Word2Vec

Tim Metzler

Department of Computer Science

HBRS / ST.A.

What have we seen so far?

1) One Hot Encoding:
   - Representation of words as binary vectors
   - Sparse
   - Lack of semantic information
   - Size: |V|

2) TFIDF Embedding
   - Term Frequency-Inverse Document Frequency
   - Reflects importance of word in document relative to document collection
   - Weighs down frequent terms, scales up rare ones
   - Still lacks contextual understanding
   - Size: |D| (number of documents)

Recap: Similarity and Relatedness

- **Similarity:** Likeliness between two words in meaning or context. (Can I replace the word in the sentence with the other word?)
- **Relatedness:** How strongly are two words associated. (Are the two words likely to appear together?)

Recap: Similarity and Relatedness

- **Similarity:** Likeliness between two words in meaning or context. (Can I replace the word in the sentence with the other word?)
- **Relatedness:** How strongly are two words associated. (Are the two words likely to appear together?)

”I like to eat pizza. I like to eat stew”
→ pizza and stew are similar in this context
”Pizza is baked”
→ pizza and baked are related since they often appear together

There is no clear distinction between these two!

Distributional Structure of Language

Consider the following sentence:

Berlin is the capital of Germany.

What does this tell us about Berlin?

Obvious: Berlin is the capital of Germany.

Distributional Structure of Language

Consider the following sentence:

Walala is the capital of Lampukistan.

What does this tell us about Walala?

Distributional Structure of Language

Consider the following sentence:

Walala is the capital of Lampukistan.

What does this tell us about Walala?

→ Walala is similar to words that appear in the context of "*is the capital of*".

→ Walala is a city.

Distributional Structure of Language

Words that appear in a similar context are similar.

How do we learn embeddings that capture these semantic relationships?

Distributional Structure of Language

Words that appear in a similar context are similar.

How do we learn embeddings that capture these semantic relationships?

Requirements:

- Fixed vector size
- Similar words should have similar representations in the vector space
- General vectors, not optimized for a specific domain
- Easy to learn
- Can learn from vast amounts of data (e.g. Wikipedia, Common Crawl, etc)

Distributional Structure of Language

We need a learning task that will produce these vectors.

Idea: Train a simple classifier to predict word from context or predict context from word.

Word2Vec

Developed by Mikolov et al, Google (2013)
“Efficient Estimation of Word Representations in Vector Space”
(https://arxiv.org/abs/1301.3781)

“Distributed Representations of Words and Phrases and their Compositionality” in Advances in Neural Information Processing Systems
(doi:10.48550/arXiv.1310.4546)

Idea: Train a simple classifier to predict word from context or predict context from word.

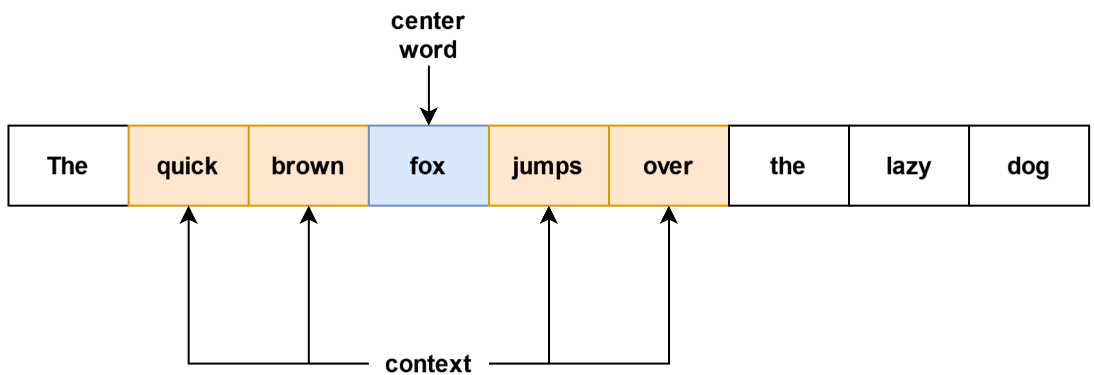
Word2Vec

Example Sentence:

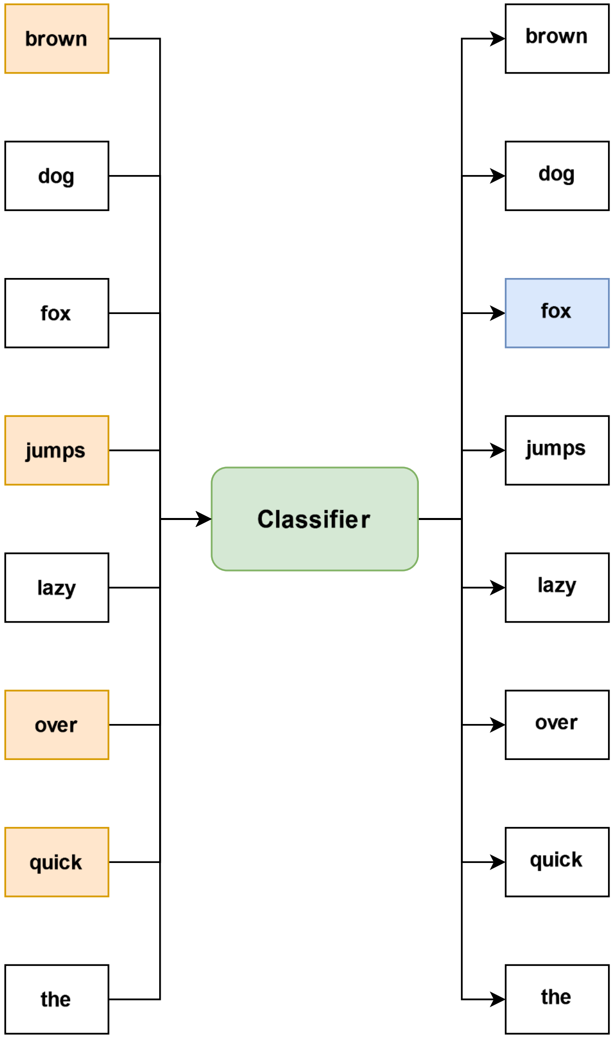
The	**quick**	**brown**	**fox**	**jumps**	**over**	**the**	**lazy**	**dog**

