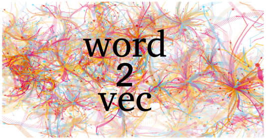
word2vec

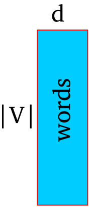
Text

WIKIPEDIA

feed in text



wait a few hours

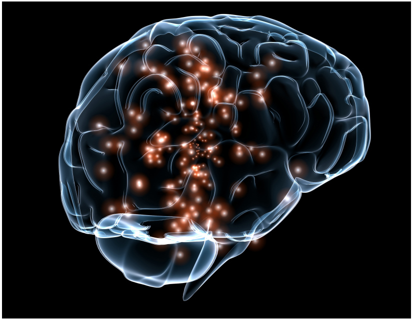


dog = (0.12,-0.32,0.92,0.43,-0.3 ...)
cat = (0.15,-0.29,0.90,0.39,-0.32 ...)

chair = (0.8,0.9,-0.76,0.29,0.52 ...)

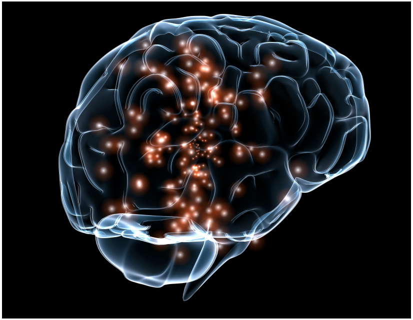
get a |V|xd matrix W where each row is a vector for a word

Seems magical.



“Neural computation, just like in the brain!”

Seems magical.



“Neural computation, just like in the brain!”

How does this actually work?

How does word2vec work?

word2vec implements several different algorithms:

Two training methods

- Negative Sampling
- Hierarchical Softmax

Two context representations

- Continuous Bag of Words (CBOW)
- Skip-grams

How does word2vec work?

word2vec implements several different algorithms:

Two training methods

- **Negative Sampling**
- Hierarchical Softmax

Two context representations

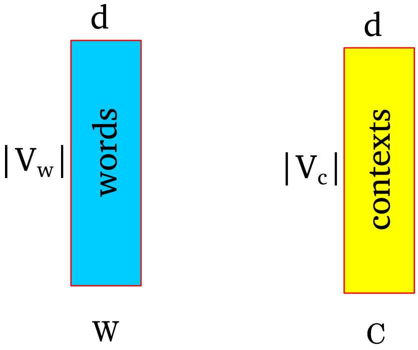
- Continuous Bag of Words (CBOW)
- **Skip-grams**

We'll focus on skip-grams with negative sampling.

intuitions apply for other models as well.

How does word2vec work?

- Represent each word as a *d* dimensional vector.
- Represent each context as a *d* dimensional vector.
- Initalize all vectors to random weights.
- Arrange vectors in two matrices, *W* and *C*.



How does word2vec work?

While more text:

- Extract a word window:

`A springer is [ a cow or` **`heifer`** `close to calving ].`

$$\begin{array}{ccccccc} c_1 & c_2 & c_3 & w & c_4 & c_5 & c_6 \end{array}$$

- w is the focus word vector (row in W).
- c_i are the context word vectors (rows in C).

How does word2vec work?

While more text:

- Extract a word window:

```
A springer is [ a   cow  or  heifer  close  to  calving ].
```

$$\begin{array}{ccccccc} c_1 & c_2 & c_3 & w & c_4 & c_5 & c_6 \end{array}$$

- Try setting the vector values such that:

$$\sigma(w \cdot c_1) + \sigma(w \cdot c_2) + \sigma(w \cdot c_3) + \sigma(w \cdot c_4) + \sigma(w \cdot c_5) + \sigma(w \cdot c_6)$$

is **high**

How does word2vec work?

While more text:

- Extract a word window:

$$\begin{array}{cccccccccccc} \texttt{A springer is [} & \texttt{a} & \texttt{cow} & \texttt{or} & \textbf{heifer} & \texttt{close} & \texttt{to} & \texttt{calving} & \texttt{].} \\ & c_1 & c_2 & c_3 & w & c_4 & c_5 & c_6 & \end{array}$$

- Try setting the vector values such that:

  $$\sigma(w \cdot c_1) + \sigma(w \cdot c_2) + \sigma(w \cdot c_3) + \sigma(w \cdot c_4) + \sigma(w \cdot c_5) + \sigma(w \cdot c_6)$$

  is **high**

- Create a corrupt example by choosing a random word w'

$$\begin{array}{ccccccccc} \texttt{[} & \texttt{a} & \texttt{cow} & \texttt{or} & \textbf{comet} & \texttt{close} & \texttt{to} & \texttt{calving} & \texttt{]} \\ & c_1 & c_2 & c_3 & w' & c_4 & c_5 & c_6 & \end{array}$$

- Try setting the vector values such that:

  $$\sigma(w' \cdot c_1) + \sigma(w' \cdot c_2) + \sigma(w' \cdot c_3) + \sigma(w' \cdot c_4) + \sigma(w' \cdot c_5) + \sigma(w' \cdot c_6)$$

  is **low**

How does word2vec work?

The training procedure results in:

- $w \cdot c$ for **good** word-context pairs is **high**.
- $w \cdot c$ for **bad** word-context pairs is **low**.
- $w \cdot c$ for **ok-ish** word-context pairs is **neither high nor low**.

As a result:

- Words that share many contexts get close to each other.
- Contexts that share many words get close to each other.

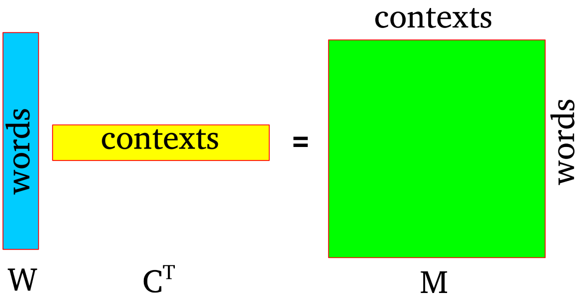
At the end, word2vec throws away C and returns W.

Reinterpretation

Imagine we didn't throw away C. Consider the product $WC^\top$

Reinterpretation

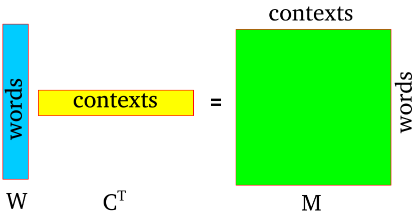
Imagine we didn't throw away C. Consider the product $WC^\top$



The result is a matrix M in which:

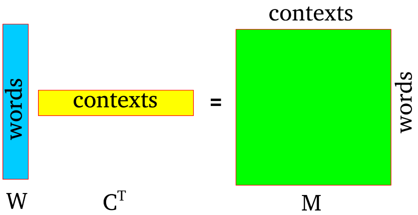
- Each row corresponds to a word.
- Each column corresponds to a context.
- Each cell correspond to $w \cdot c$, an association measure between a word and a context.

Reinterpretation



Does this remind you of something?

Reinterpretation



Does this remind you of something?

Very similar to SVD over distributional representation: