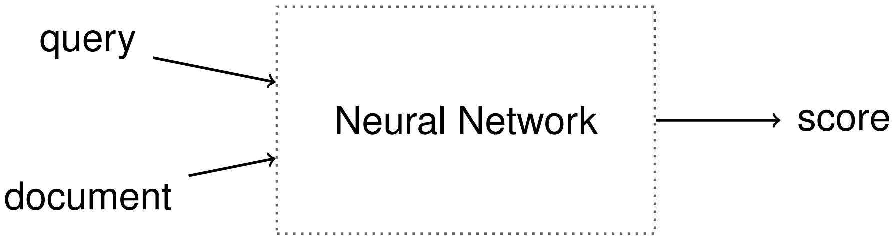
Word Embeddings
Overview

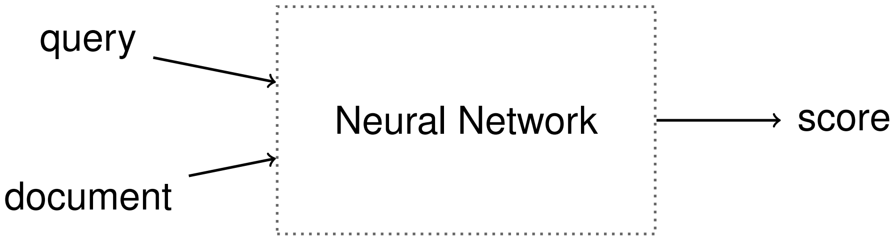
Goal:



Problem: How do we represent text so we can feed it to the neural network?

Word Embeddings
Overview

Goal:



Problem: How do we represent text so we can feed it to the neural network?

Solution: Turn words into numbers.

Word Embeddings
Representing Words

`apples are great`

Word Embeddings
Representing Words

`apples are great`

Assign each word a random value.

- apples ➜ 6.3
- are ➜ -3.5
- great ➜ 4.2

Word Embeddings

Representing Words

```
apples are great

apples are awesome
```

Assign each word a random value.

- apples ➜ 6.3
- are ➜ -3.5
- great ➜ 4.2
- awesome ➜ -32.1

Word Embeddings
Representing Words

`apples are great`

`apples are awesome`

Assign each word a random value.

- apples ➜ 6.3
- are ➜ -3.5
- great ➜ 4.2
- awesome ➜ -32.1

Problems:

- `great` and `awesome` mean similar things and used in similar ways.
- They are likely to have very different values.
- Bad for neural networks, requiring more complexity and training.

Word Embeddings

Developing a Better Representation

How can we let similar words have similar values?

➜ Learning how to use one word helps use the other at the same time.

Word Embeddings

Developing a Better Representation

How can we let similar words have similar values?

➜ Learning how to use one word helps use the other at the same time.

Words can be used in many contexts, pluralised, and so on.

➜ Assign each word multiple values for different contexts.

Word Embeddings

Developing a Better Representation

How can we let similar words have similar values?

➜ Learning how to use one word helps use the other at the same time.

Words can be used in many contexts, pluralised, and so on.

➜ Assign each word multiple values for different contexts.

How to decide which words are similar? How to learn multiple values?

➜ Neural network + clever training.

Word Embeddings

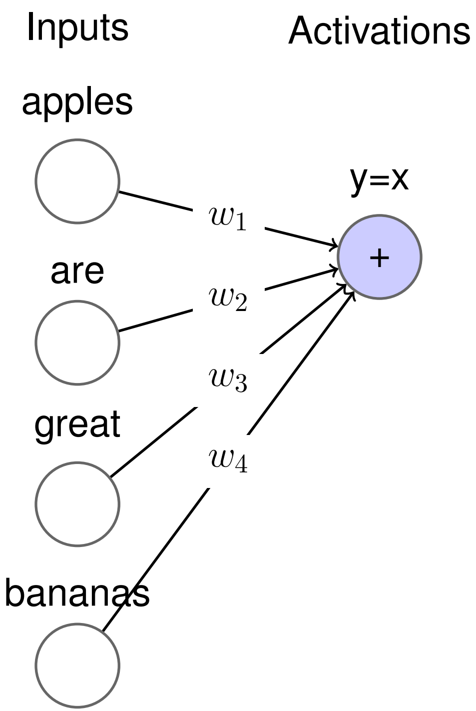
Training a Neural Network

Training data: `apples are great, bananas are great.`

Word Embeddings

Training a Neural Network

Training data: `apples are great`, `bananas are great`.

