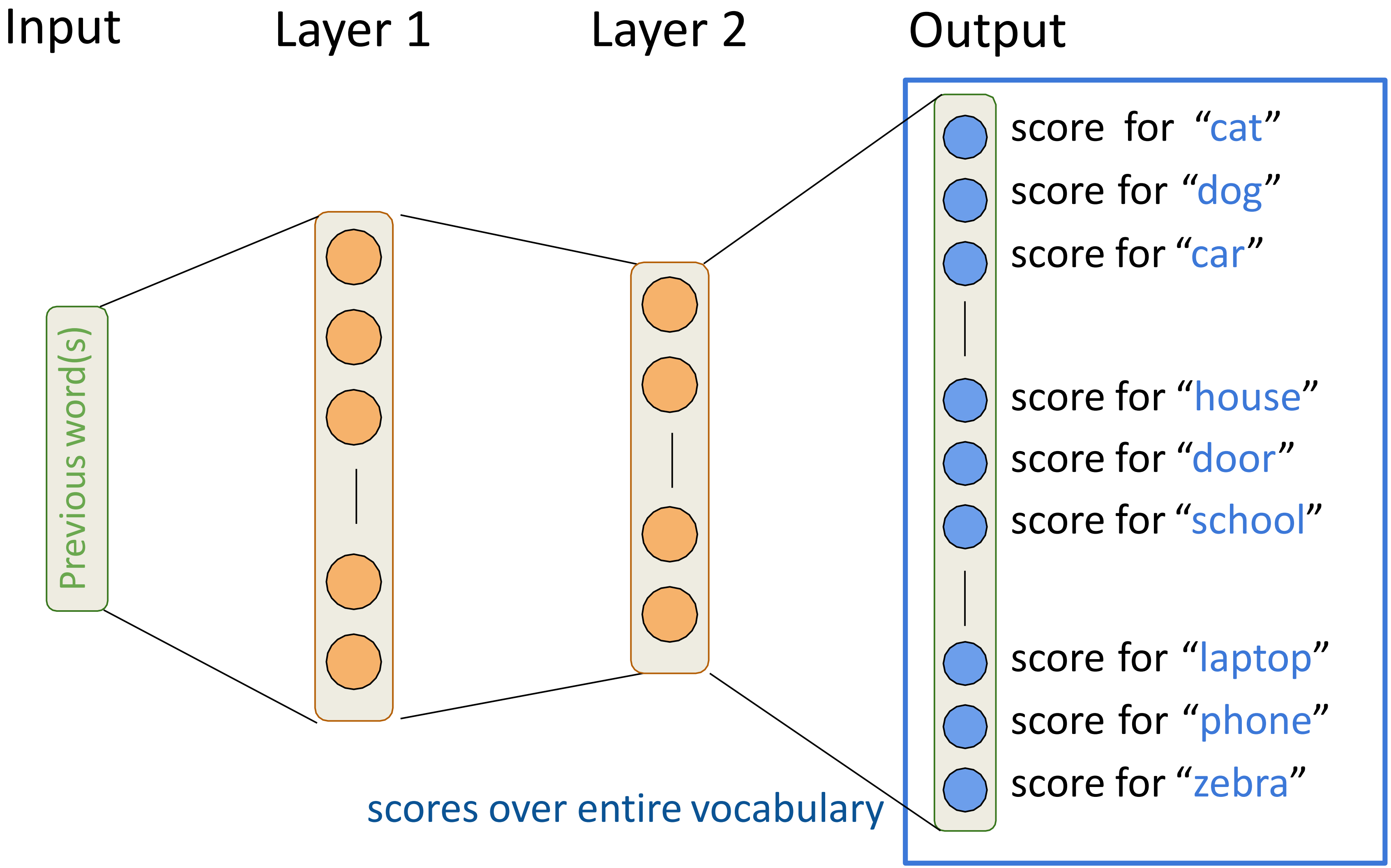
Neural Network Language Model



Neural Network Language Model



Neural Network Language Model



Input Representation

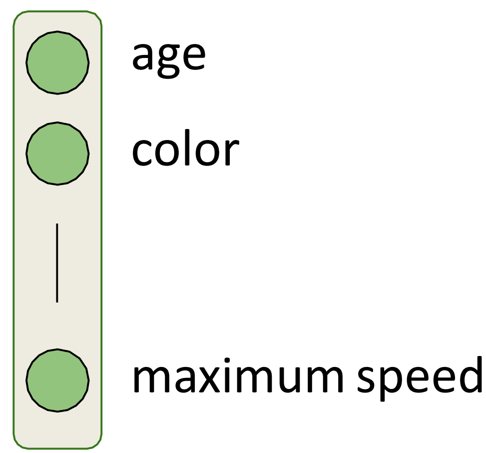
Input

Previously we've used a vector as input, where each element of the vector represented some "feature" of the input

Input Representation

Input

Previously we've used a vector as input, where each element of the vector represented some "feature" of the input

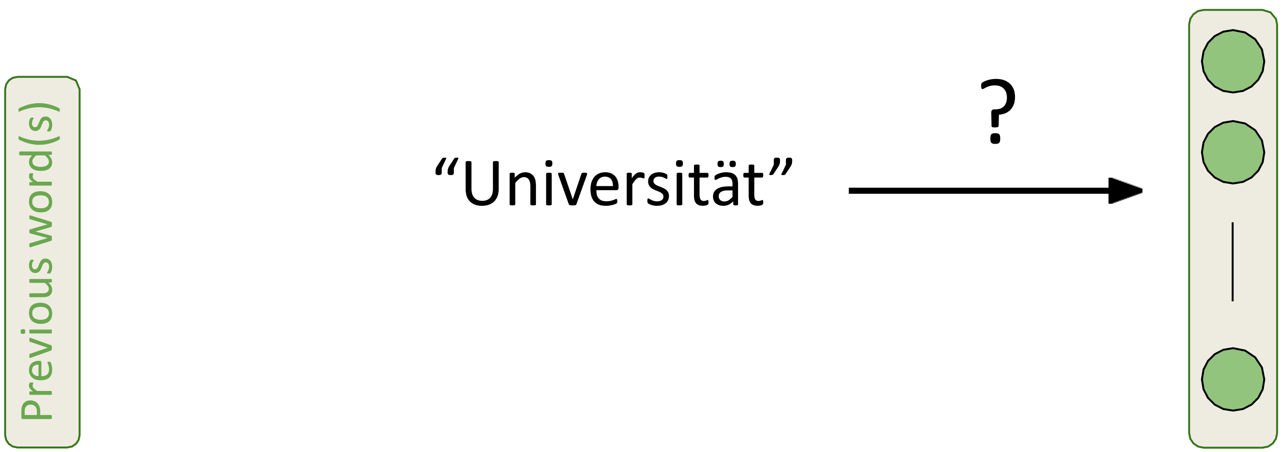


Car

Input Representation

Input

Can we represent a word as a feature vector?



One Hot Vector Representation

- Every word can be represented as a ***one hot vector***

One Hot Vector Representation

- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000

One Hot Vector Representation

- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000
- Assign each word a unique index:

```
Universität: 1
cat: 2
house: 3
car: 4
    ⋮
apple: 10,000
```

One Hot Vector Representation

- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000
- Assign each word a unique index:

```
Universität: 1
cat: 2
house: 3
car: 4
    ⋮
apple: 10,000
```

Dictionary

$$cat = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

One-hot representation

One Hot Vector Representation

- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000
- Assign each word a unique index:

Only index that represents the input word will be one

$$cat = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

One-hot representation

One Hot Vector Representation

- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000
- Assign each word a unique index:

Only index that represents the input word will be one

$$cat = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad car = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

One-hot representation

One Hot Vector Representation

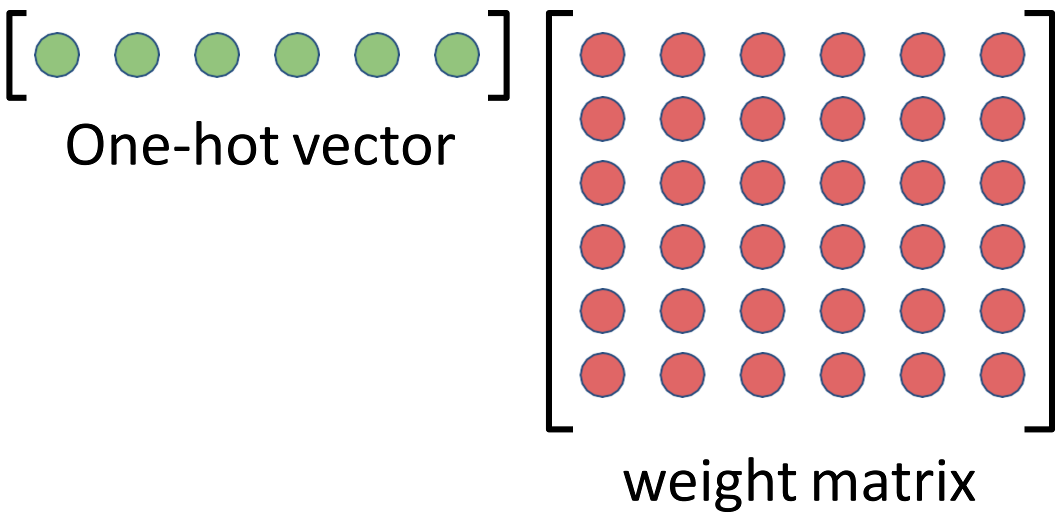
- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000
- Assign each word a unique index:

Vector size will be the size of the vocabulary, i.e. 10,000 in this case

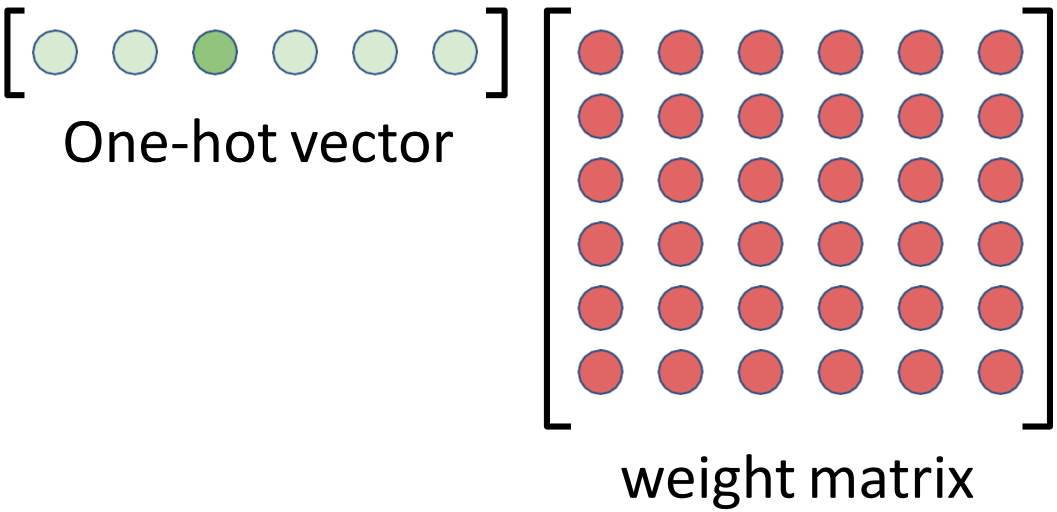
$$cat = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad car = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

One-hot representation

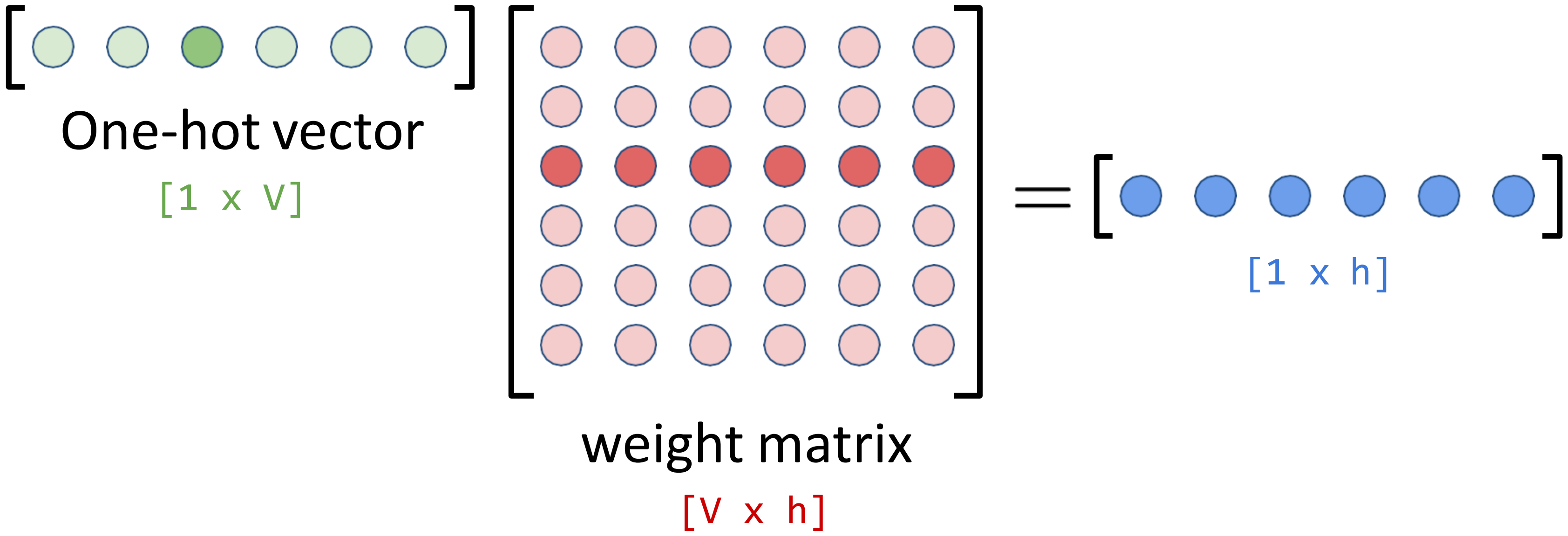
One Hot Vector Representation



One Hot Vector Representation



One Hot Vector Representation



One-hot vector will “turn on” one row of weights

Higher ngram Vector Representation

- What about representing multiple words?

Bag of words approach

Higher ngram Vector Representation

- What about representing multiple words?

Bag of words approach

Bigram: indices of the *two previous words* are 1 in the vector

$$\text{a car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \qquad \text{a cat} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

Higher ngram Vector Representation

- What about representing multiple words?

Bag of words approach

Bigram: indices of the *two previous words* are 1 in the vector

$$\text{a car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \qquad \text{a cat} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

Higher ngram Vector Representation

- What about representing multiple words?

Bag of words approach

Trigram: indices of the *three previous words* are 1 in the vector

$$\text{a fast car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \quad \text{a fast cat} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

Higher ngram Vector Representation

- What about representing multiple words?

Context-aware approach

In the **bag of words** approach, order information is lost!

Higher ngram Vector Representation

- What about representing multiple words?

Context-aware approach

In the **bag of words** approach, order information is lost!

Solution: for *N* words, concatenate one-hot vectors for each of the words in the correct order

Higher ngram Vector Representation

Context-aware approach

$$
\text{a} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \quad
\text{car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \longrightarrow \quad
\text{a car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}
$$

Higher ngram Vector Representation

Context-aware approach

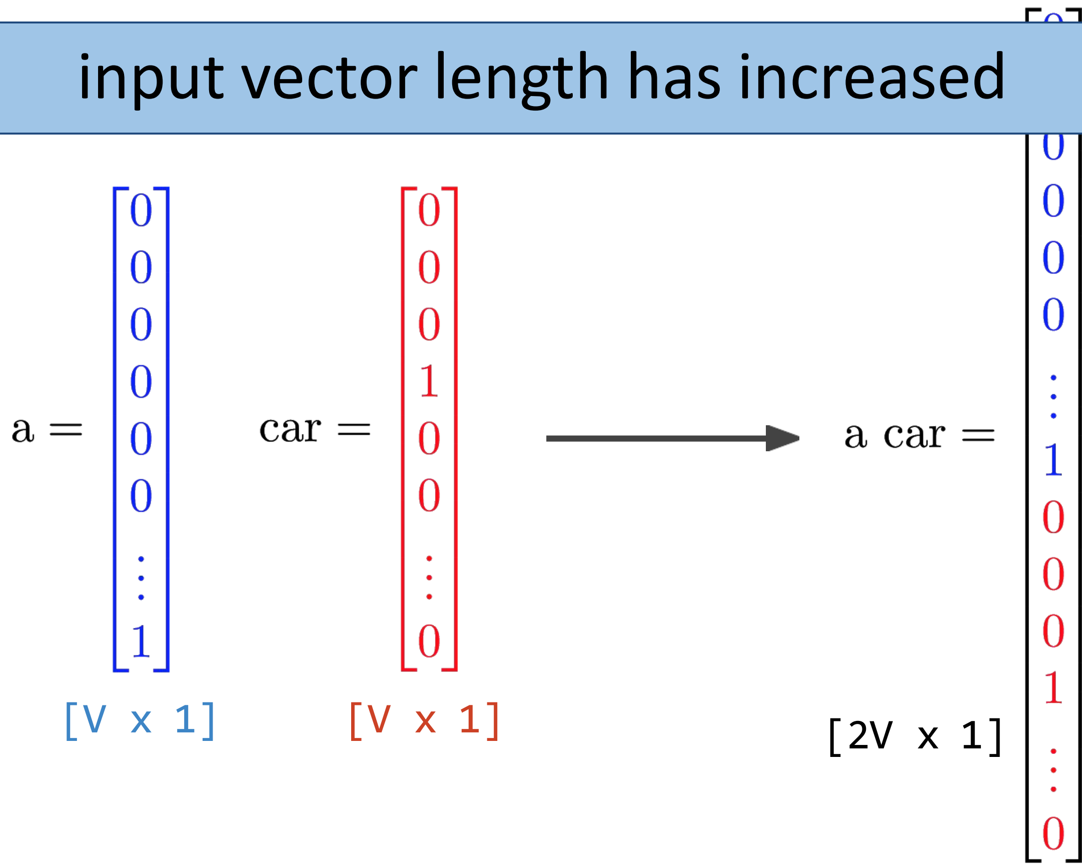
$$a = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \quad car = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \longrightarrow a\ car = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

[V x 1] [V x 1] [2V x 1]

Higher ngram Vector Representation

Context-aware approach

input vector length has increased



Higher ngram Vector Representation

Context-aware approach

input vector length has increased

- order information is available for the training

Advantages

- long vectors in case of large context size
- number of parameters increases with context size

Disadvantages

Higher ngram Vector Representation

- Bag of words vs. context-aware approach?
  - Given the disadvantages of the context-aware approach, Bag of words is more commonly used
  - Works well in practice

Input Representation

Generally, the size of the vocabulary is very large

- Results in very large one-hot vectors!

Input Representation

Generally, the size of the vocabulary is very large

- Results in very large one-hot vectors!