Word2Vec

Example Sentence:



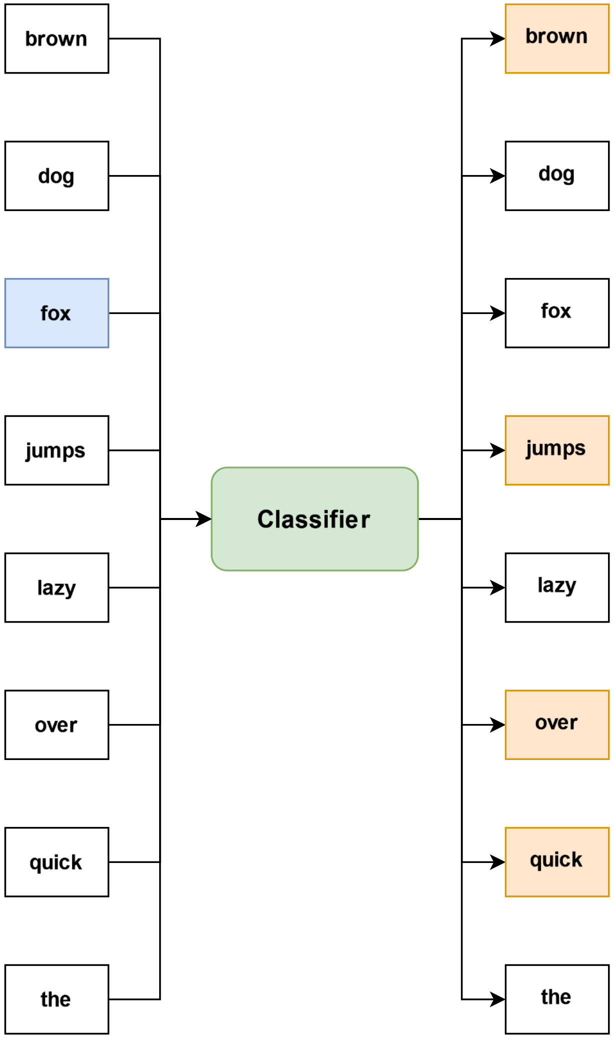
Word2Vec



Continuous Bag-Of-Word Model

Predict center word from context

Word2Vec



Skip Gram Model

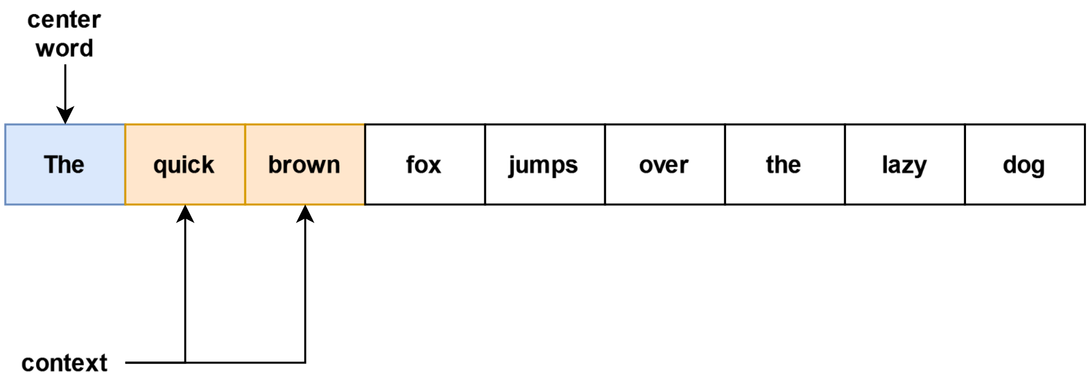
Predict context from center word.

We will use this approach for our examples!

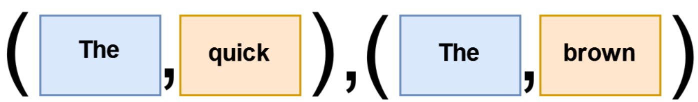
Word2Vec – Creating Training Examples

Create training examples from each sentence in the corpus.
Training examples are of the form (word1, word2).
Our context never crosses sentence boundaries!

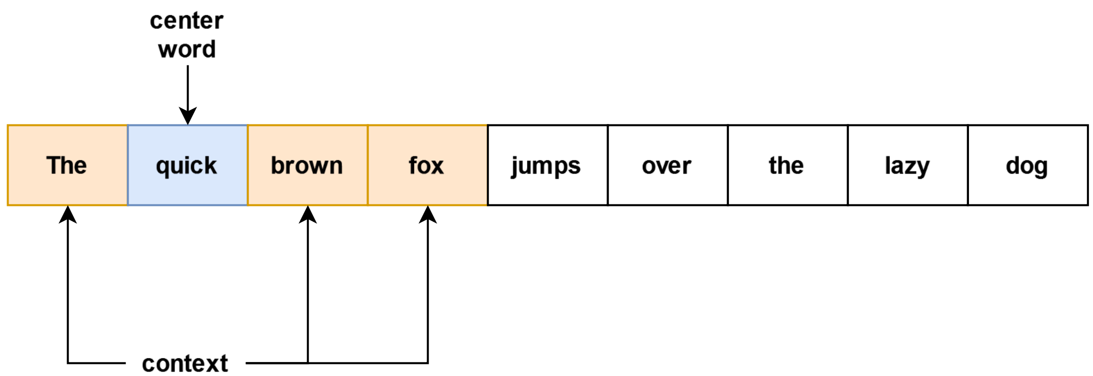
Word2Vec – Creating Training Examples



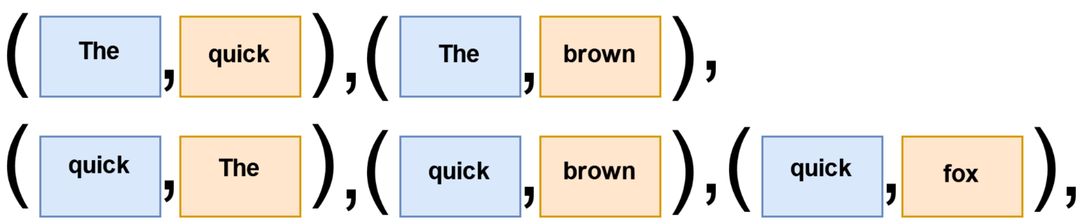
Training Examples:



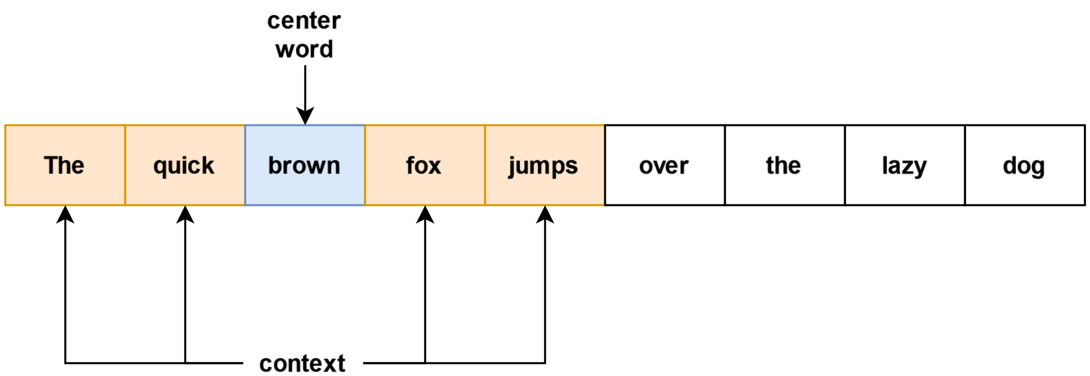
Word2Vec – Creating Training Examples



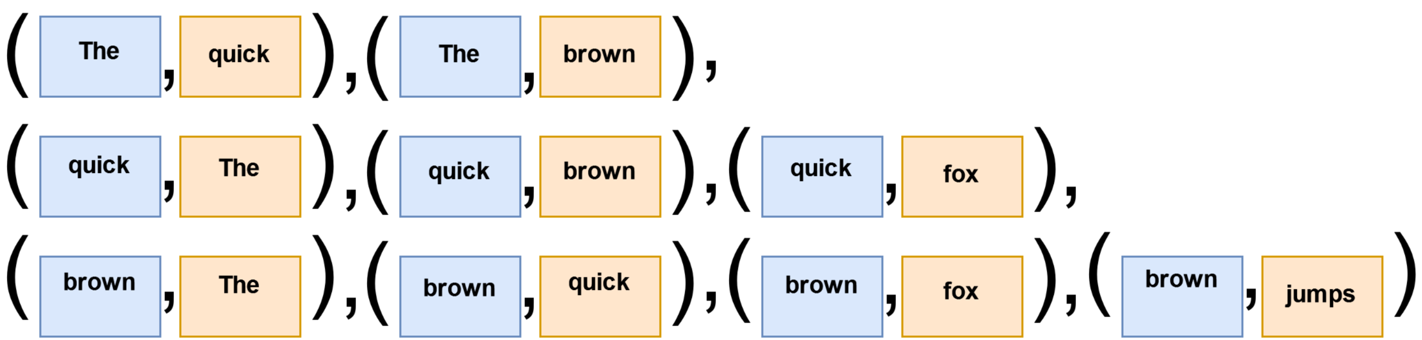
Training Examples:



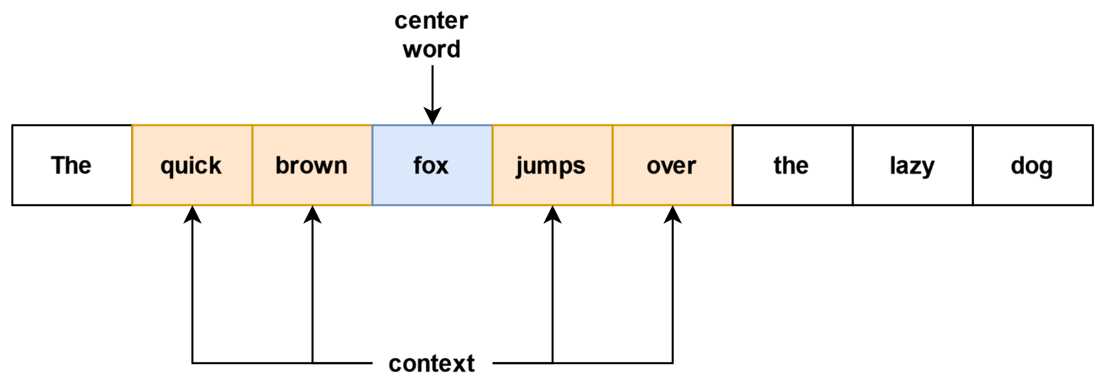
Word2Vec – Creating Training Examples



Training Examples:



Word2Vec – Creating Training Examples



Training Examples:

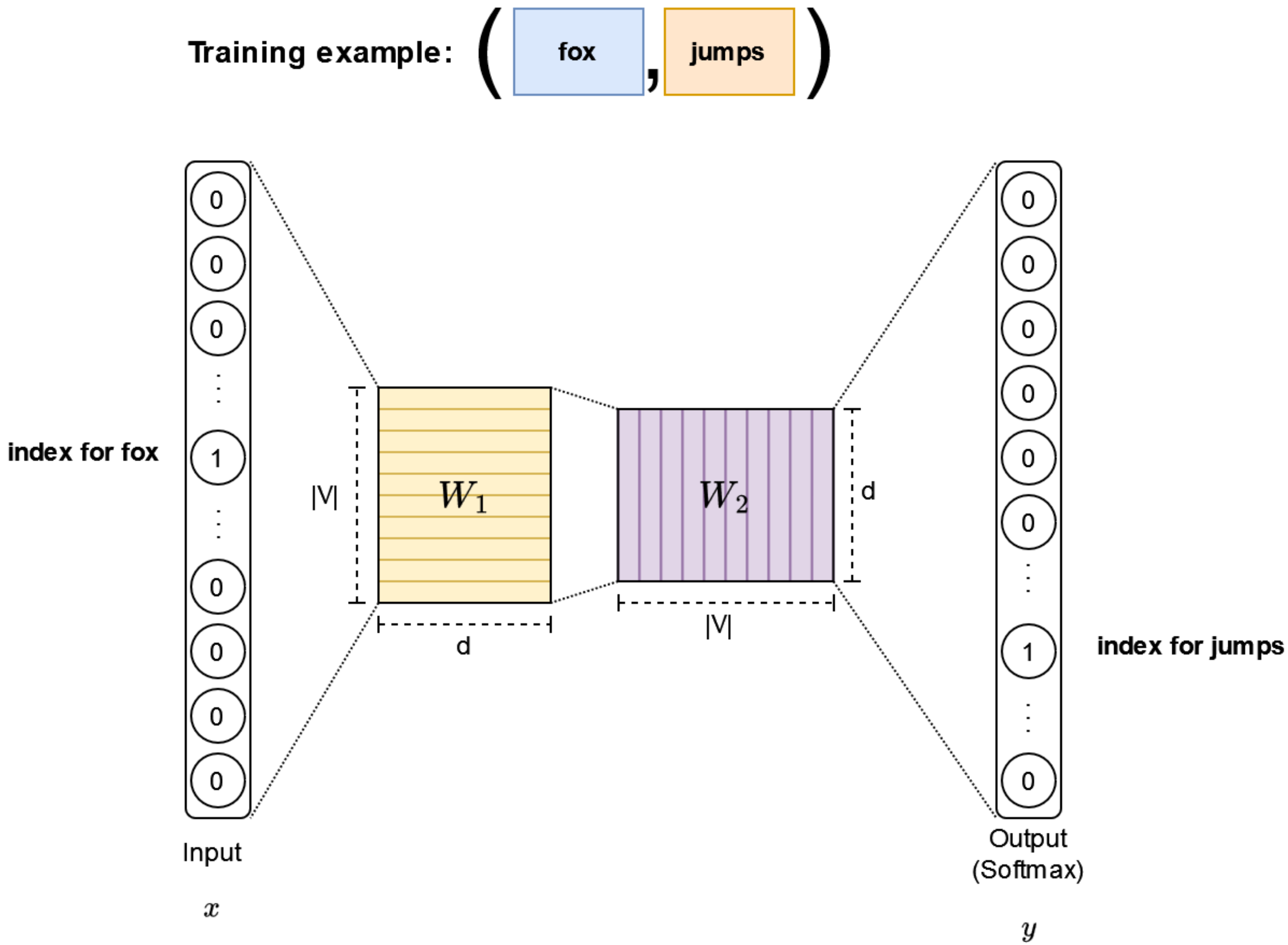
(The, quick), (The, brown),

(quick, The), (quick, brown), (quick, fox),

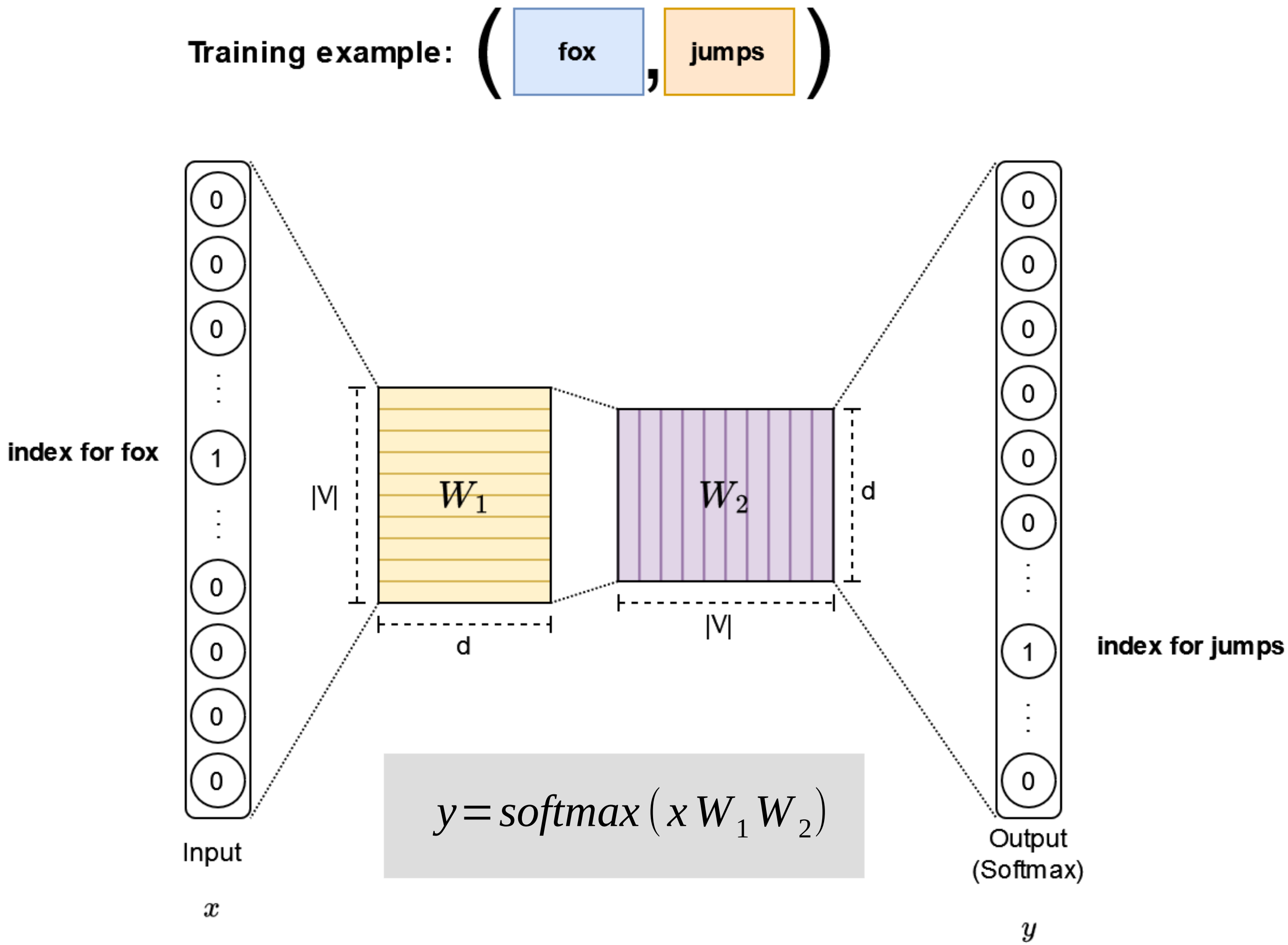
(brown, The), (brown, quick), (brown, fox), (brown, jumps),