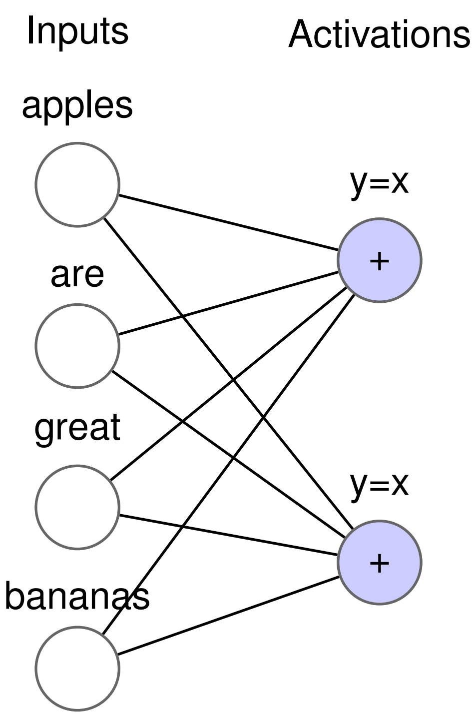


- Four unique inputs, each corresponding to a word.
- Linear activation function does nothing, just a place to do addition.
- Weights randomly initialised and optimised with backpropagation.

Word Embeddings

Training a Neural Network

Training data: `apples are great, bananas are great.`

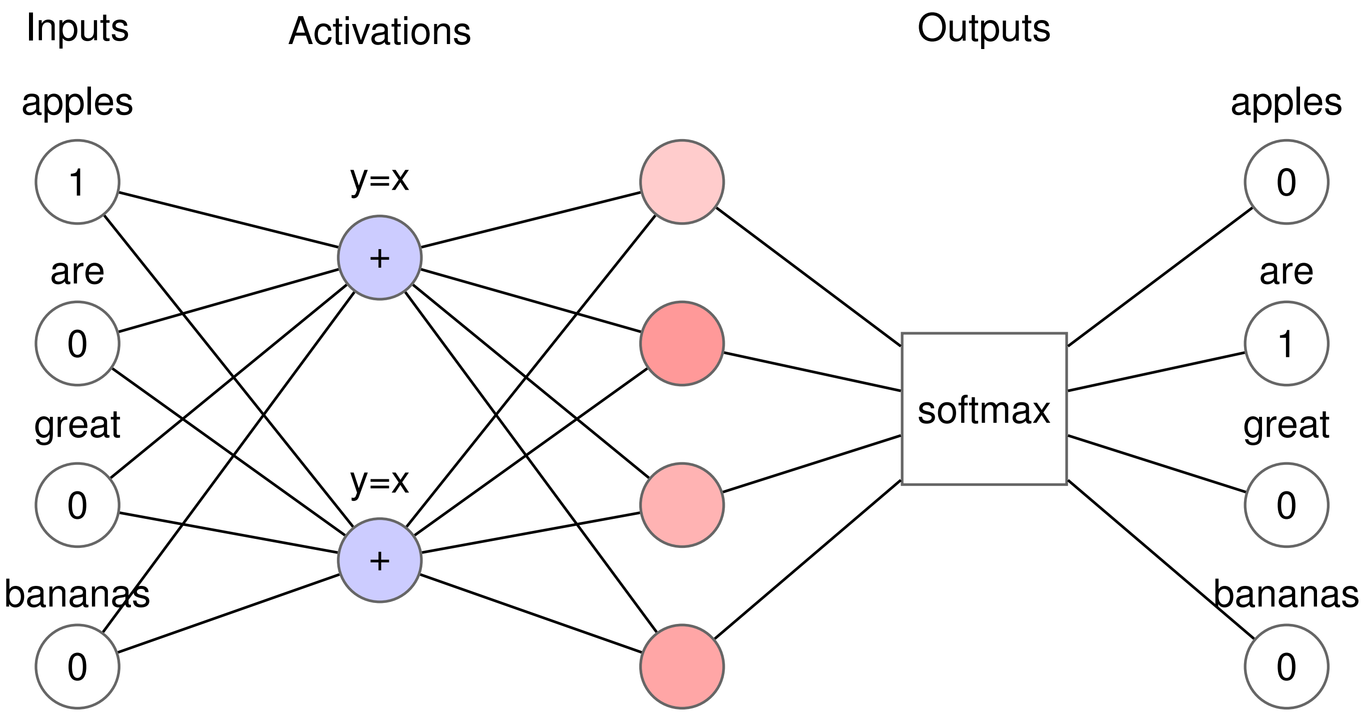


- To represent words with multiple values, add additional activation functions.
- Each activation function is associated with another weight for each word.

