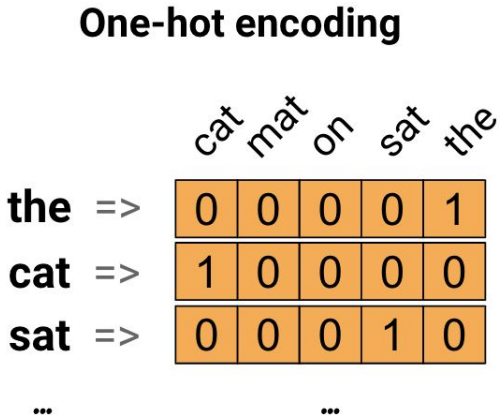
Embeddings

What are Embeddings?

- Embeddings are dense representation of categorical data
- What is “dense representation”?
- Dense is the opposite of sparse. Sparse representations are:
  - One hot encoding
  - Dummy variables
  - Tf-idf vectors
  - Bag-of-words vectors

Drawbacks of sparse representations

Consider the sentence "The cat sat on the mat":



To create a vector that contains the encoding of the sentence, you could then concatenate the one-hot vectors for each word. What are the drawbacks of such a representation? Think about it in technical and informational terms

Drawbacks of sparse representations

Problem: in web search, if user searches for “Seattle motel”, we would like to match documents containing “Seattle hotel”.

But:

- motel = **[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]**
- hotel = **[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]**

These two vectors are orthogonal. There is no natural notion of similarity for one-hot vectors!
Solution:

- Could try to rely on WordNet’s list of synonyms to get similarity?
- But it is well-known to fail badly: incompleteness, etc.
- Instead: learn to encode similarity in the vectors themselves

Drawbacks of sparse representations

Problem:

- To contact, or to aggregate (summing, averaging etc.)?

Drawbacks of sparse representations

To summarize:

- Large memory and expensive computation (long vectors)
- Significant semantic loss as order of words is not preserved
- Hard to model as the number of model parameters to train will be in the scale of input vector length which is huge
- Similar meaning words (or categories) are not bundled
- Etc.

Word Embeddings

Word embeddings - The idea

Distributional semantics: **A word's meaning is given by the words that frequently appear close-by**

- **"You shall know a word by the company it keeps" (J. R. Firth 1957: 11)**
- One of the most successful ideas of modern statistical NLP!
- The context of a word is the set of words that appear nearby
- Use the many contexts of w (word) to build up a representation of w
- E.g.:
  - …government debt problems turning into **banking** crises as happened in 2009…
  - …saying that Europe needs unified **banking** regulation to replace the hodgepodge…
  - …India has just given its **banking** system a shot in the arm… These context words will represent banking

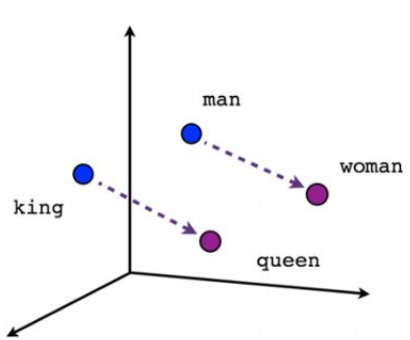
Word Embeddings - Dense Vectors

We will build a dense vector for each word, chosen so that it is similar to vectors of words that appear in similar contexts

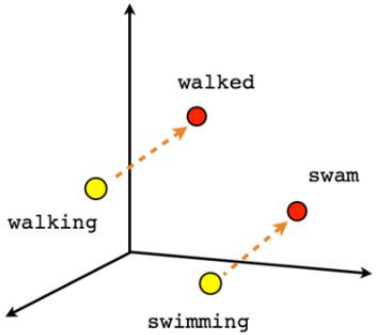
$$\textit{banking} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