Some tricks to reduce vocabulary size:

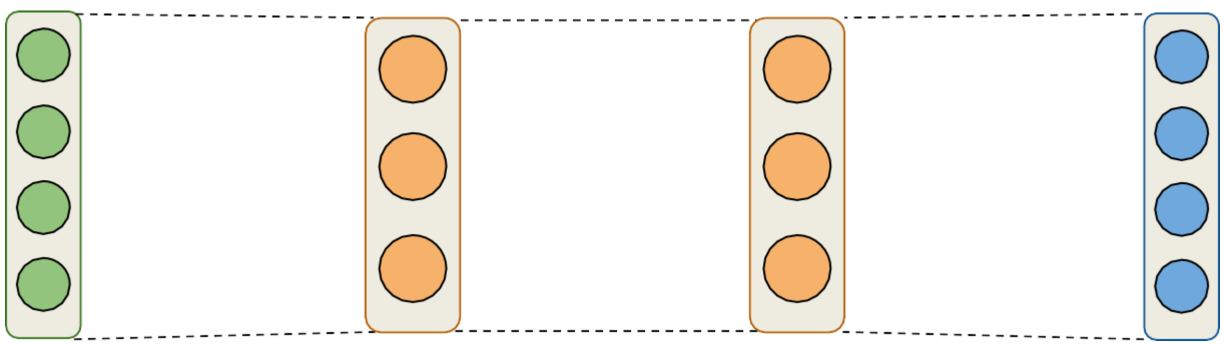
1) Take most frequent top words. For example, consider only 10,000 most frequent words and map the rest to a unique token `<UNK>`
2) Cluster words
   a) based on context
   b) based on linguistic properties

Neural Network Language Model

Let us look at a complete example:

Vocabulary: {“how”, “you”, “hello”, “are”}

Network Architecture: 2 hidden layers of size 3 each

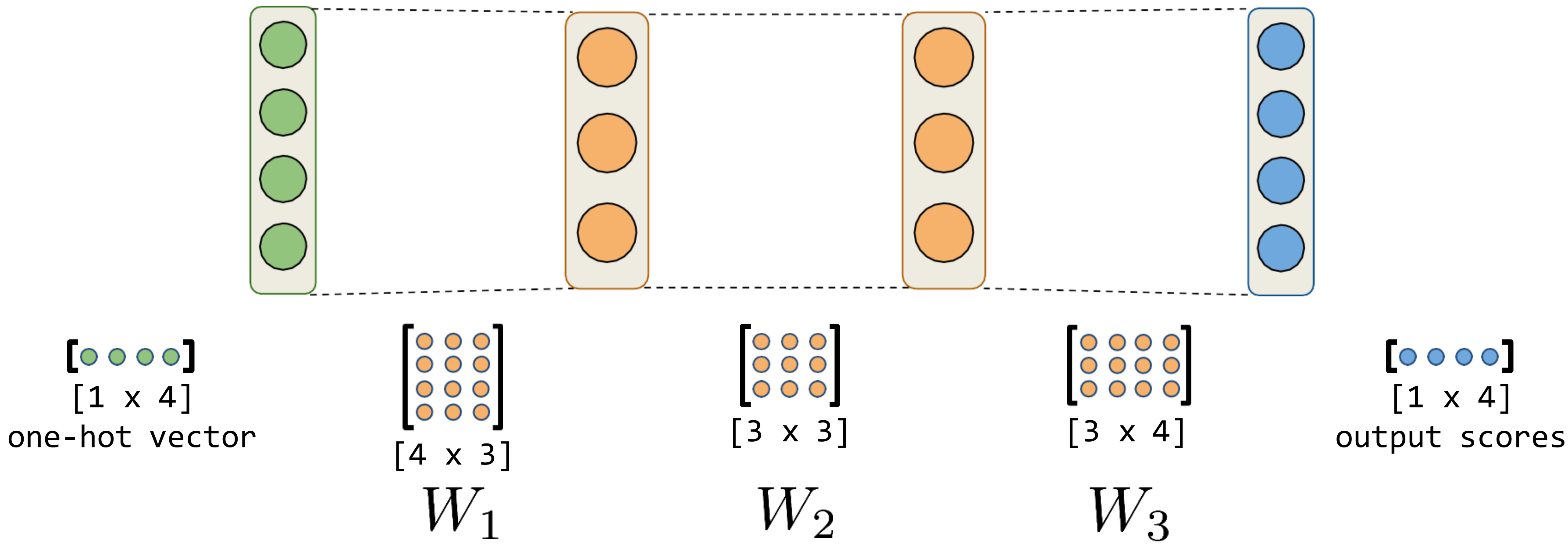


Neural Network Language Model

Let us look at a complete example:

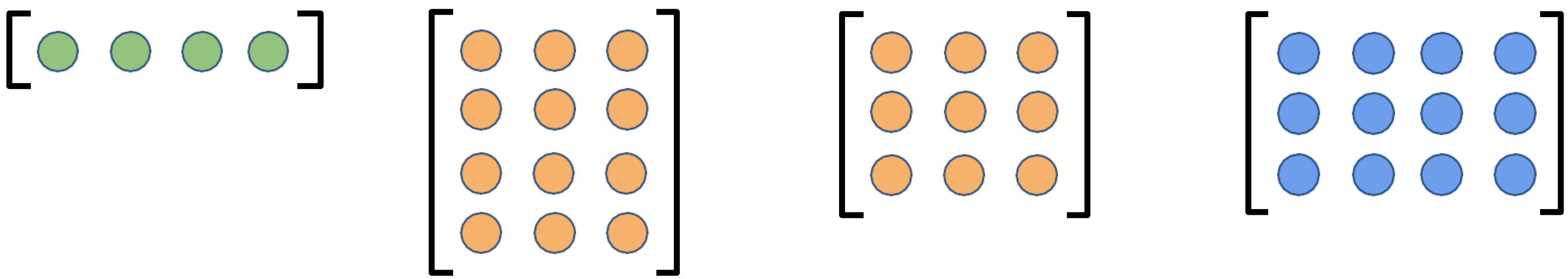
Vocabulary: {“how”, “you”, “hello”, “are”}

Network Architecture: 2 hidden layers of size 3 each



Neural Network Language Model

Vocabulary: {“how”, “you”, “hello”, “are”}



W_1 W_2 W_3

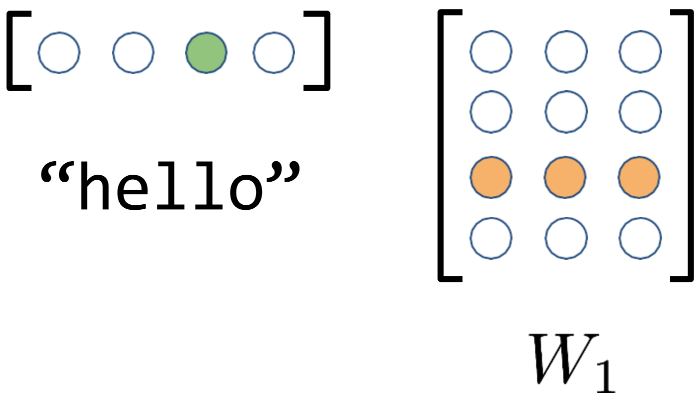
Neural Network Language Model

Vocabulary: `{"how", "you", "hello", "are"}`

"hello"