Word Embeddings

Training a Neural Network

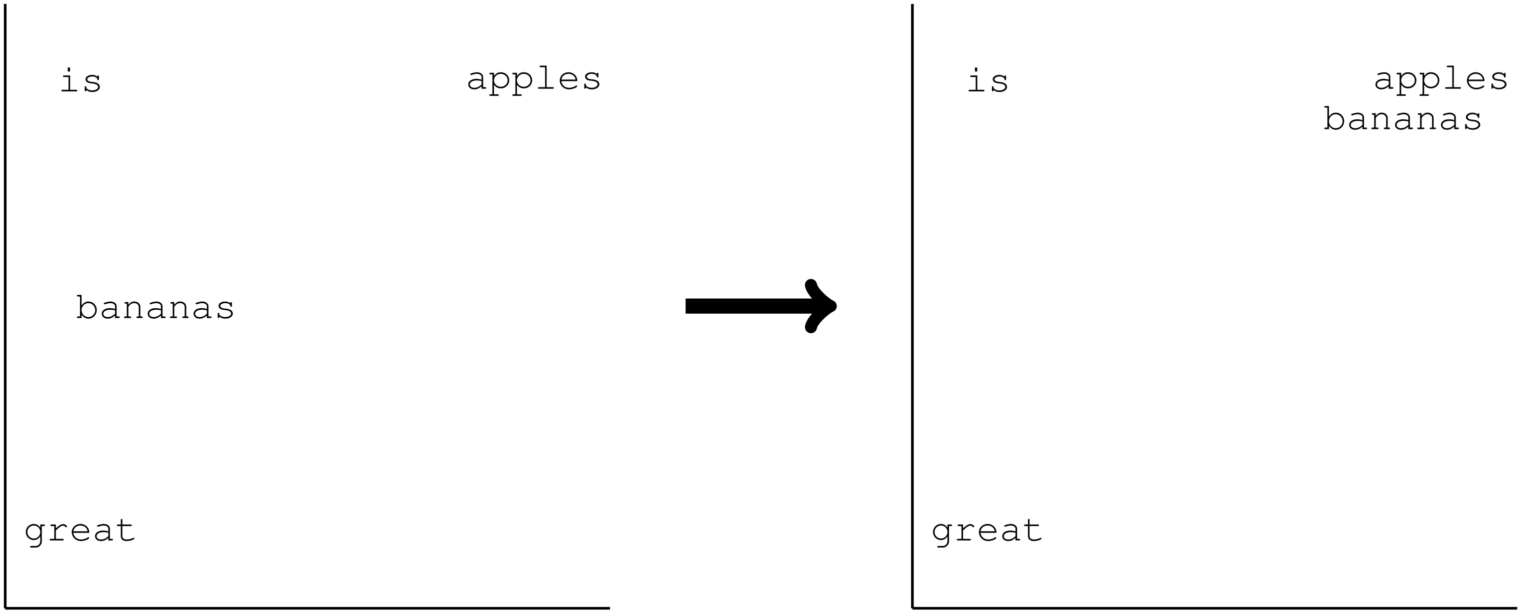
Training data: `apples are great, bananas are great.`



- Use input word to predict next word in phrase ➜ `apples`
- We want the largest output value after softmax to be the target word.
- Cross entropy loss with backpropagation to optimise weights.

Word Embeddings

Visualising Word Embeddings



- Weights going into activation layer are the values associated with each word.
- When words appear in similar contexts, values (weights) become similar.
- All the weights for a given word are called the **word embedding**.

Word Embeddings
Summary

Word embeddings let us represent text as values for machine learning problems.

- ❑ Rather than using random values, use a neural network to learn values.
- ❑ Use context of words in training dataset to optimise weights for embeddings.
- ❑ Similar words get similar embeddings, which helps with training.

