Word2Vec Implementation – Presentation

by Sajid Ahmed

What is word2vec?

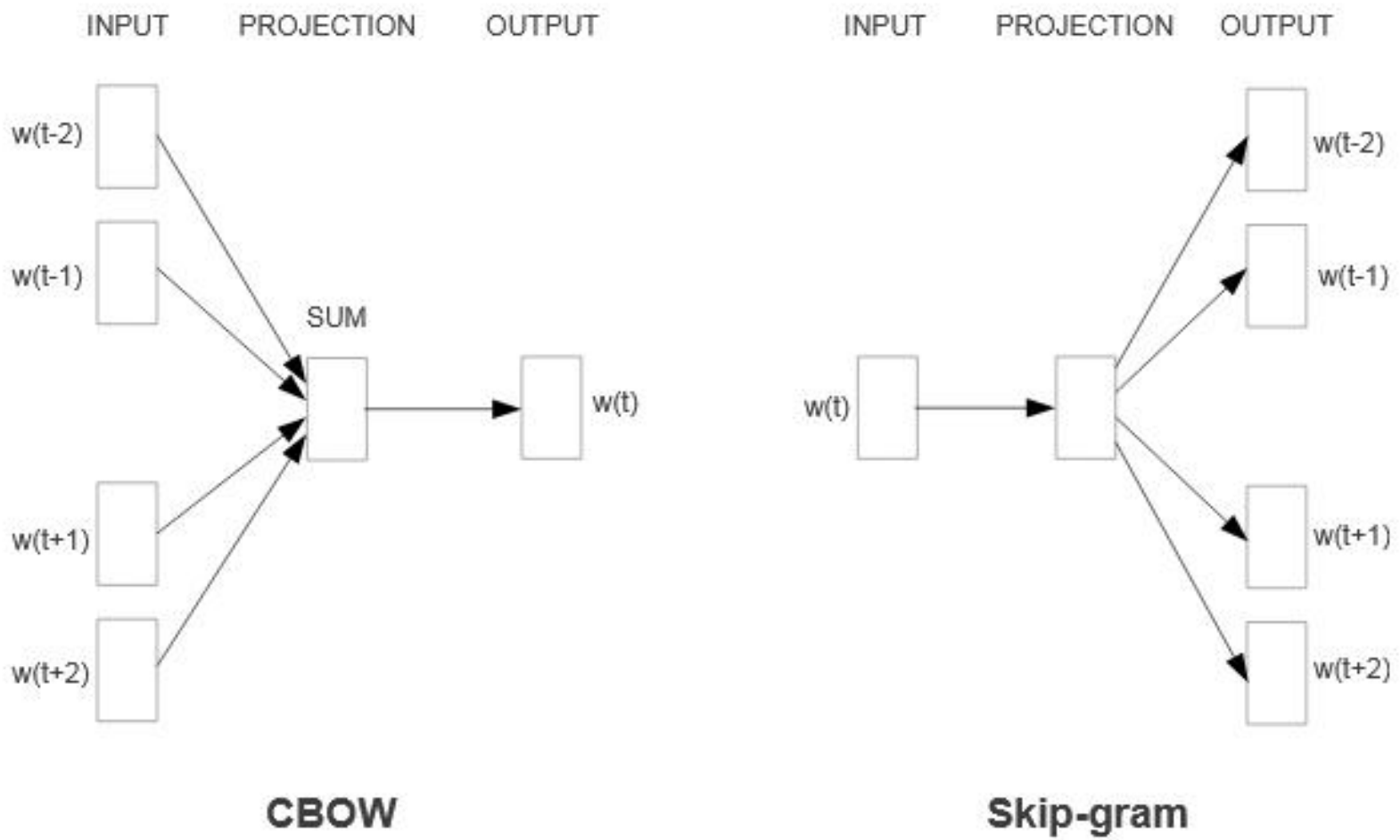
Word2vec is an approach to create word embeddings.
Word embedding is a representation of a word as a numeric vector.
Except for word2vec there exist other methods to create word embeddings, such as fastText, GloVe, ELMO, BERT, GPT-2, etc.

There are two word2vec architectures proposed in the paper:

CBOW (Continuous Bag-of-Words) : a model that predicts a current word based on its context words.
Skip-Gram: a model that predicts context words based on the current word.

Word2Vec Model Architecture

Word2vec is based on the idea that a word's meaning is defined by its context. Context is represented as surrounding words.

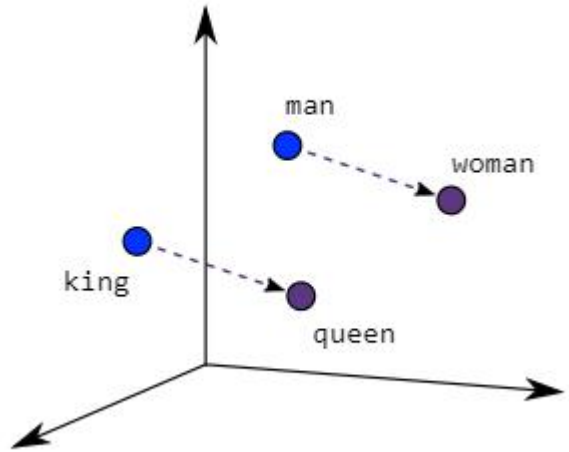


CBOW & **Skip** Gram Model

The Model consists of two models:

Continuous Bag-of-Words model (CBOW)

- In this approach, the model uses **context** words to **predict** the
- The input may be a group of words or a single word.

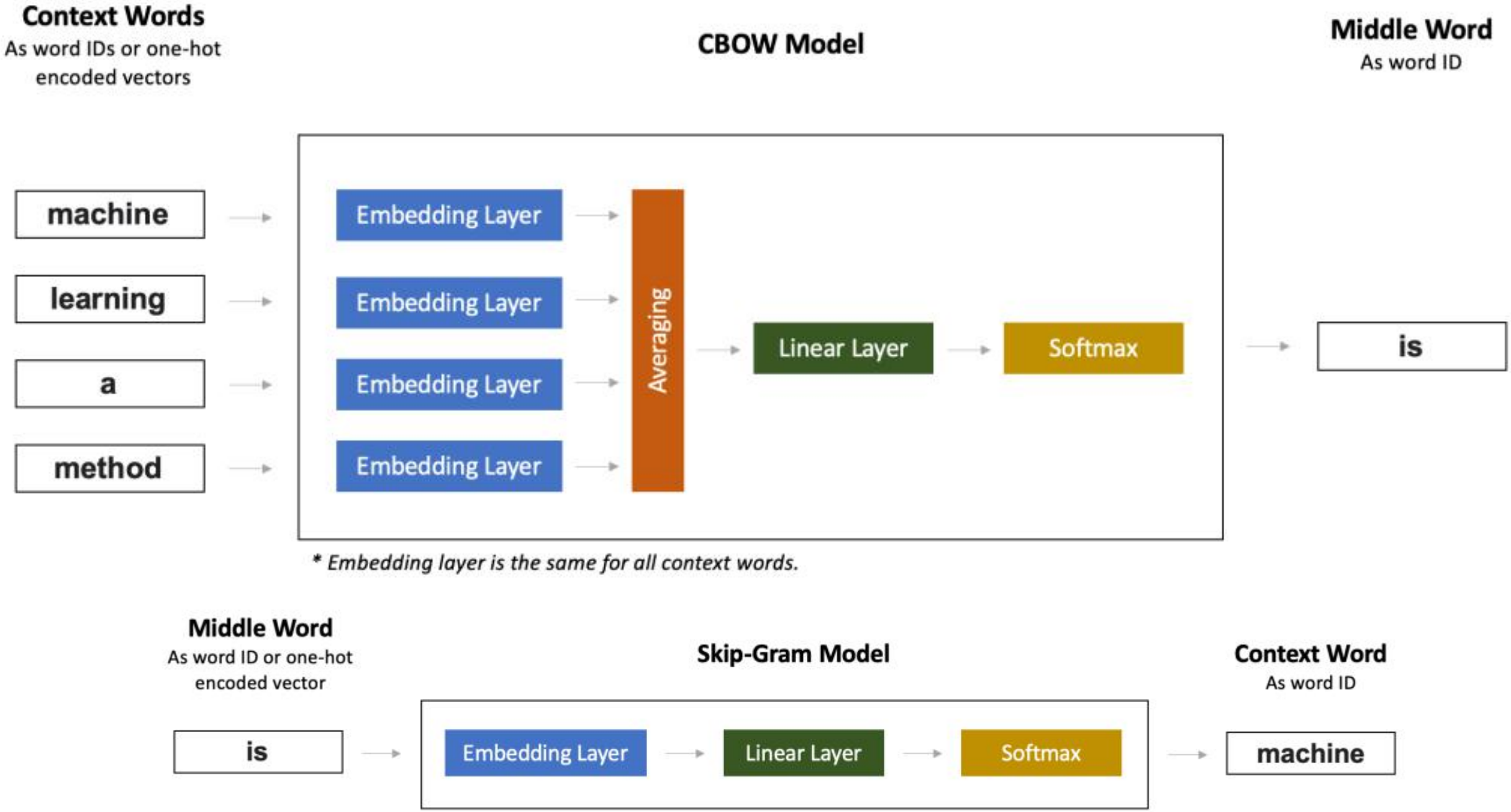


Skip-gram model

- In this approach, the model uses the **target** words to predict the **context** words.
- involves training a neural network to learn the weights of the hidden layer
- Given a specific word as its input, the model's goal is to look into the dictionary and pick a word whose **context is closely related to the target word.**

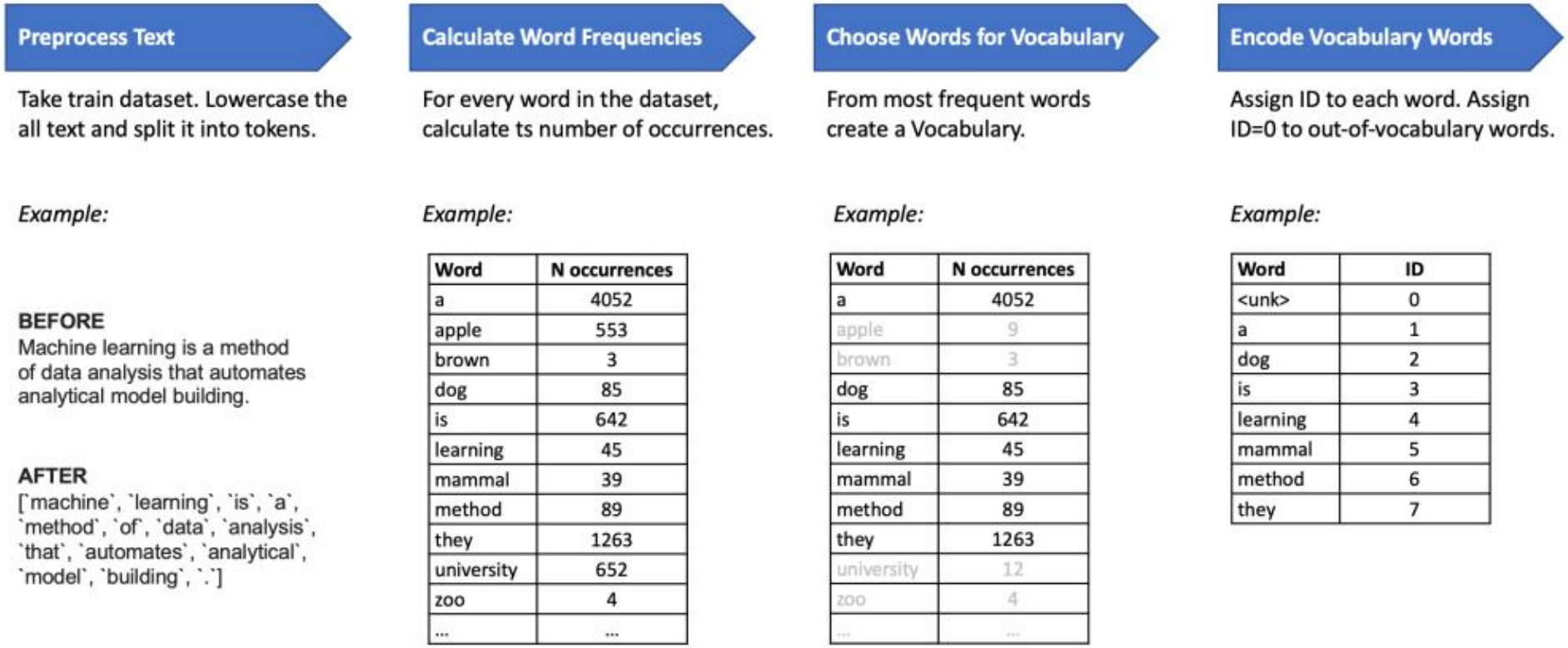
Model Architecture and Training

- The initial step would be to encode all **words with their IDs**. ID is an integer (index) that identifies word position in the vocabulary
- Then it passed the word Ids to the **Embedding layer**, which takes word ID and returns its 300-dimensional vector. Word2vec embeddings are 300-dimensional, as authors proved this number to be the best in terms of embedding quality and computational costs.
- embedding layer as a simple lookup table with learnable weights, or as a linear layer without bias and activation.
- Then the **Linear (Dense) layer** with a SoftMax activation.
- A model for a multi-class classification task, where the number of classes is equal to the number of words in the vocabulary.
- The difference between **CBOW** and **Skip-Gram** model is in the number of input words. CBOW model takes several words, each goes through the same Embedding layer, and then word embedding vectors are averaged before going into the Linear layer
- The Skip-Gram model takes a single word instead


Context Words
As word IDs or one-hot encoded vectors
CBOW Model
Middle Word
As word ID
machine
learning
a
method
Embedding Layer
Embedding Layer
Embedding Layer
Embedding Layer
Averaging
Linear Layer
Softmax
is
* Embedding layer is the same for all context words.
Middle Word
As word ID or one-hot encoded vector
Skip-Gram Model
Context Word
As word ID
is
Embedding Layer
Linear Layer
Softmax
machine

Data Preparation

- The main step in data preparation is to create a **vocabulary** by filtering out rare words, that occurred less than 5 times in the corpus;



- vocabulary created only from the words that appeared at least countable times within a text.
- word_counts = Counter(index_words)
- Used basic_english tokenizer from PyTorch that lowercases text, splits it into tokens by whitespace, but putting punctuation into separate tokens.

Skip Gram Training

- **Subsampling**
- To make optimization more efficient, the researchers subsampled the dataset so that words with high occurrence can be ignored while training. To be specific, they calculated the *subsampling probability* to be

$$P(w_i) = \left(\sqrt{\frac{z(w_i)}{t}} + 1\right) \times \frac{t}{z(w_i)}$$

- With the skipgram architecture, for each word in the text, we want to define a surrou nding _*context*_ and grab all the words in a window around that word, with size C.
- Since the more distant words are usually less related to the current word than those close to it, we give less weight to the distant words by sampling less from those word s in our training examples.
- If we choose C = 5, for each training word we will select randomly a number R in ran ge [1: C], and then use R words from previous words and R words from the future w ords and make the Word as correct labels.

Negative Sampling

- For every example we give the network, we train it using the output from the softmax layer. That means for each input, we're making very small changes to millions of weights even though we only have one true examples. This makes training the network very inefficient.
- We can approximate the loss from the softmax layer by only updating a small subset of all the weights at once. We'll update the weights for the correct example, but only a small number of incorrect, or noise, examples. This is called "negative sampling".

$$-\log \sigma\left({u_{w_O}}^\top v_{w_I}\right) - \sum_{i}^{N} \mathbb{E}_{w_i \sim P_n(w)} \log \sigma\left(-{u_{w_i}}^\top v_{w_I}\right)$$

Cbow Model Results

- 'beautiful' 'over', 'but', 'brown’,
- 'brown': 'has', 'jumps', "king's,
- 'fox': 'a', 'toast', 'brown’,
- 'lazy': 'quick', 'a', 'this’,
- 'the': 'is', 'over', 'quick'

Skip-Gram Results

Epoch: 1/30 Loss: **8.31590747833252** **<= Initial Training epoch**

Target Word | predicted context words / Similar words

- **can't** | firefox, ctrl+w, visible, right-click, error
- **this** | loop, skin, shouldn't, script, management
- **not** | cvs, launch, quickly, migration, than
- **to** | tabbing, everything, empty, drawn, plugin
- **password** | chars, connection, below, 2003, contain
- **all** | exists, xp, ibm, blocked, do
- **firefox** | can't, strict, ctrl+w, history, starts
- **for** | select, background, selection, centered, any
- **quickly** | not, every, app, server, options
- **garbled** | small, see, closes, position, destination
- **modern** | applet, text, tar, holding, because
- **crashing** | most, control, wish, blue, occurs
- **produces** | pages, talkback, each, dialogs, check
- **management** | parent, this, amount, quickly, manually
- **finishes** | paint, filename, redirects, rfe, write
- **anchor** | standard, happens, tags, fault, middle

Skip-Gram Results

Epoch: 30/30 Loss: 0.06470464915037155 **<= Last Training epoch**

Target Word | predicted context words / Similar words

- **open** | impossible, png, radio, fast, don’t
- **error** | started, was, strict, existing, rather
- **option** | accesskeys, realplayer, which, progress, visible
- **to** | whenever, searching, encryption, brings, secure
- **should** | malformed, responding, profile, 4, logging
- **firebird** | users, column, more, speed, return
- **javascript** | talkback, extension, install, leaving, sends
- **button** | checked, horizontal, child, something, every
- **hit** | protocol, follow, overflow, focus, true
- **servers** | offer, highlighted, sites, responding, way
- **such** | few, weird, changes, instances, fonts
- **separate** | take, renaming, moving, tar, scroll
- **case** | browse, fullscreen, addressbar, creating, dragged
- **dragged** | case, brings, transparent, created, unexpected
- **handler** | compile, closes, skin, having, breaks
- **example** | filename, your, numbers, advanced, macos

Word Visualization with t-SNE