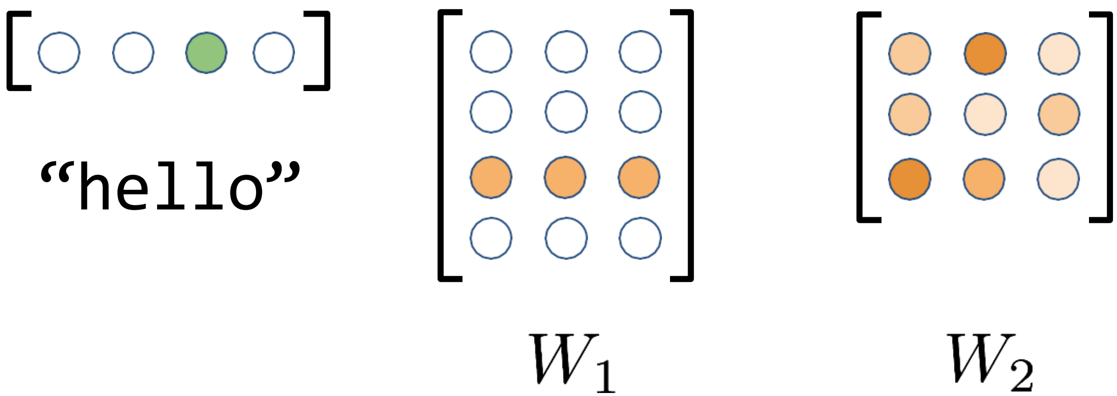
Neural Network Language Model

Vocabulary: {“how”, “you”, “hello”, “are”}



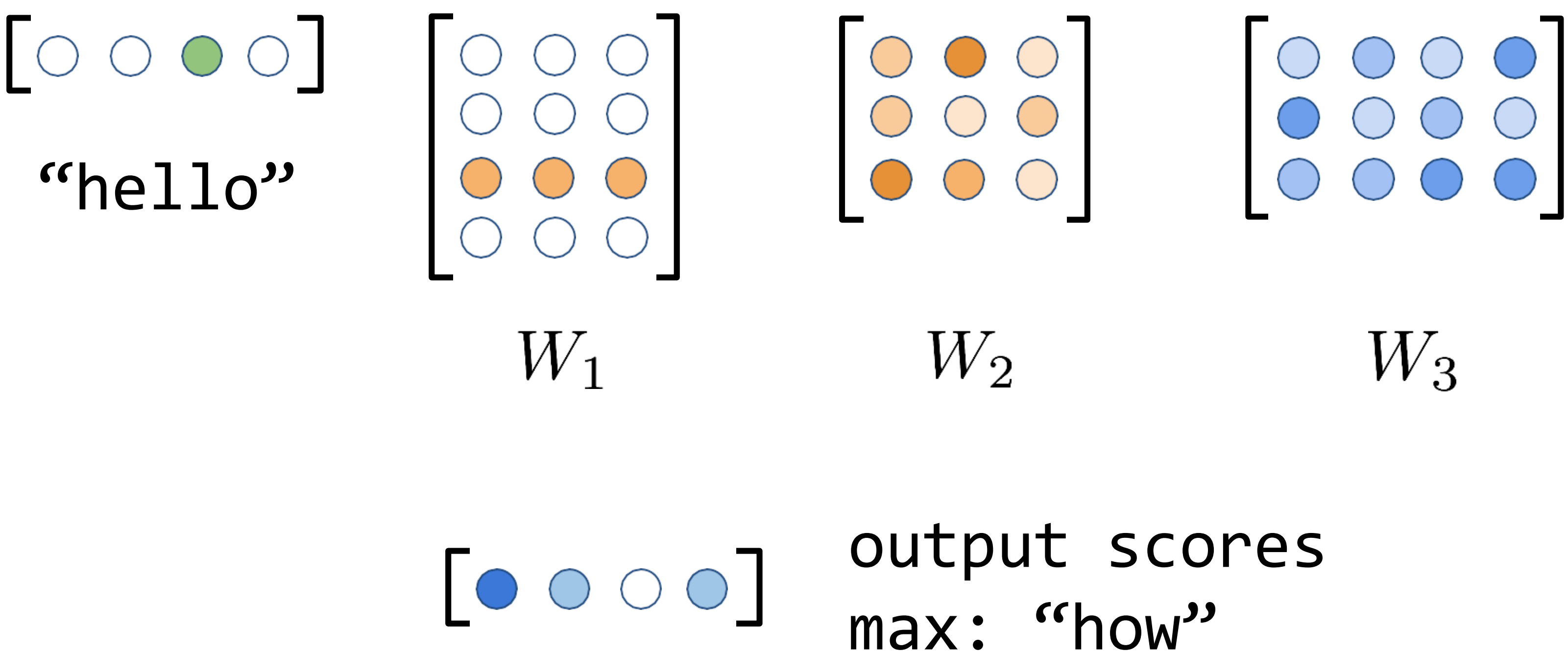
Neural Network Language Model

Vocabulary: `{“how”, “you”, “hello”, “are”}`



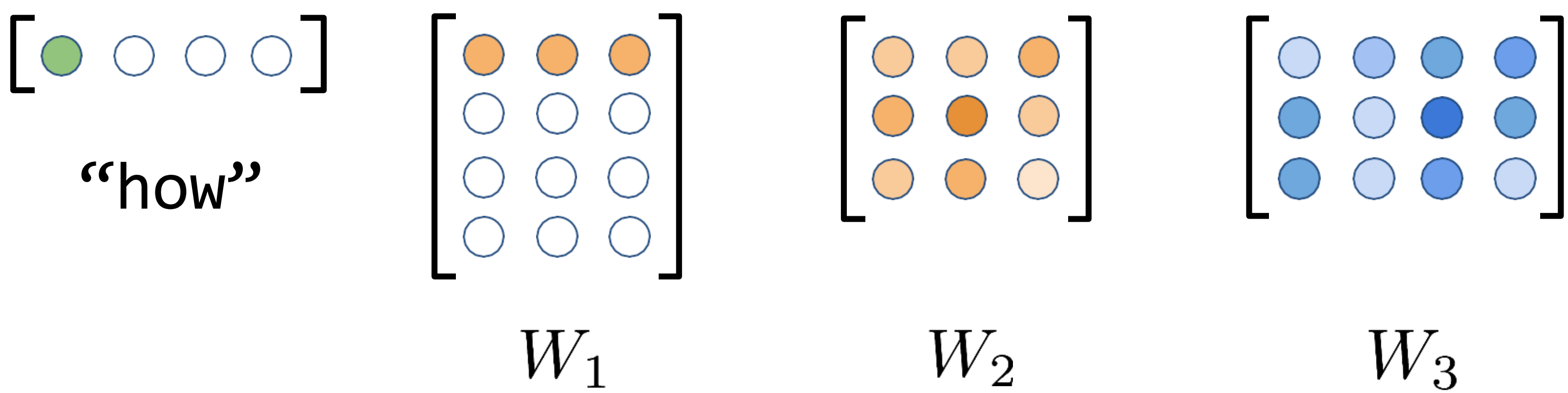
Neural Network Language Model

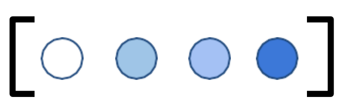
Vocabulary: {“how”, “you”, “hello”, “are”}



Neural Network Language Model

Vocabulary: {“how”, “you”, “hello”, “are”}

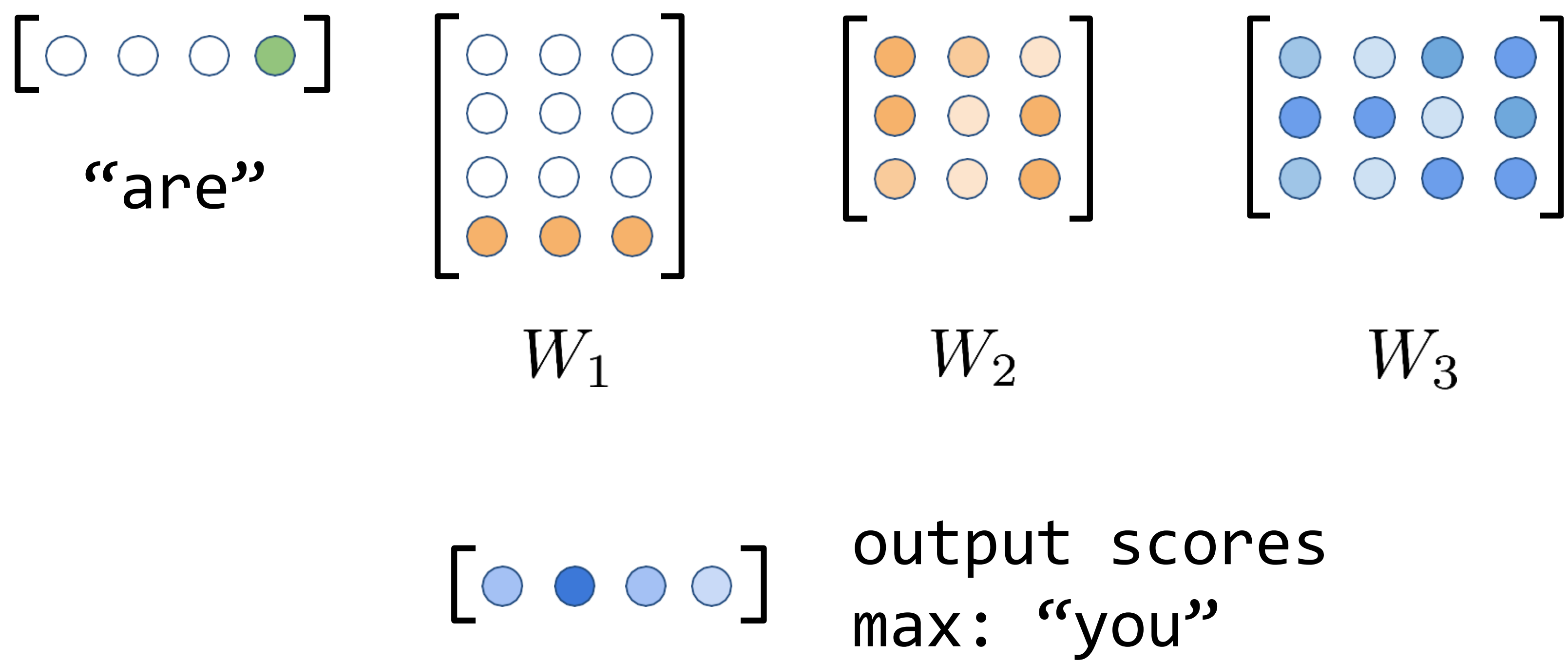




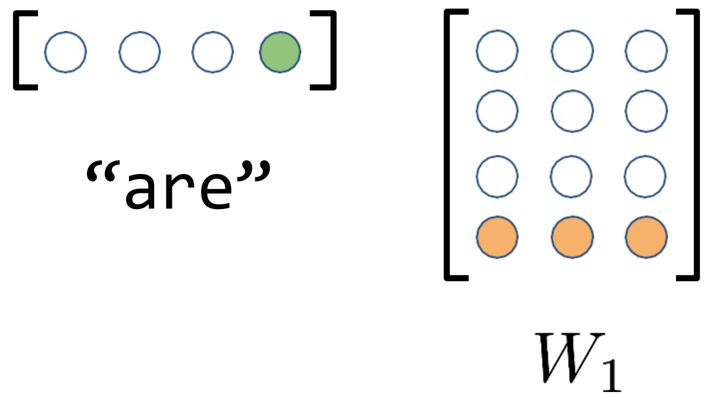
output scores
max: “are”

Neural Network Language Model

Vocabulary: {“how”, “you”, “hello”, “are”}

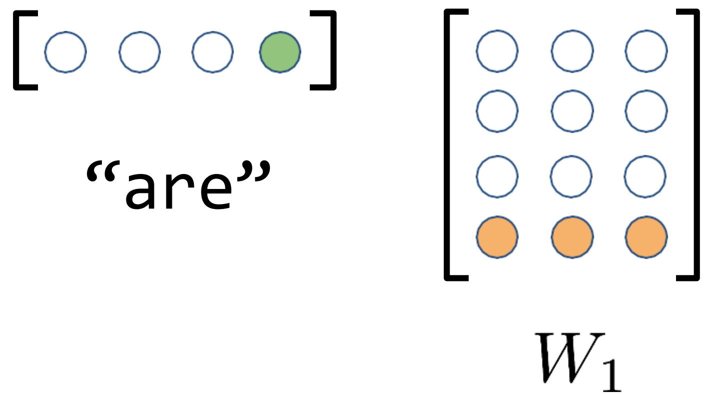


Neural Network Language Model



Each one-hot vector turns on one row in the weight matrix and results in `[1 x 3]` vector

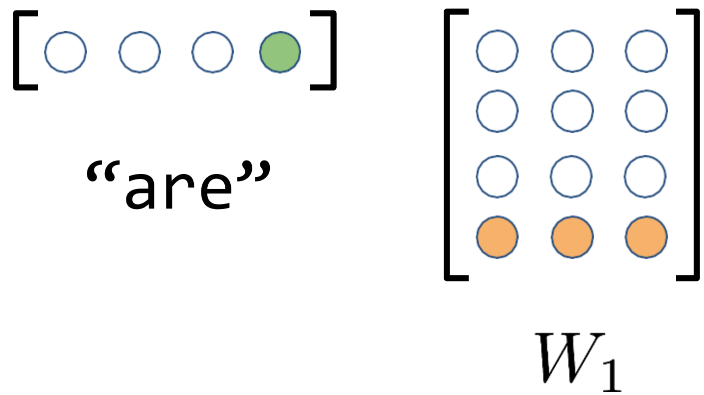
Neural Network Language Model



Each one-hot vector turns on one row in the weight matrix and results in `[1 x 3]` vector

Can we say that the `[1 x 3]` *vector represents the input word?*

Neural Network Language Model



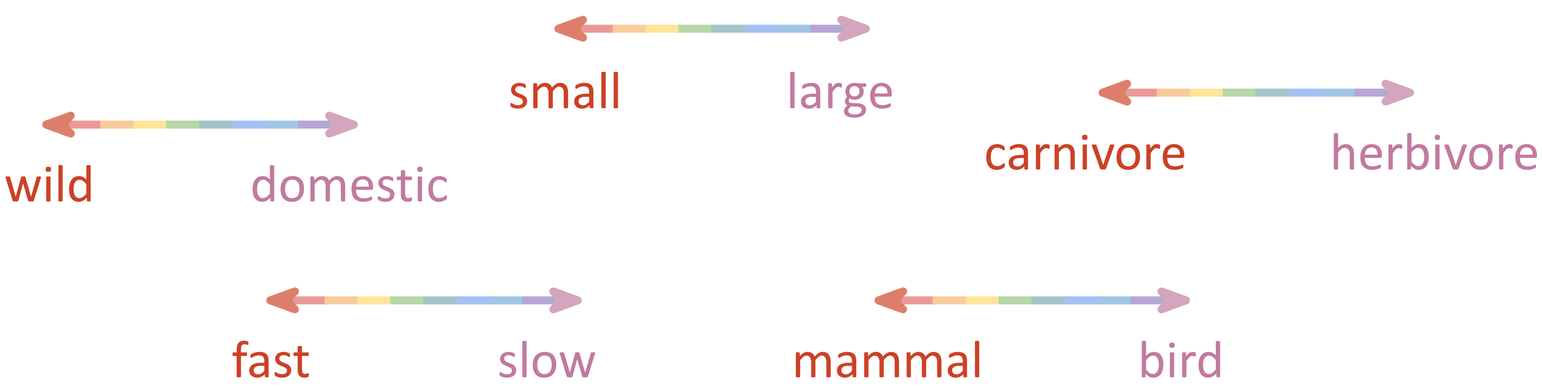
Each one-hot vector turns on one row in the weight matrix and results in `[1 x 3]` vector

Can we say that the `[1 x 3]` *vector represents the input word?* **Yes**

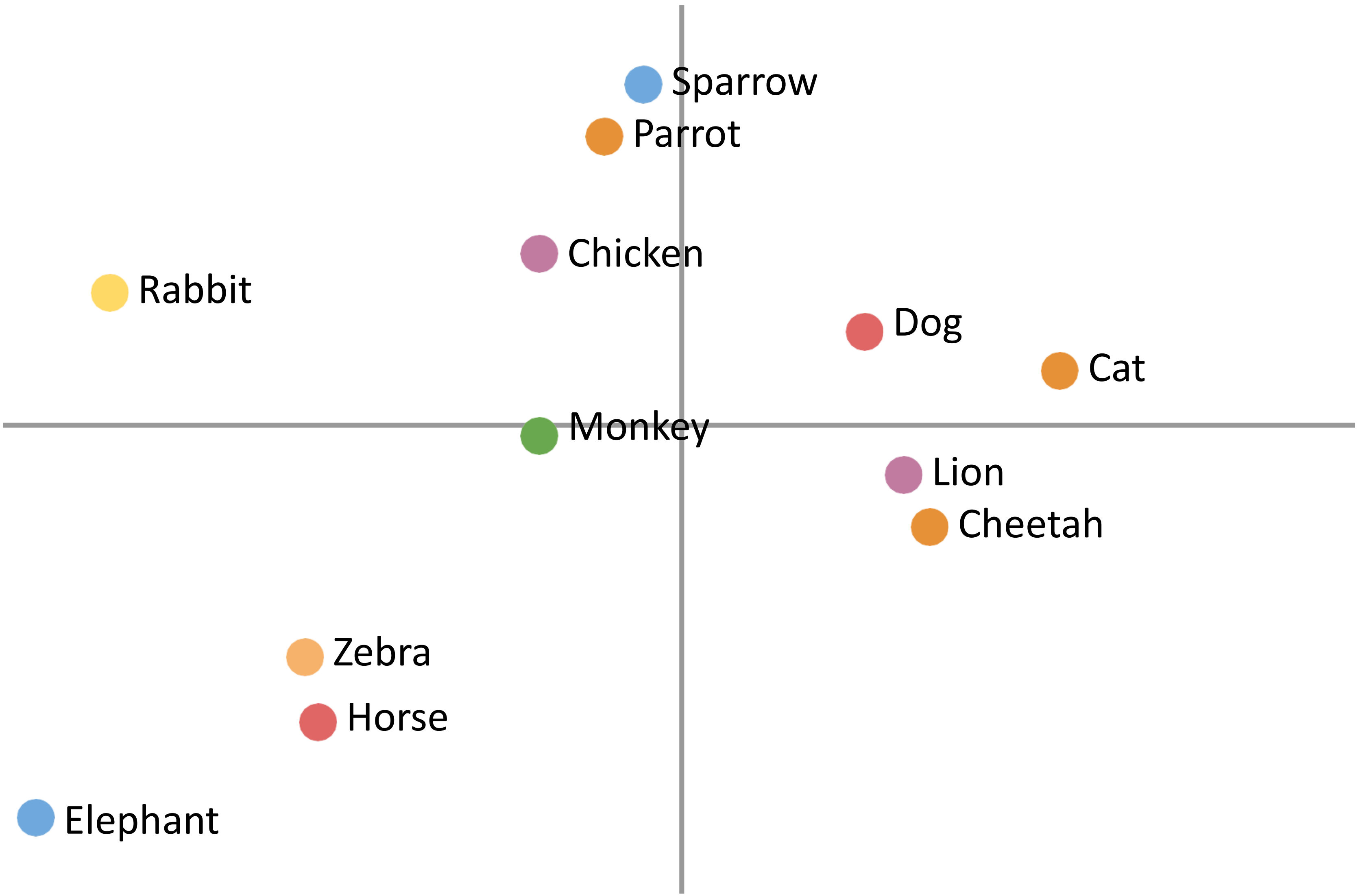
Exercise

Create a 2D vector space representation of the following words:

`dog, lion, cat, rabbit, horse, zebra, cheetah, parrot, sparrow, elephant, chicken, monkey`

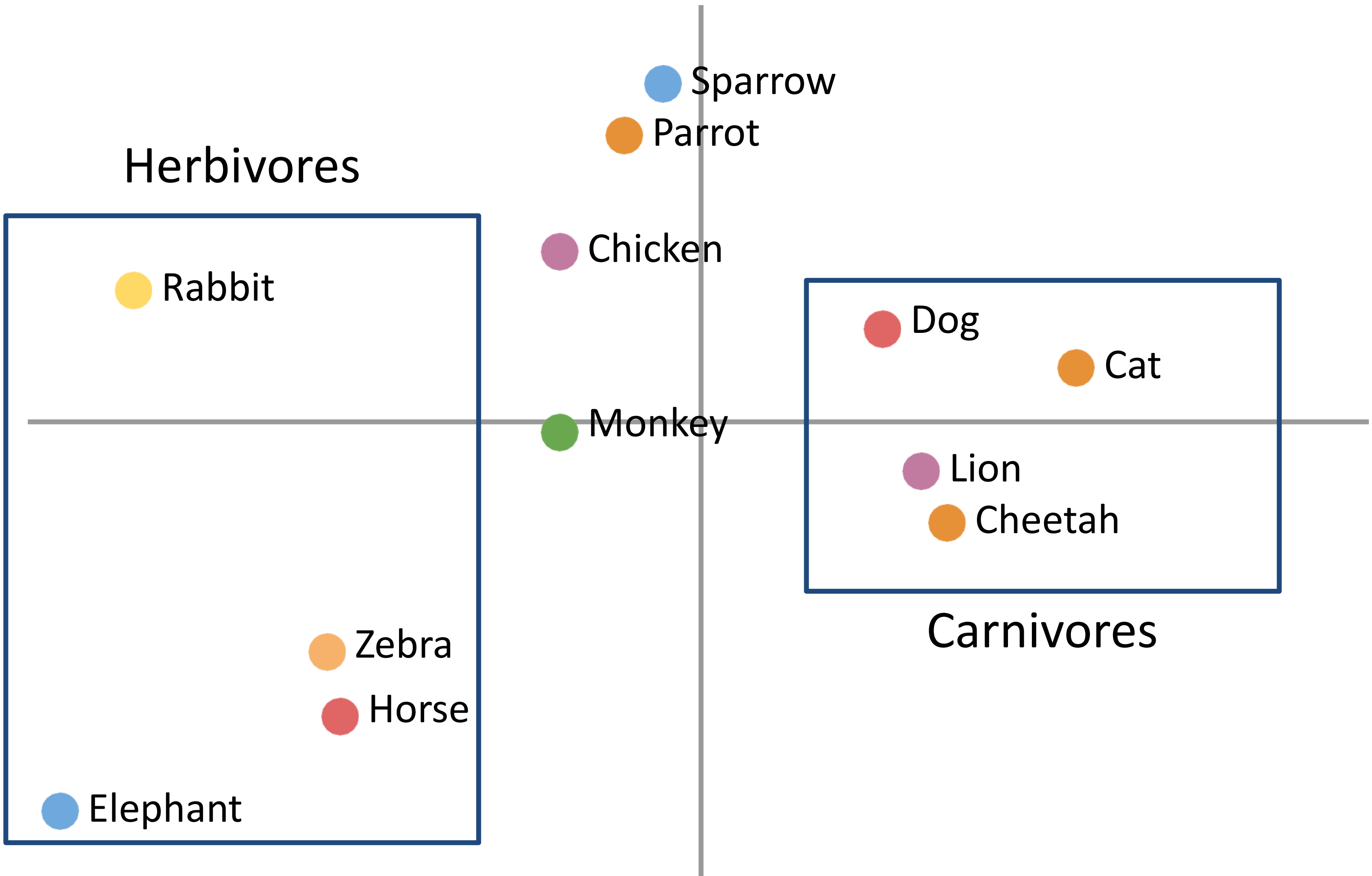


Exercise



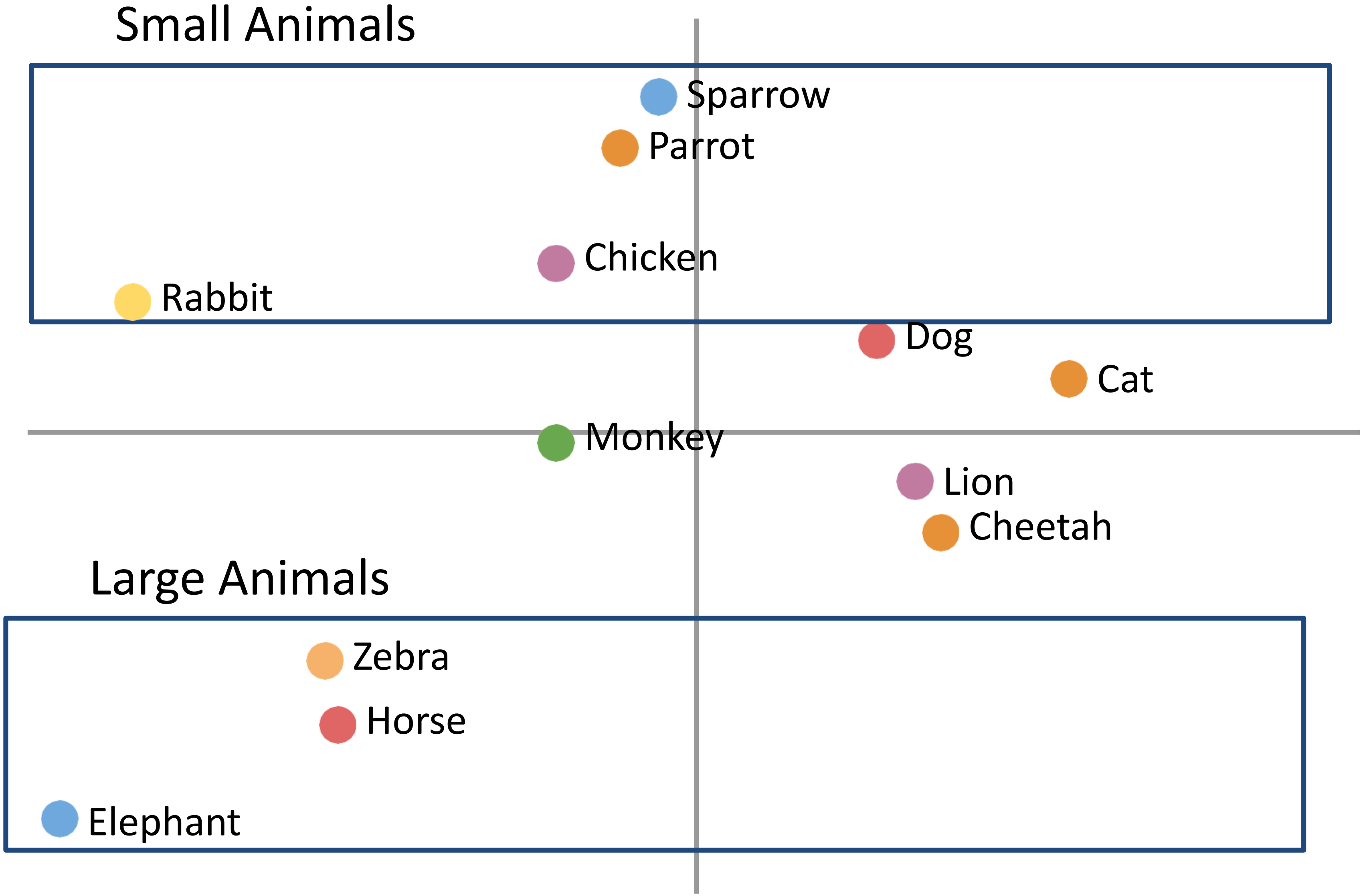
Sparrow
Parrot
Chicken
Rabbit
Dog
Cat
Monkey
Lion
Cheetah
Zebra
Horse
Elephant

Exercise



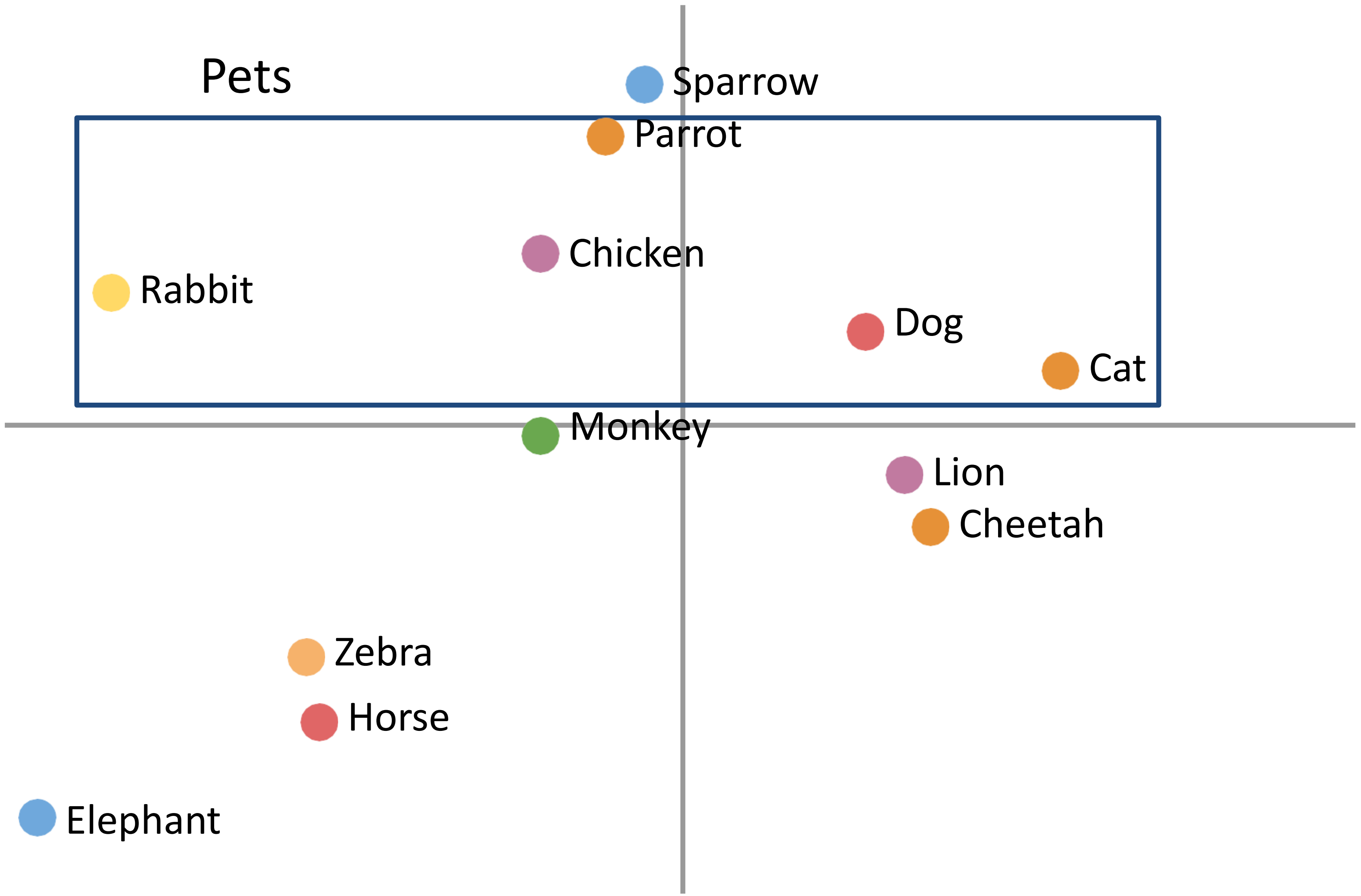
Sparrow
Parrot
Herbivores
Chicken
Rabbit
Dog
Cat
Monkey
Lion
Cheetah
Carnivores
Zebra
Horse
Elephant

Exercise



Small Animals
Sparrow
Parrot
Chicken
Rabbit
Dog
Cat
Monkey
Lion
Cheetah
Large Animals
Zebra
Horse
Elephant

Exercise


Pets
Sparrow
Parrot
Chicken
Rabbit
Dog
Cat
Monkey
Lion
Cheetah
Zebra
Horse
Elephant

Word Embeddings

How did you decide which animals need to be closer?

How did you handle conflicts between animals that belong to multiple groups?

How does having this kind of vector space representation help us?

Word Embeddings

- In one-hot vector representation, a word is represented as one large ***sparse*** vector

only one element is 1 in the entire vector

vectors of different words do not give us any information about the potential relations between the words!