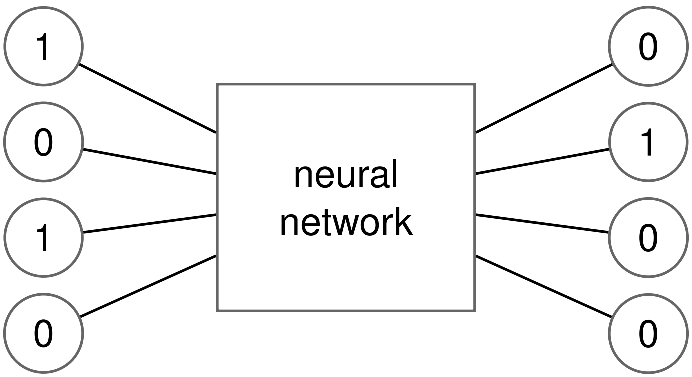
Problem: Just predicting the next word doesn’t provide much context.

Word Embeddings

word2vec

Continuous Bag of Words (CBOW)

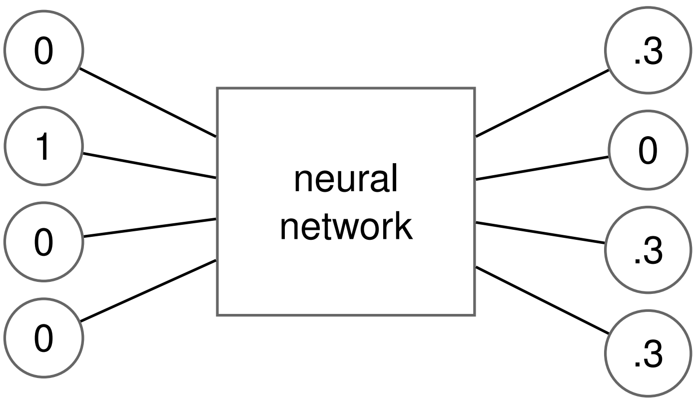
➜ Increase context by using surrounding words to predict what occurs in the middle.



Word Embeddings
word2vec

Skip gram

➜ Increase context by using word in the middle to predict surrounding words.



Word Embeddings
Efficiently Training word2vec

- In practice, there are hundreds of activation functions.
- And significantly more training data (e.g., all of Wikipedia).
- Vocabulary (input size) is much larger, typically 3,000,000 words and phrases.

Total weights to optimise:

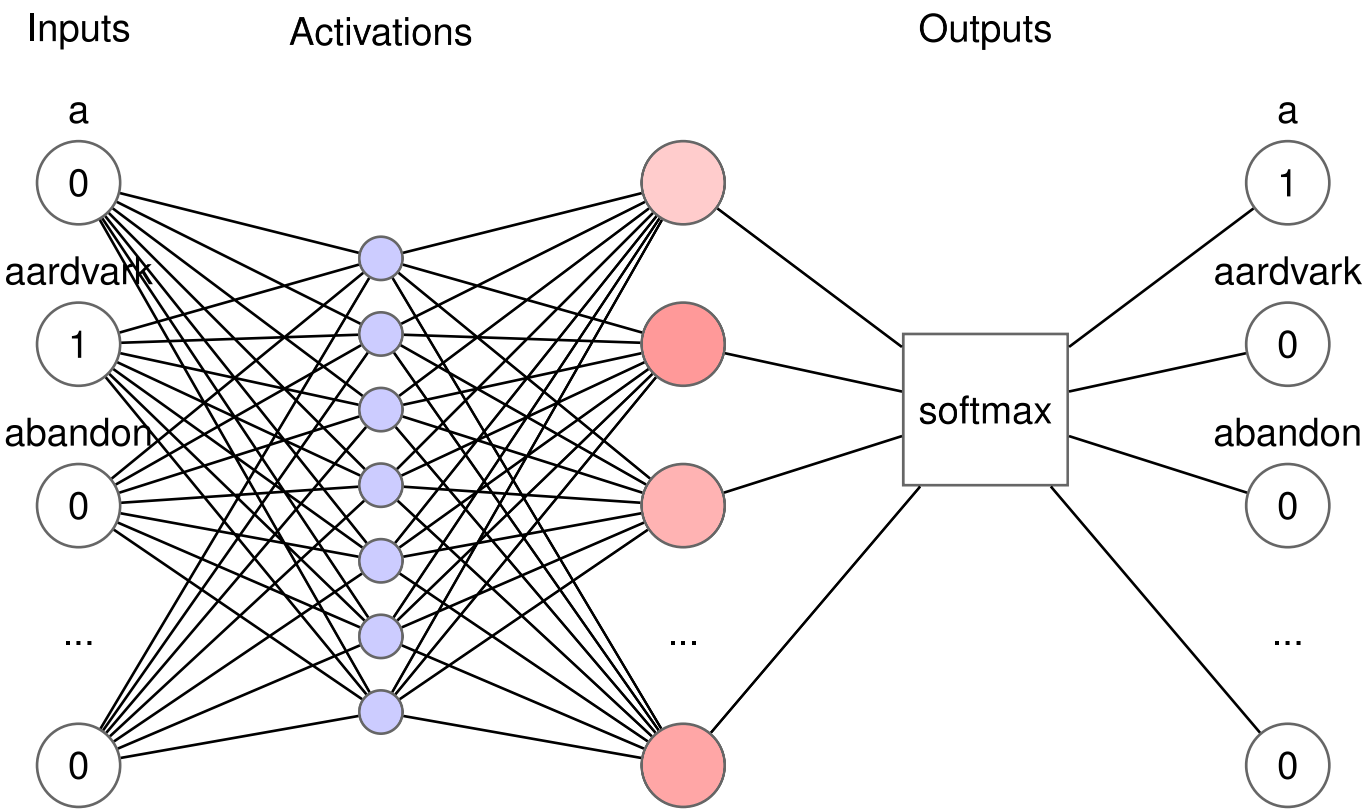
$$3,000,000 \cdot 100 \cdot 2 = 600,000,000$$

