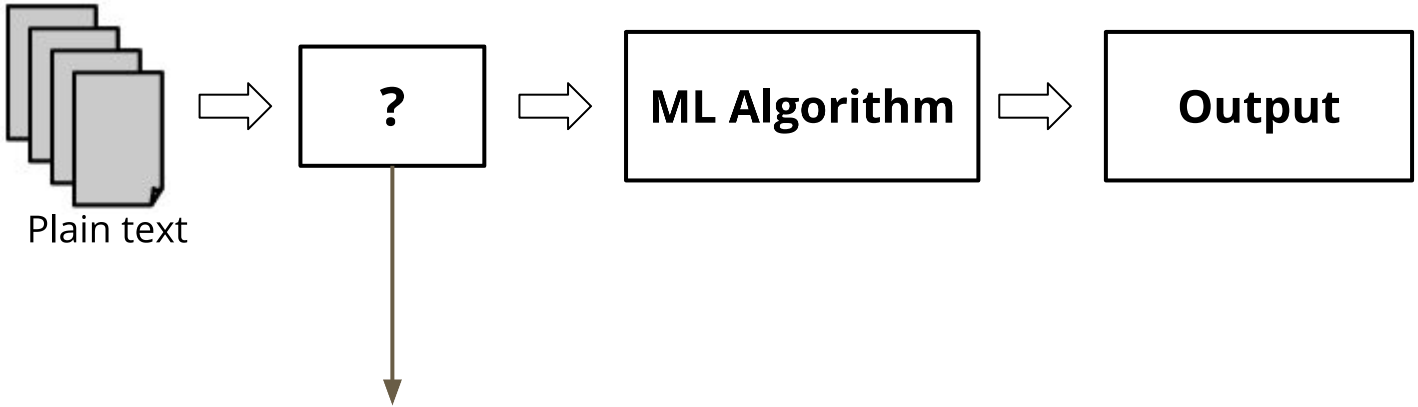
Word Embedding

Juan Luis García Mendoza, Mario Ezra Aragón, Adrián Pastor López Monroy, Luis Villaseñor Pineda, Manuel Montes y Gómez

Overview

- One-hot encoded representation
- Word embeddings (word2vec)
- Optimizing Computational Efficiency
- Exercises

How to represent text?



Text representation

1. Term Frequency (TF)
2. TF-IDF
3. **One-hot encoded**
4. **Embeddings**

One-hot encoded representation

If we have a vocabulary **V**, for each i^{th} word w_i

w_i = [0, 0, 0, …, 0, 1, 0, …, 0, 0, 0, …, |V|] where the i^{th} element is 1 and other elements are zero.

Examples

Bob and Mary are good friends.

V = {Bob,and,Mary,are, good,friends} **|V| = 6**

Bob = [1,0,0,0,0,0]
and = [0,1,0,0,0,0]
Mary = [0,0,1,0,0,0]
are = [0,0,0,1,0,0]
good = [0,0,0,0,1,0]
friends = [0,0,0,0,0,1]

Examples

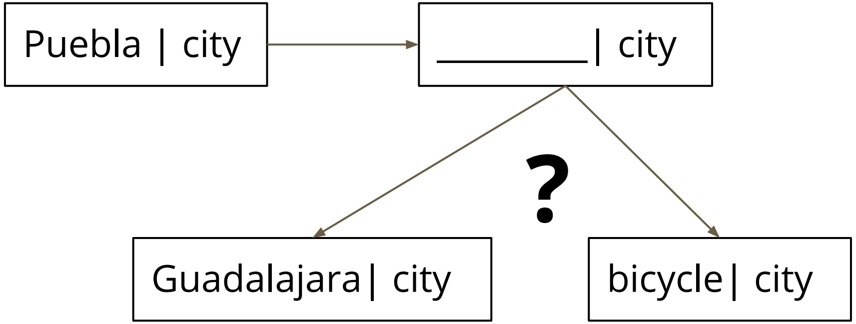
Puebla is an important city of Mexico

V = {Puebla,is,an,important,city,of,Mexico} **|V| = 7**

Puebla = [1,0,0,0,0,0,0]
is = [0,1,0,0,0,0,0]
an = [0,0,1,0,0,0,0]
important = [0,0,0,1,0,0,0]
city = [0,0,0,0,1,0,0]
of = [0,0,0,0,0,1,0]
Mexico = [0,0,0,0,0,0,1]