Word Embeddings

- In one-hot vector representation, a word is represented as one large ***sparse*** vector
- Instead, **word embeddings** are ***dense*** vectors in some vector space

Word Embeddings

- In one-hot vector representation, a word is represented as one large ***sparse*** vector
- Instead, **word embeddings** are ***dense*** vectors in some vector space

word vectors are ***continuous*** representations of words

vectors of different words give us information about the potential relations between the words - words closer together in meaning have vectors closer to each other

Word Embeddings

$$\text{cat} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \longrightarrow \text{cat} = \begin{bmatrix} 0.523 \\ -0.813 \\ 0.417 \\ 1.738 \\ -0.145 \\ -3.451 \\ \vdots \\ 0.451 \end{bmatrix}$$

one-hot vector

word embedding

Word Embeddings

"Representation of words in continuous space"

Inherit benefits

- Reduce dimensionality
- Semantic relatedness
- Increase expressiveness
  - one word is represented in the form of several features (numbers)

Word Embeddings

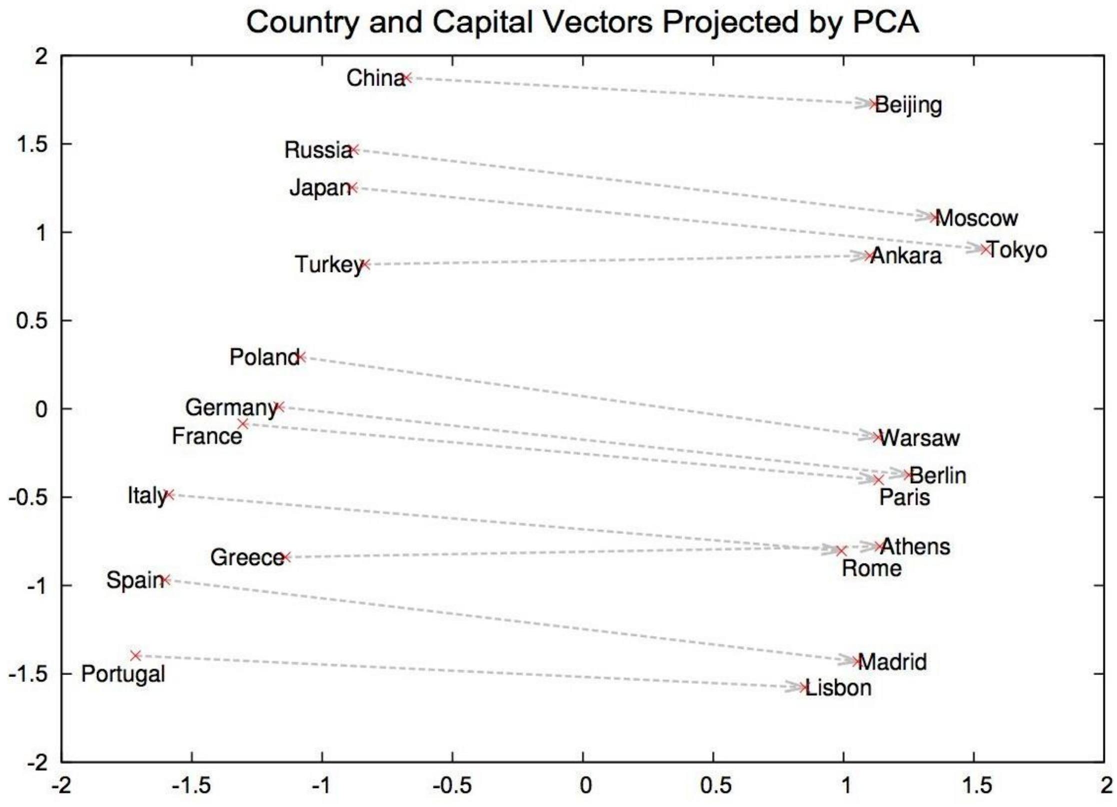
Play with some embeddings!

https://rare-technologies.com/word2vec-tutorial/#bonus_app

Try various relationships...

Word Embeddings

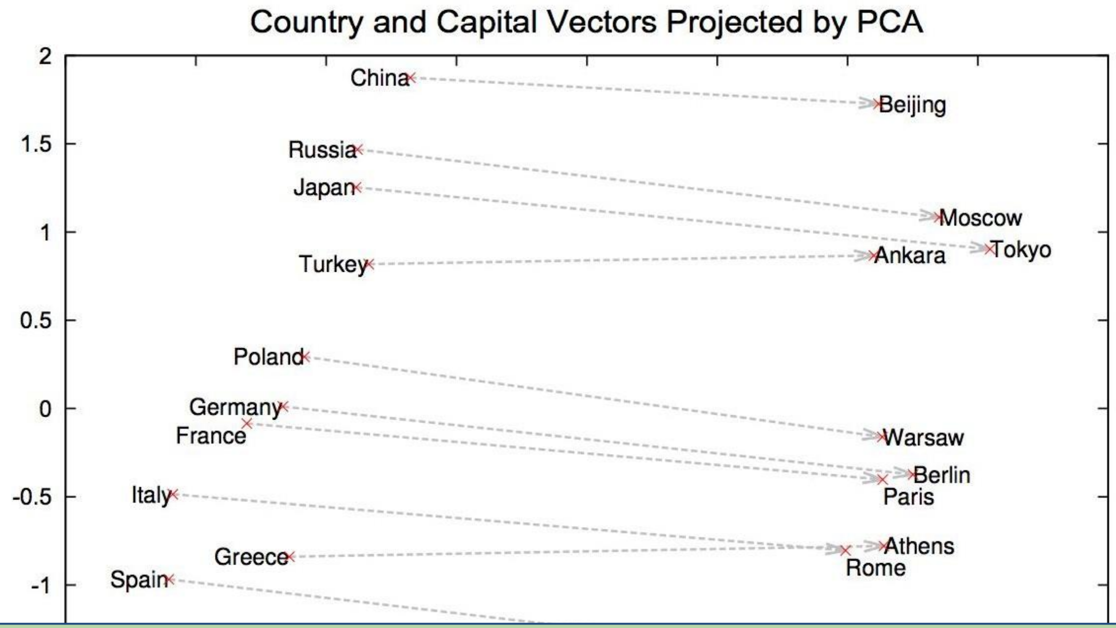
- Plot the embedding vectors



Mikolov et al. 2013

Word Embeddings

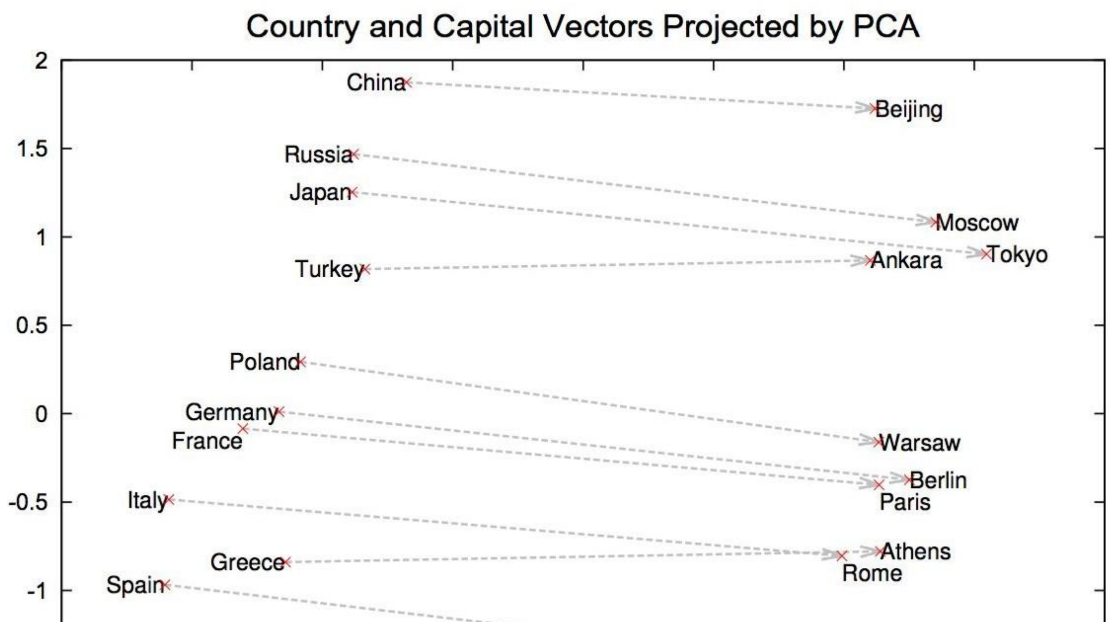
- Plot the embedding vectors



Plot shows the relationship between vectors representing related concepts

Word Embeddings

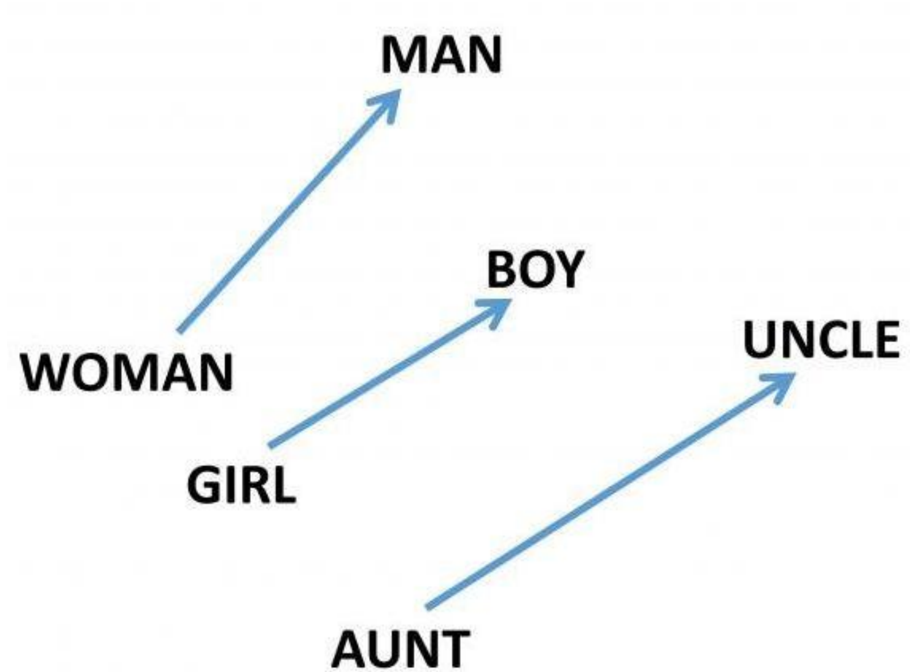
- Plot the embedding vectors



The vectors from countries to capitals point roughly in the same direction

Word Embeddings

- Similarly, learning the gender relationship



Word Embeddings

Q: How can we learn these embeddings automatically?

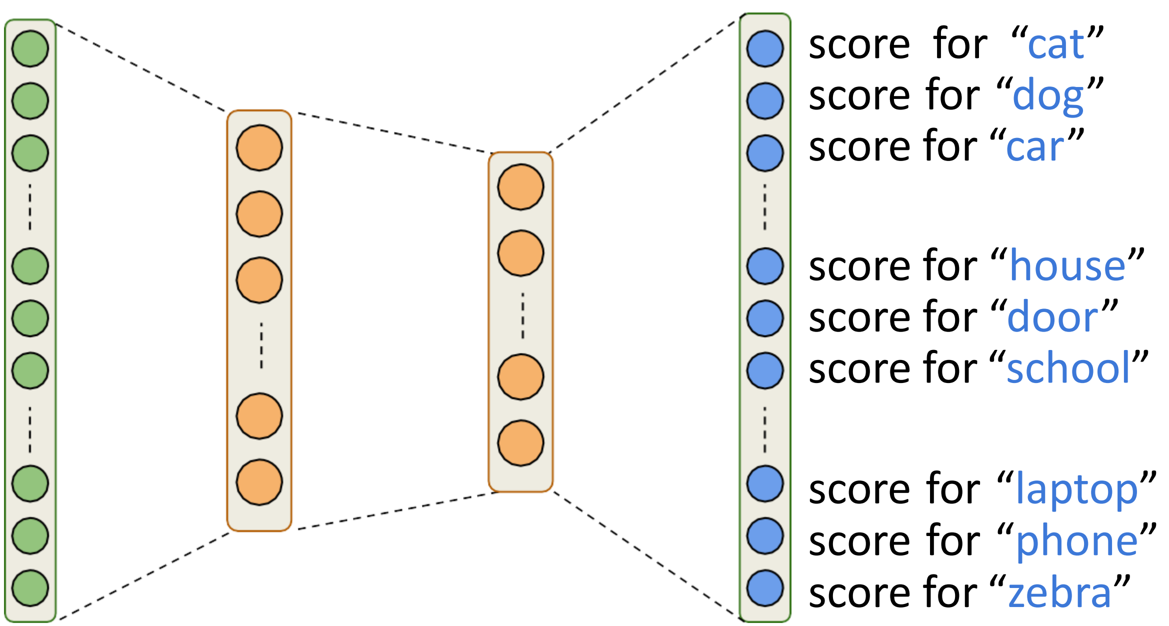
Word Embeddings

Q: How can we learn these embeddings automatically?

A: Neural Networks are a step ahead - embeddings are already learned as “richer” features

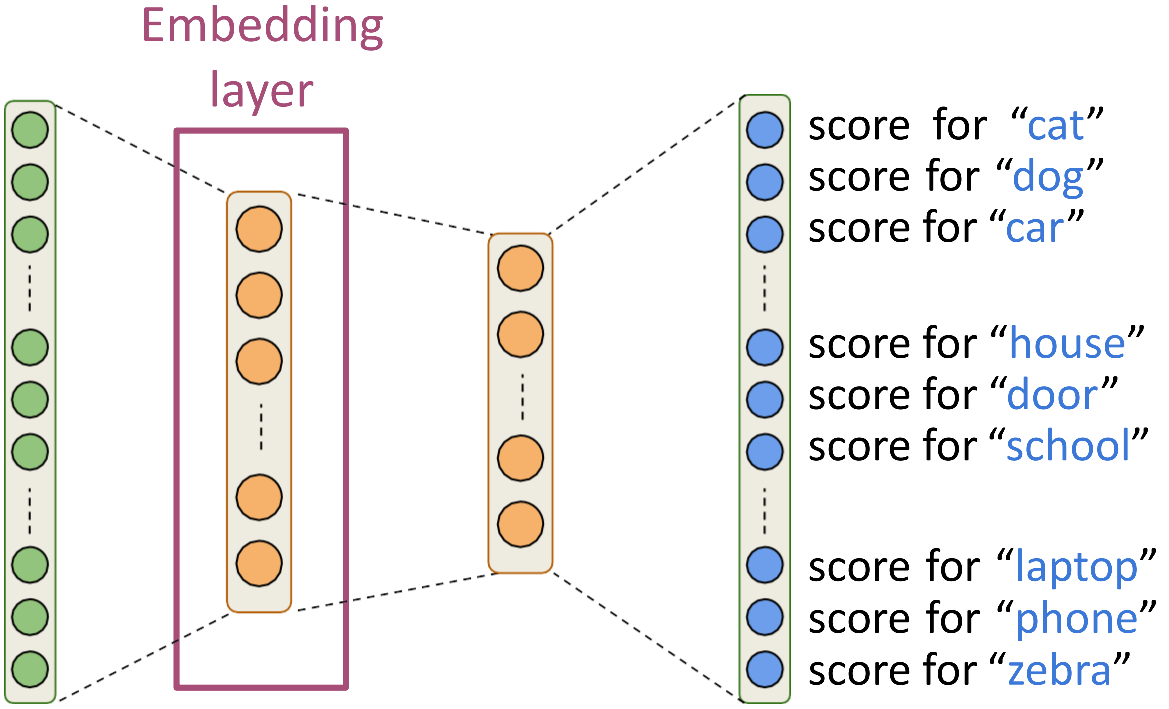
Word Embeddings

Neural Networks are a step ahead - embeddings are already learned as “richer” features



Word Embeddings

Neural Networks are a step ahead - embeddings are already learned as “richer” features



Word Embeddings

The overall **training task** defines the relationships which will be learned by the model

For example:

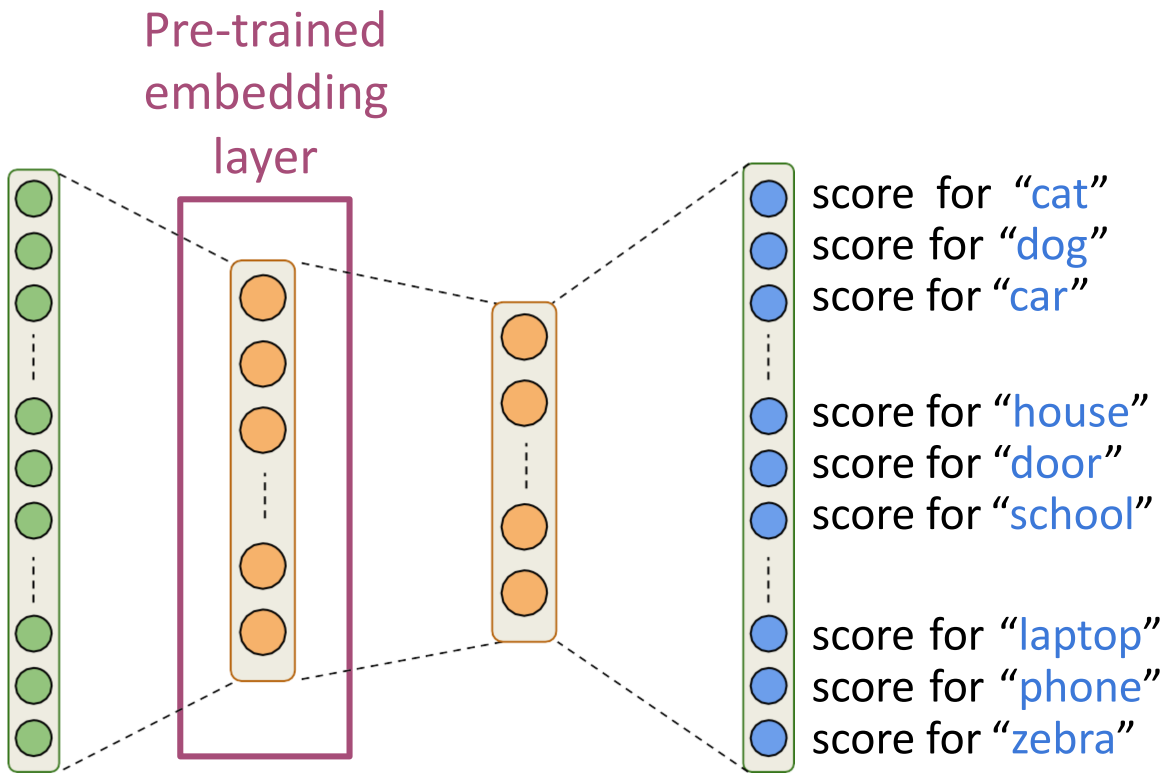
- In language modeling, the model uses neighboring context thus bringing words with similar context closer
- In doing POS tagging task, words with similar POS tags will come close to each other
- If our network is doing machine translation, the embeddings will be tuned for translation

Word Embeddings

- Generally, task specific embeddings are better than generic embeddings
- In case of small amount of training data, generic embeddings learned on large amount of data works better
- Generic embeddings can also be used as a starting point

Word Embeddings

We can use pre-trained embeddings as well - just initialize the weights in the first layer with some learned embeddings



Word Embedding Tools

Some tools to learn word embeddings:

- Word2Vec (from Google)
- FastText (from Facebook)
- GloVe (from Stanford)

Word Embeddings

A few pre-trained word embeddings

- GloVe: Wikipedia plus Gigaword https://goo.gl/1XYZhc
- FastText: Wikipedia of 294 languages https://goo.gl/1v423g
- Dependency-based https://goo.gl/tpgw4R

Using pre-trained embeddings in keras:

https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html