3M words, 100 activations (times 2 for input+output).

Solution: negative sampling.

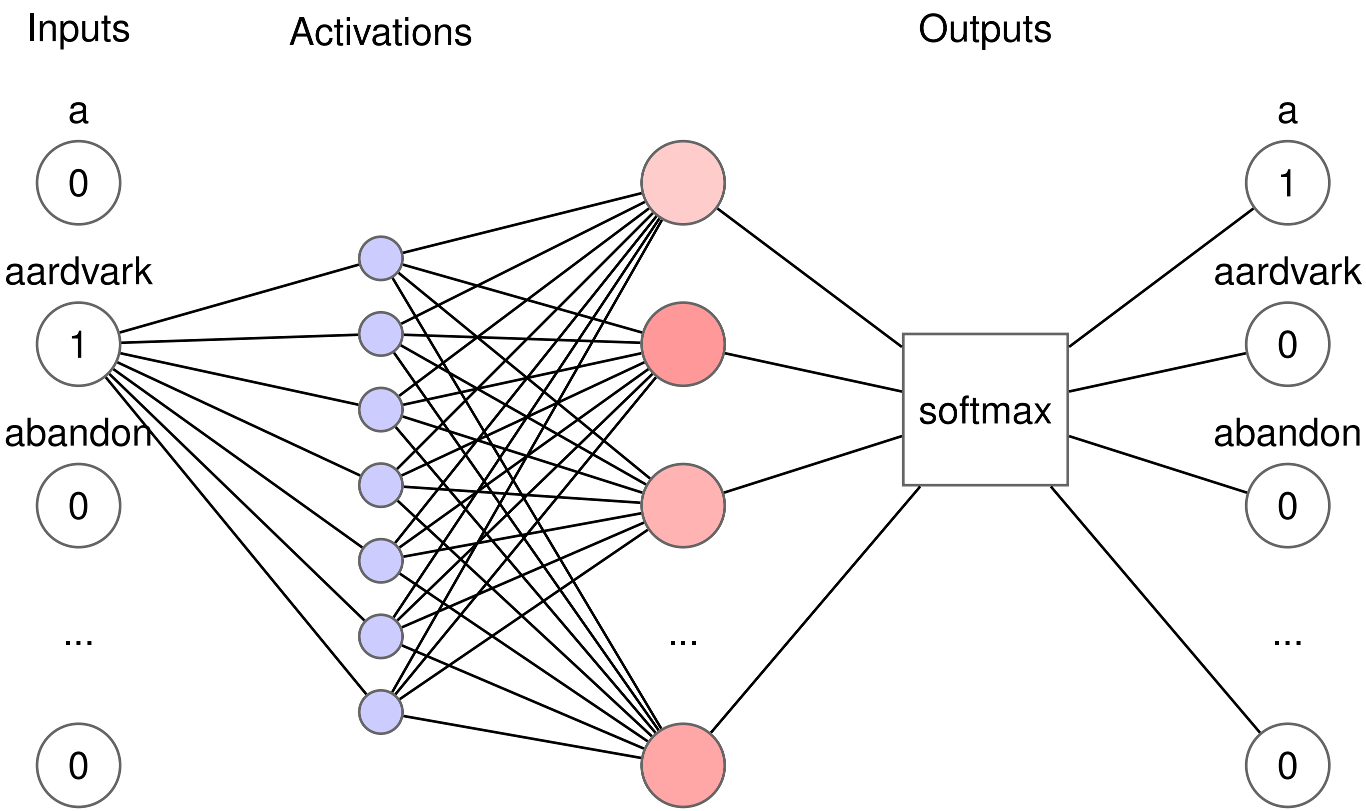
Word Embeddings

Efficiently Training word2vec



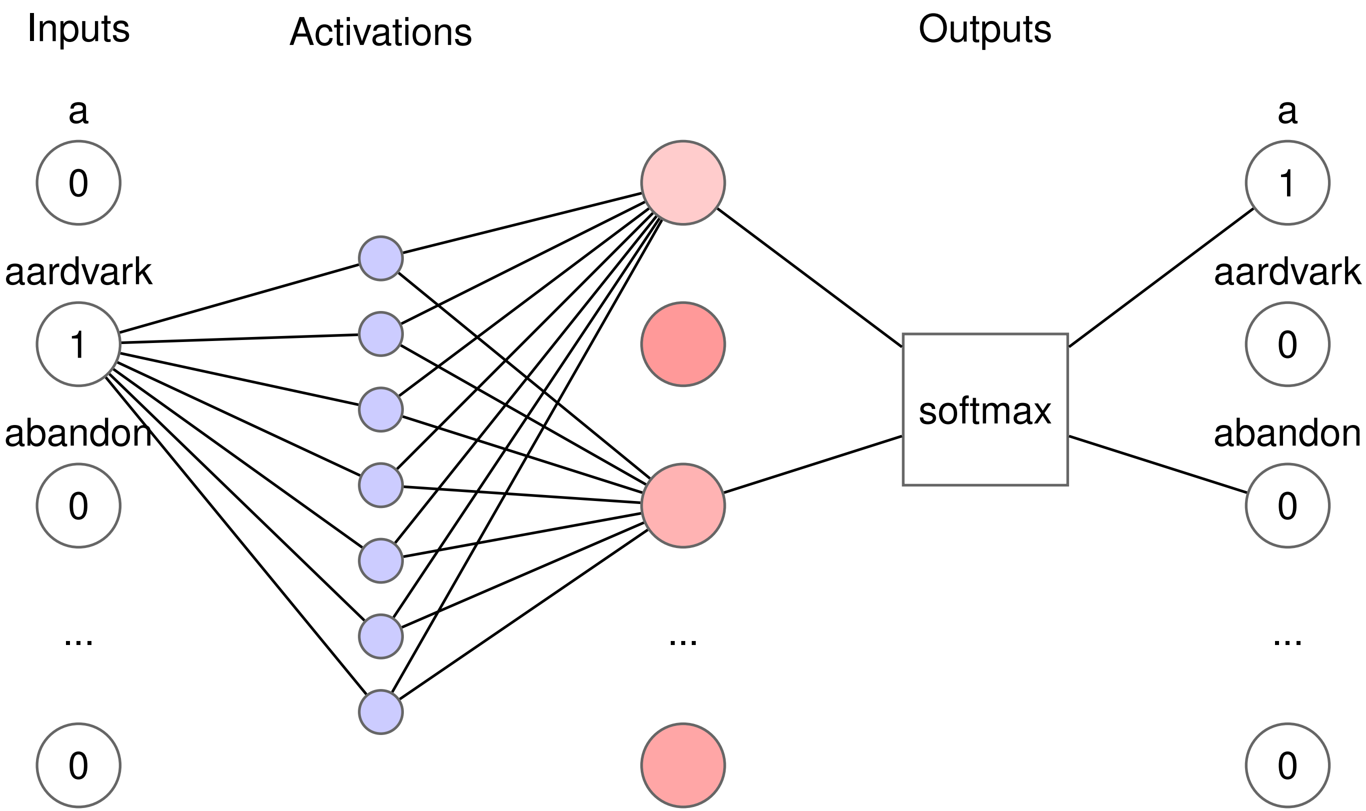
Word Embeddings

Efficiently Training word2vec



- Drop weights that do not contribute to prediction.
- Still left with over 300,000,000 weights to optimise.

Word Embeddings
Efficiently Training word2vec



- Randomly select subset of words will be ‘negative’ samples.
- `a` is still our target word, but now `abandon` is a negative sample.
- Now only need to optimise approximately 300 weights per step.