Word Embeddings -Advantages

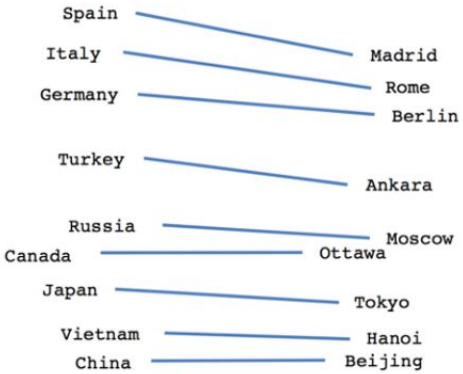
- Short - Word vectors can range from size 8 to size 1024
- Re-use - Train once, use as much as you want
- Train yourself or used somebody's else
- Captures meaning - similar words (categories) have similar, in-close-proximity vectors



Male-Female



Verb tense



Country-Capital

Open-sources embedding models

- There are two main open source embedding models:
  - Word2Vec -
    - published in 2013 by researchers from Google
    - The word2vec algorithm uses a neural network model to learn word associations from a large corpus of text
    - Algorithm: CBOW or Skipgram
    - V(queen) = V(king) − V(man) + V(woman)
  - GloVe -
    - GloVe - Global Vectors
    - Developed as an open-source project at Stanford - 2014
  - Also:
    - Fasttext
    - Transformers - Bert/Elmo/RoBerta

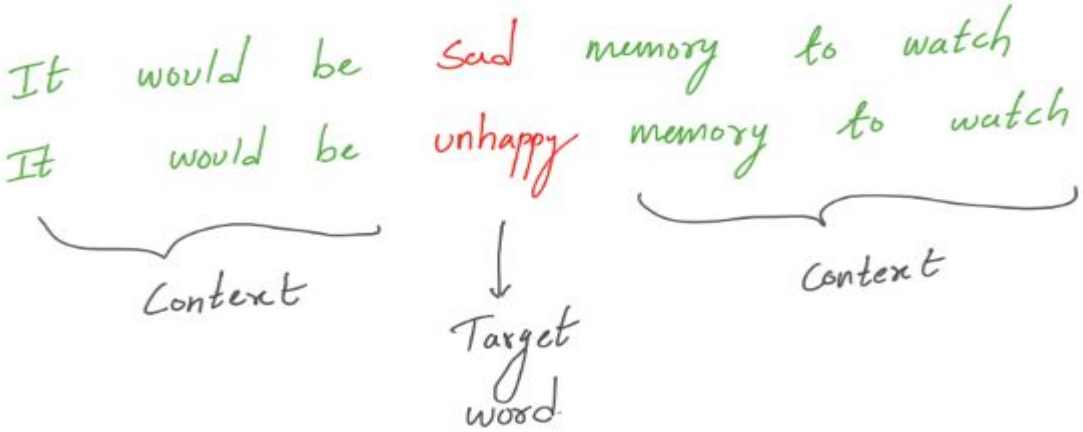
Training Embeddings

- You can train embeddings on your own dataset - embeddings are relevant to all category inputs, not only words
- Actually surprisingly simple in its most basic form.
- Train a simple neural network with a single hidden layer to perform a certain task
- But.. we’re not actually going to use that neural network for the task we trained it on!
- Instead, the goal is actually just to learn the weights of the hidden layer
- This can be achieved with any classification algorithm, not only NNs
- Open source word embeddings are trained in an unsupervised approach:
    - CBOW algorithm
    - Skipgram algorithm

Training Embeddings

CBOW:

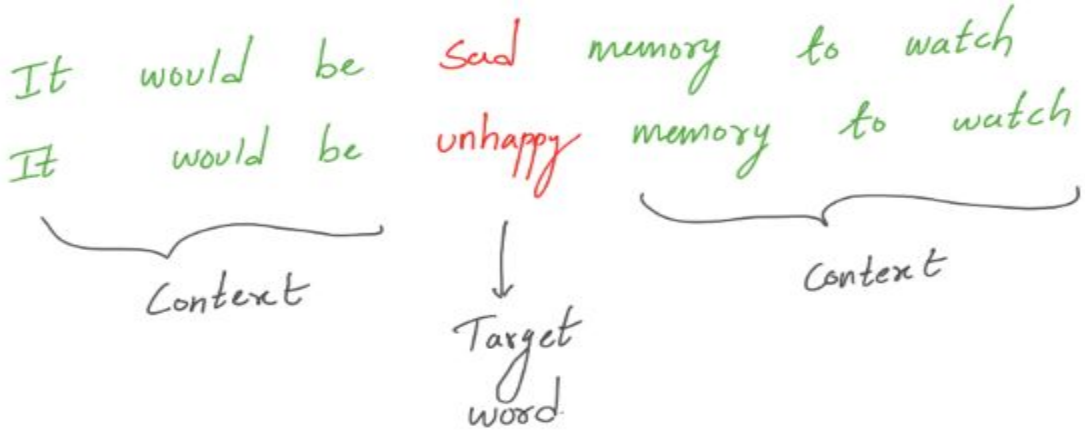
- Intuition: Word used in similar ways/context result in similar representations.
- The neighbouring words will give use the context of the target word



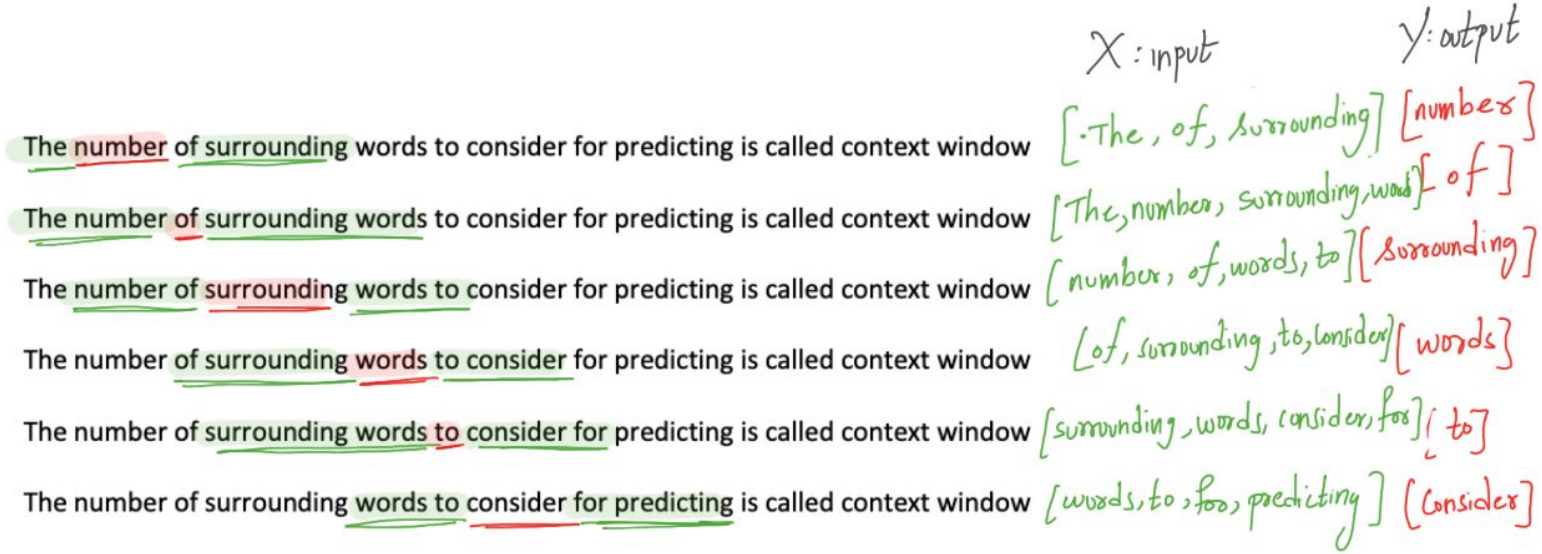
Training Embeddings

CBOW:

- The learning task: Predict the current target word (“sad”) based on the context words (surrounding words).
- The number of surrounding words to consider for predicting is called context window.
- All we need to give is huge corpus(set of all documents) nothing more than that.