(fox, quick), (fox, brown), (fox, jumps), (fox, over),

Word2Vec – Skip Gram Architecture

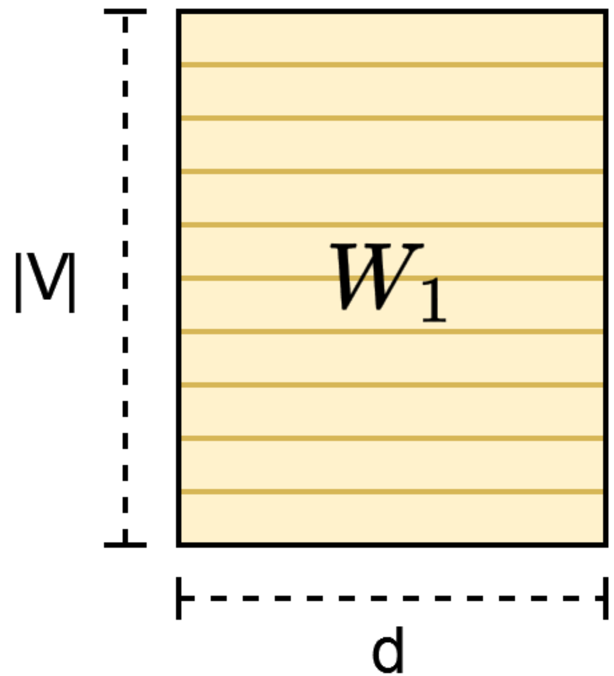


Word2Vec – Skip Gram Architecture

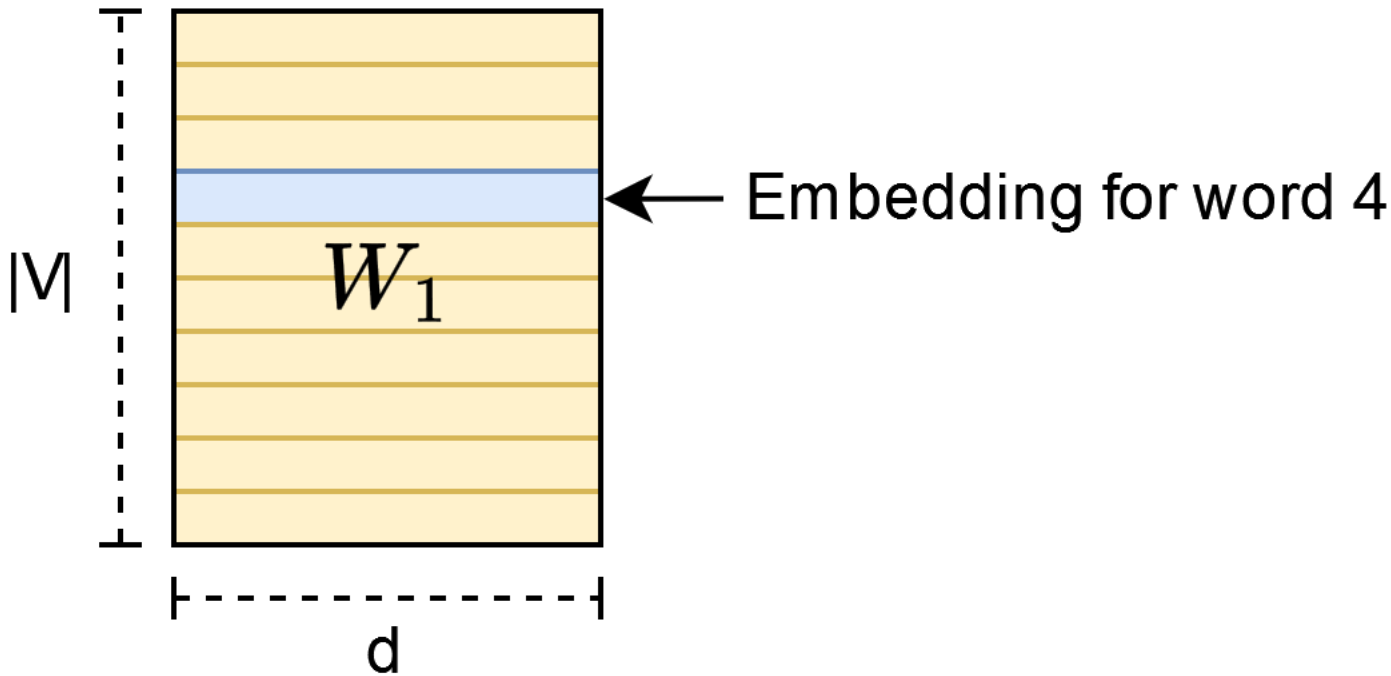


$$y = softmax\left(x W_1 W_2\right)$$

Word2Vec – Extracting the Embeddings



Word2Vec – Extracting the Embeddings



Word2Vec – Skip Gram Architecture

$$y = softmax(x W_1 W_2)$$

Recap Softmax:

Softmax turns vector into probability distribution s.t. it sums to 1.

$$softmax(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Word2Vec – Skip Gram Architecture

$$y = softmax(x W_1 W_2)$$

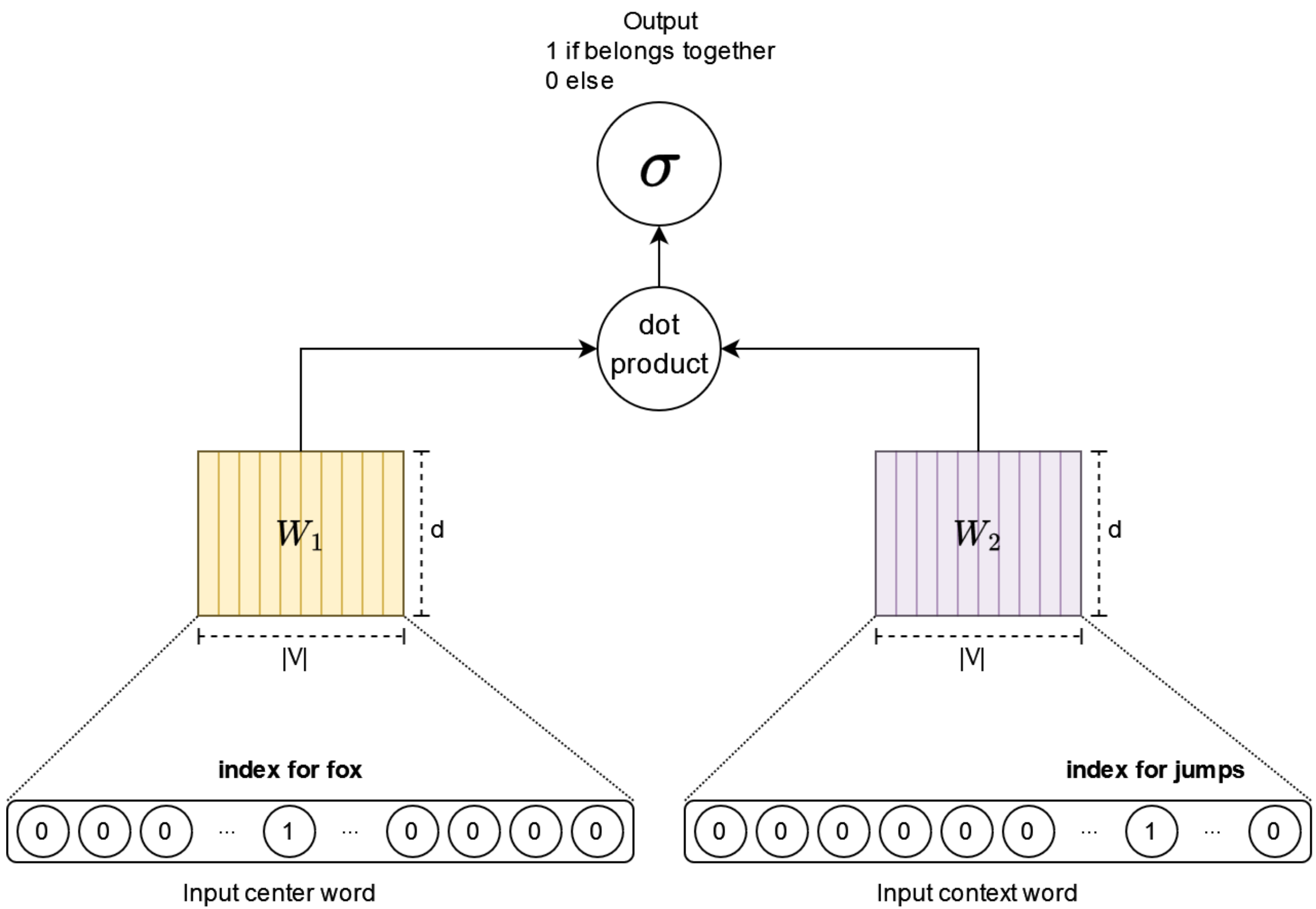
Recap Softmax:

Softmax turns vector into probability distribution s.t. it sums to 1.

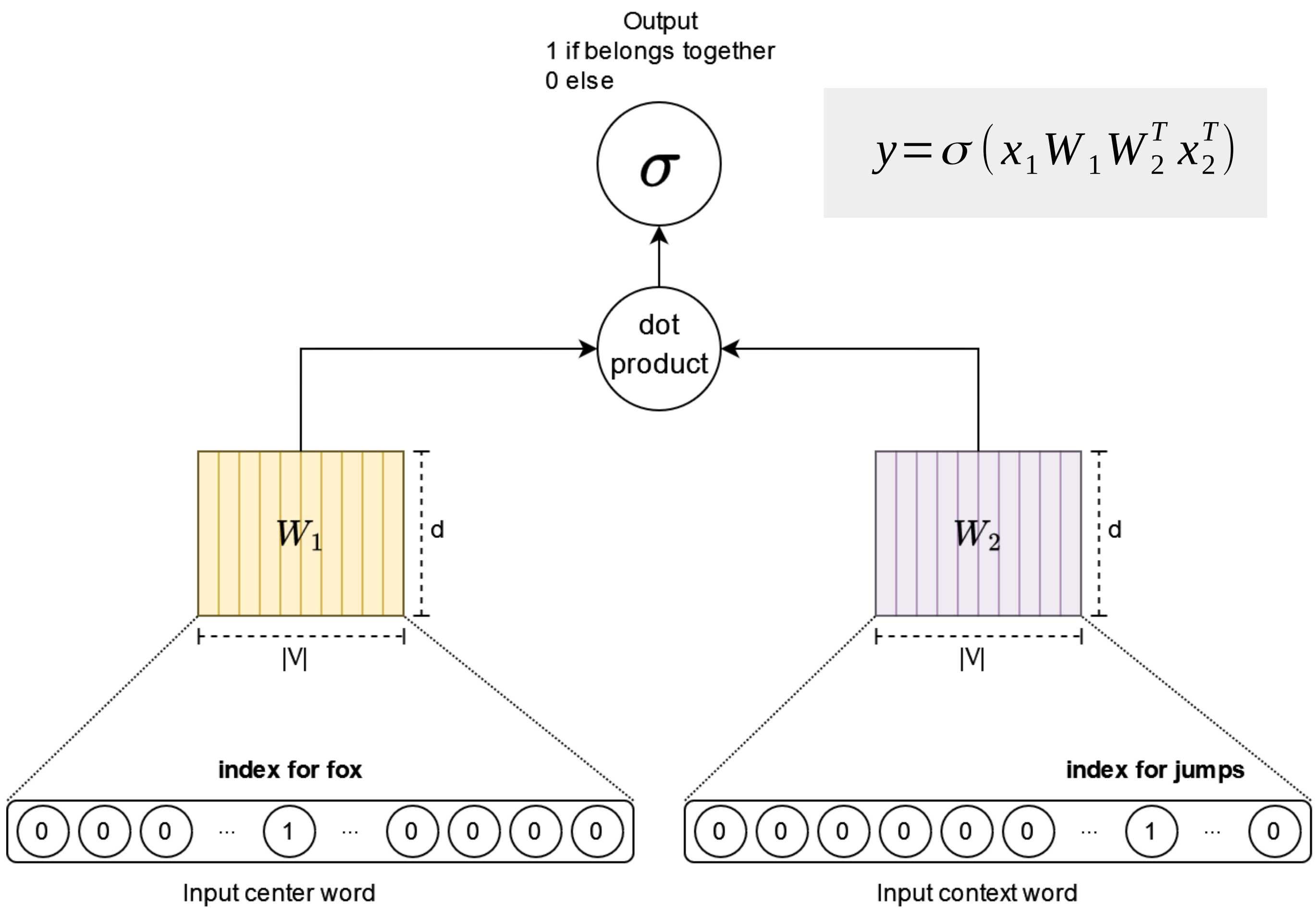
$$softmax(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Problem:
Vector z has vocabulary size (10,000; 100,000; 1,000,000?)
→ Computationally expensive!!!

Word2Vec - Negative Sampling Skip Gram Model



Word2Vec - Negative Sampling Skip Gram Model



$$y = \sigma\left(x_1 W_1 W_2^T x_2^T\right)$$

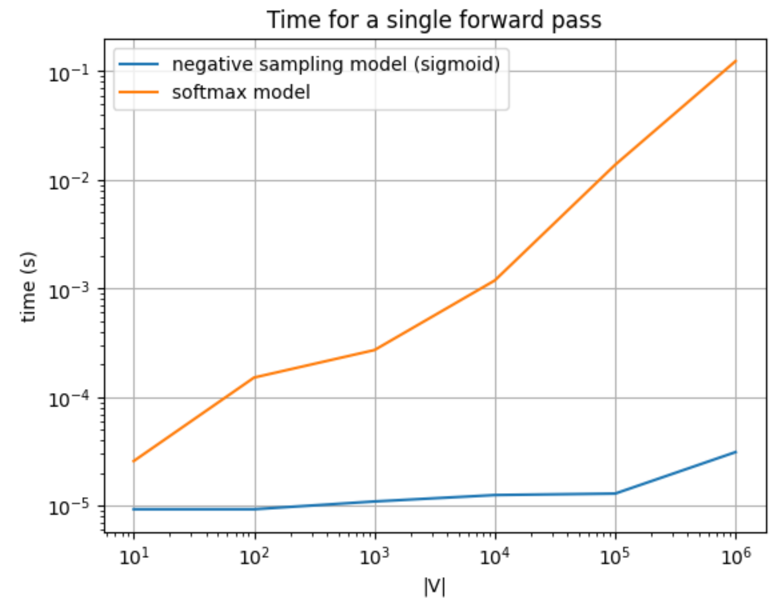
Word2Vec - Negative Sampling Skip Gram Model

Previous approach vs this approach:

Time for one forward pass in a notebook:
Previous:
~150 ms for vocabulary size of 1,000,000!

This approach:
~30 µs for vocabulary size of 1,000,000!

~5000 times faster!



Word2Vec - Negative Sampling Skip Gram Model

This is not the full truth!

So far we have only created positive examples where:
(word1, word2) → 1

Without negative examples our classifier could always predict 1
and achieve a 100% accuracy.

Word2Vec - Negative Sampling Skip Gram Model

Loss function for a batch of N examples (p is output, t is target label):

$$L = -\frac{1}{N}\left[\sum_{j=1}^{N}\left[t_j \log(p_j) + (1-t_j)\log(1-p_j)\right]\right]$$

Single positive example:

$$L = -\log(p_j)$$

Single negative example:

$$L = -\log(1-p_j)$$

Word2Vec - Negative Sampling Skip Gram Model

How do we create negative examples?

For one positive example, do we need to create (|V| - 1) negative examples?

Word2Vec - Negative Sampling Skip Gram Model

How do we create negative examples?

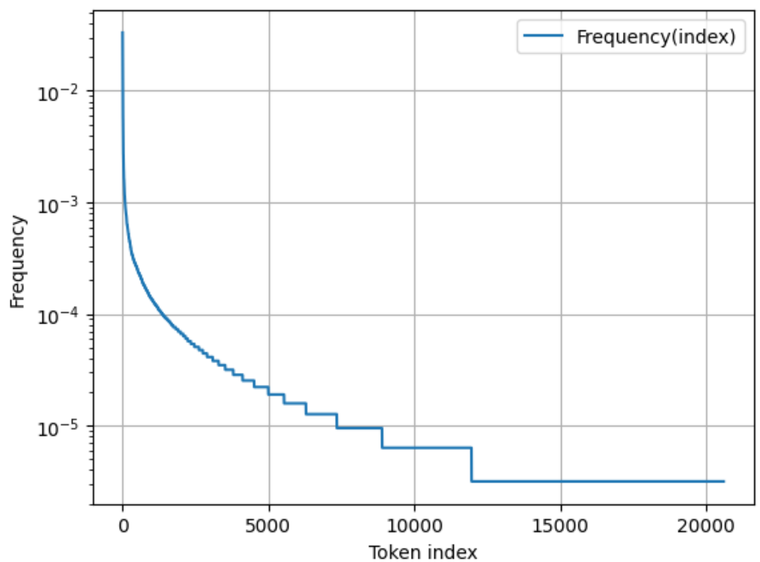
For one positive example, do we need to create (|V| - 1) negative examples?

No! Experiments show 5-20 negative examples per positive example are enough.

Word2Vec - Negative Sampling Skip Gram Model

How do we create negative examples?

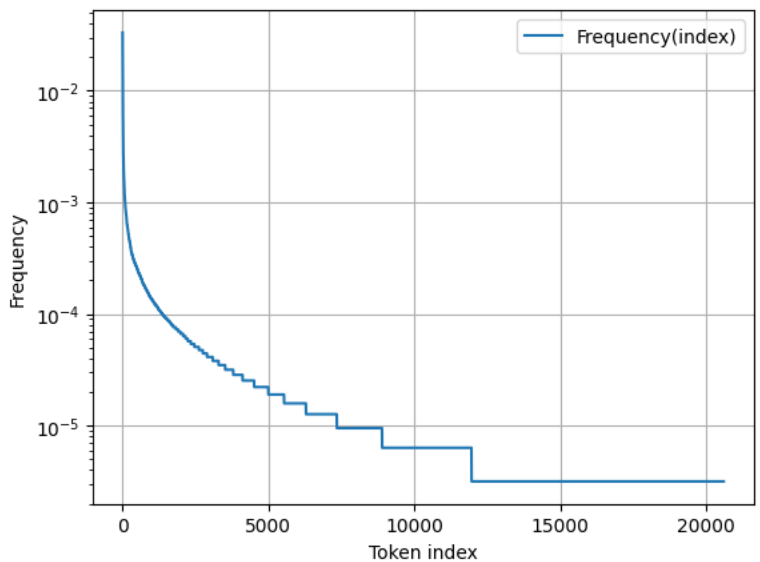
We sample according to frequency of word.

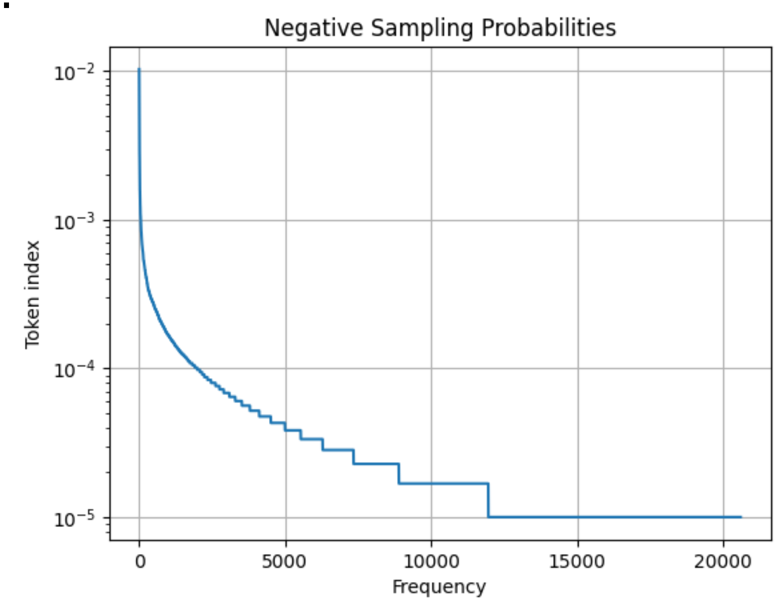


Word2Vec - Negative Sampling Skip Gram Model

How do we create negative examples?

We sample according to frequency of word.



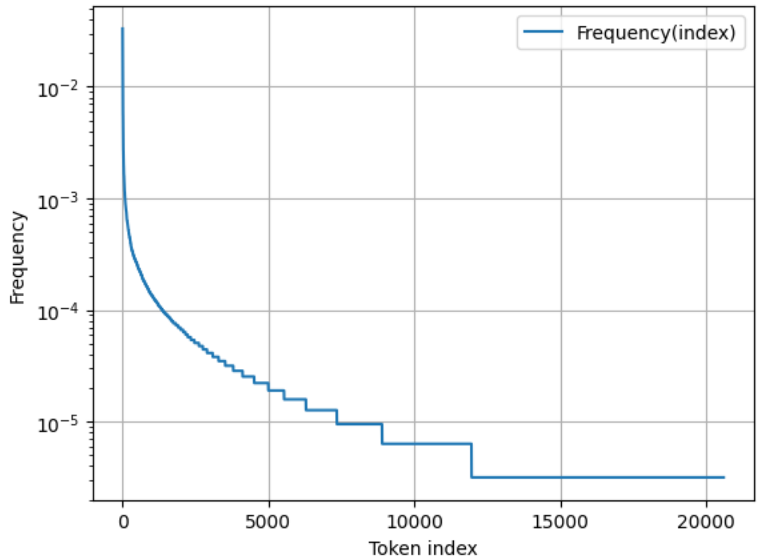


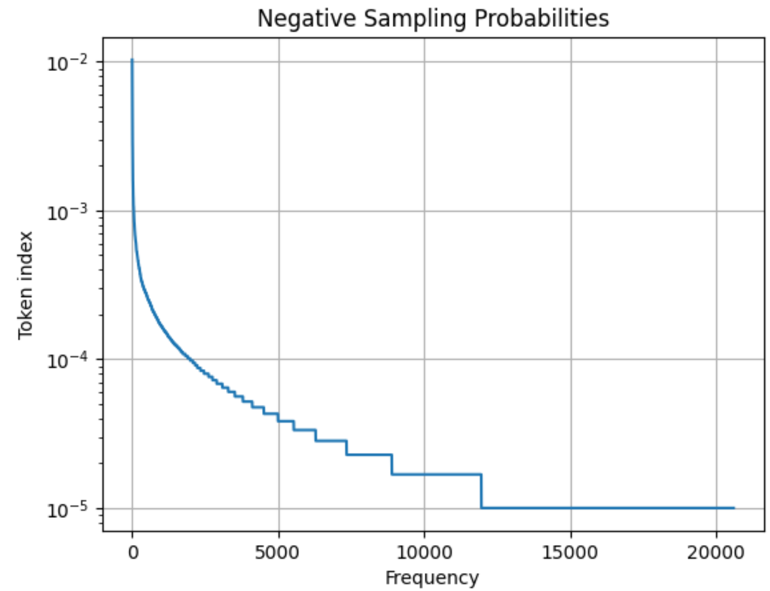
Word2Vec - Negative Sampling Skip Gram Model

How do we create negative examples?

We sample according to frequency of word.

$$f^{\frac{3}{4}}$$





Word2Vec - Negative Sampling Skip Gram Model

Important hyperparameters:

- Vector dimensionality (commonly 50-300)
- Window size (how large is our context. Often given as size of window in one direction, commonly 2-10)
- Min count (how often a word needs to appear in our corpus in order to be used for training, often 5)
- Batch size (commonly 4-64)
- Number of epochs (commonly 5-20)
- Number of negative samples (commonly 5-20)
- CBOW or Skip-Gram (commonly skip-gram)