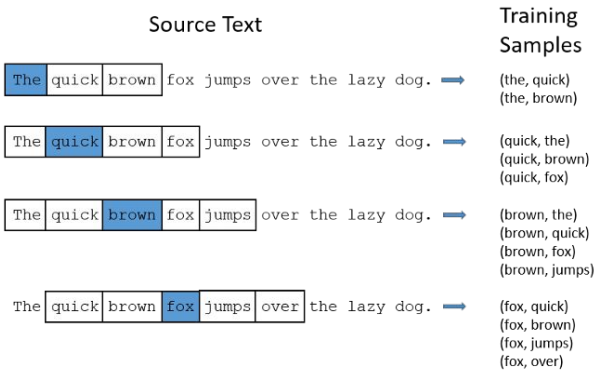
Training Embeddings



Training Embeddings - Skip gram

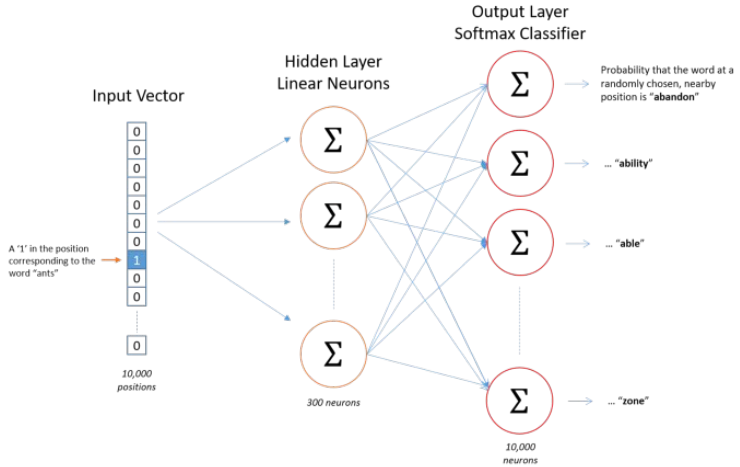
We're going to train the neural network to do the following:

- Given a specific word in the middle of a sentence (the input word), look at the words nearby and pick one at random.
- The network is going to tell us the probability for every word in our vocabulary of being the "nearby word" that we chose.
- We'll train the neural network to do this by feeding it word pairs found in our training documents.



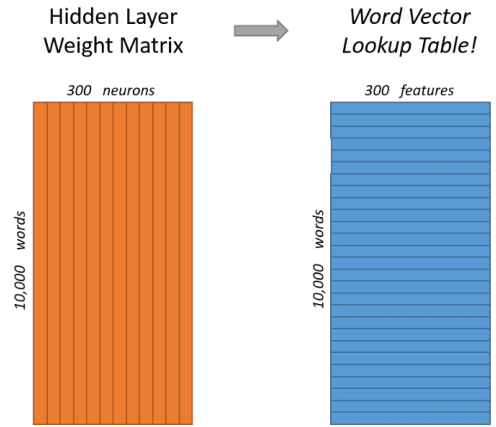
Training Embeddings - Skip gram

- Input words are simply a one-hot vectors
- Word vectors will have X (e.g 10,000) components (one for every word in our vocabulary)
- The output of the network is a single vector (also vocabulary size)
- The output neurons use softmax.



Training Embeddings - Skip gram

- For example, say we want to learn word vectors with 300 features.
- The hidden layer is going to be represented by a weight matrix with 10,000 rows (one for every word in our vocabulary) and 300 columns (one for every hidden neuron).
- The end goal of all of this is really just to learn this hidden layer weight matrix



Let’s see it in action:

- Word embeddings, Visualized
- T-SNE for Dimensionality reduction - Beginner guide