One-hot encoded representation

Puebla = [0,0,0,1,0,0,0,0,...,0,0,0,0]
Guadalajara = [0,1,0,0,0,0,0,0,...,0,0,0,0]
bicycle = [0,0,0,0,0,0,1,0,...,0,0,0,0]



cosine_sim(Puebla,Guadalajara) = 0

cosine_sim(Puebla,bicycle) = 0

Drawbacks

1. It does not encode the similarity between words in any way
2. Completely ignores the context in which the words are used.
3. This method becomes extremely ineffective for large vocabularies (sparse matrix).

One-hot encoding plays an important role even in the state-of-the-art word embedding learning algorithms.

Word embeddings

Learning the meaning of words without any human intervention?

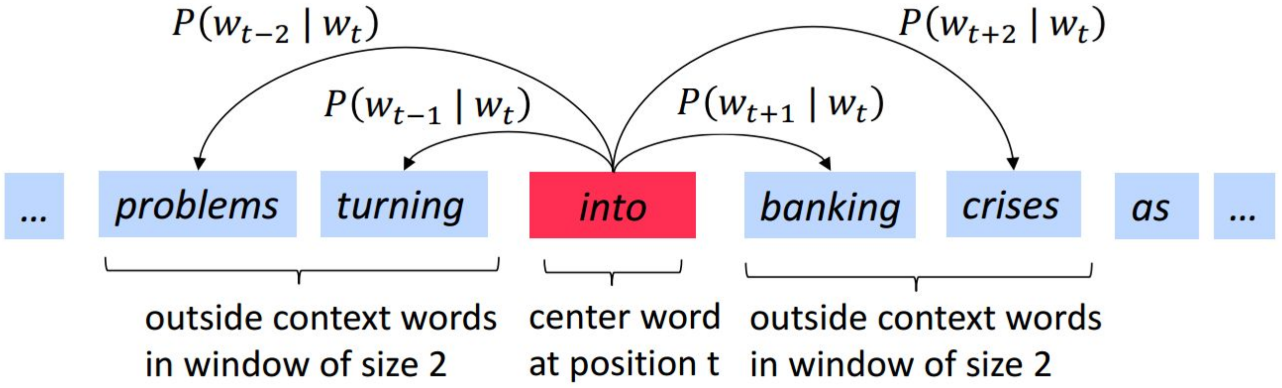
Word embeddings learns numerical representations of words by looking at the words surrounding a given word.

- word2vec
- Glove
- fastText

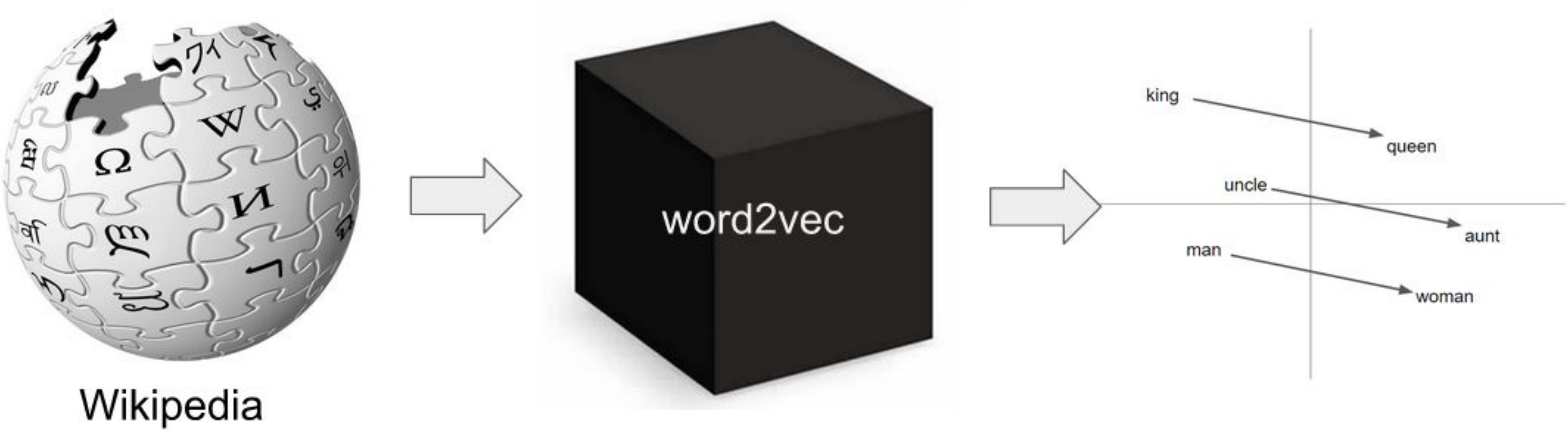
Word embeddings

Core idea: Word is represented by context in use

- When a **word** *w* appears in a **text**, its **context** is the set of words that appear **nearby** (within a **fixed-size window**).
- Use the **many contexts** of *w* to build up a representation of *w*.

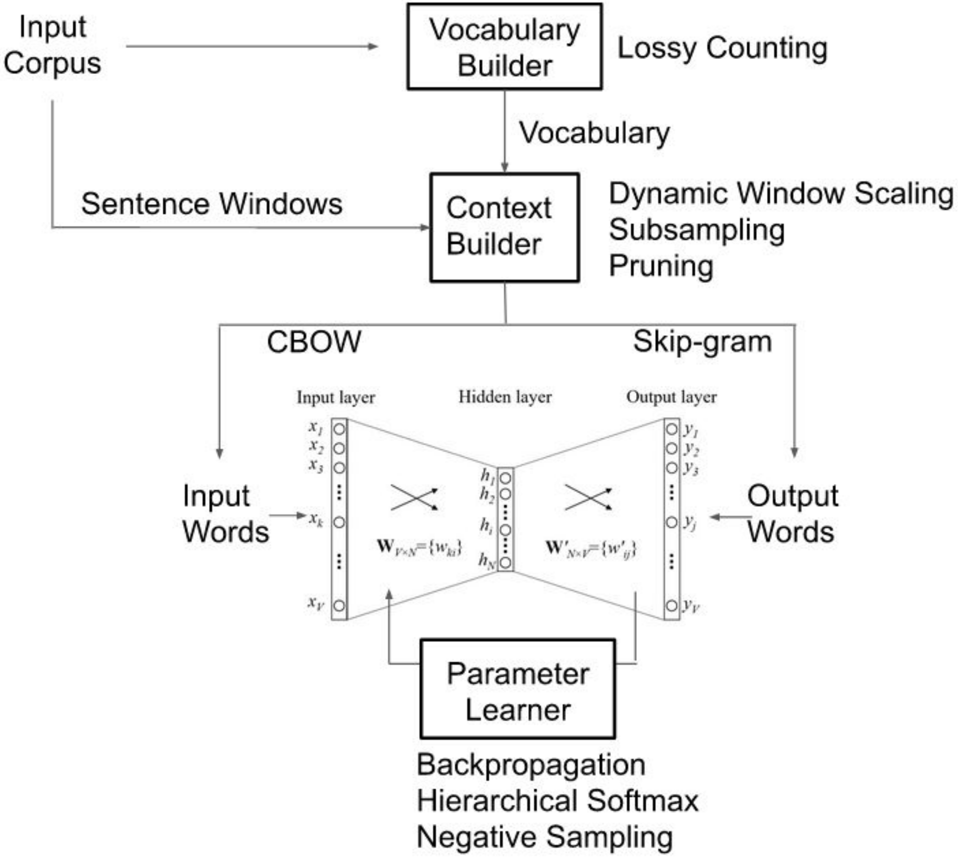


word2vec



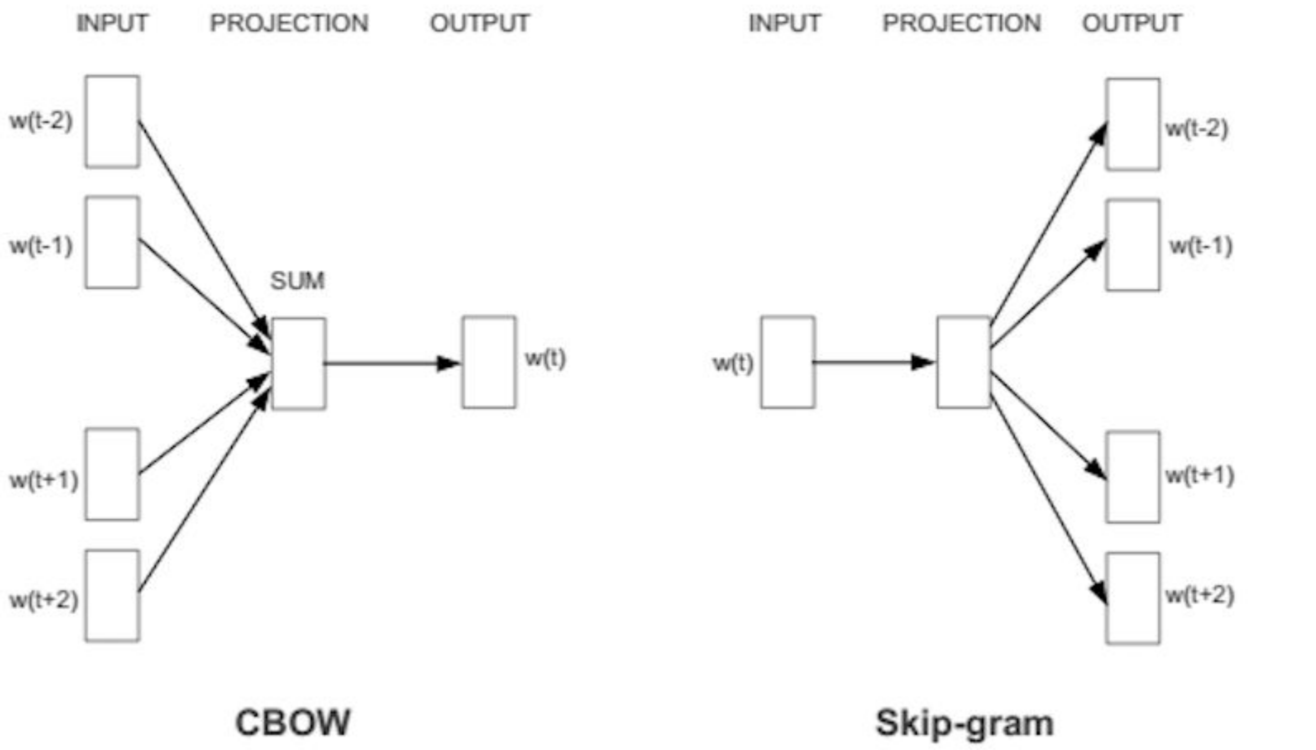
Rong, X. (2014). word2vec Parameter Learning Explained. ArXiv.Org:1411.2738v4 [Cs.CL], 1–21.

word2vec (black box)



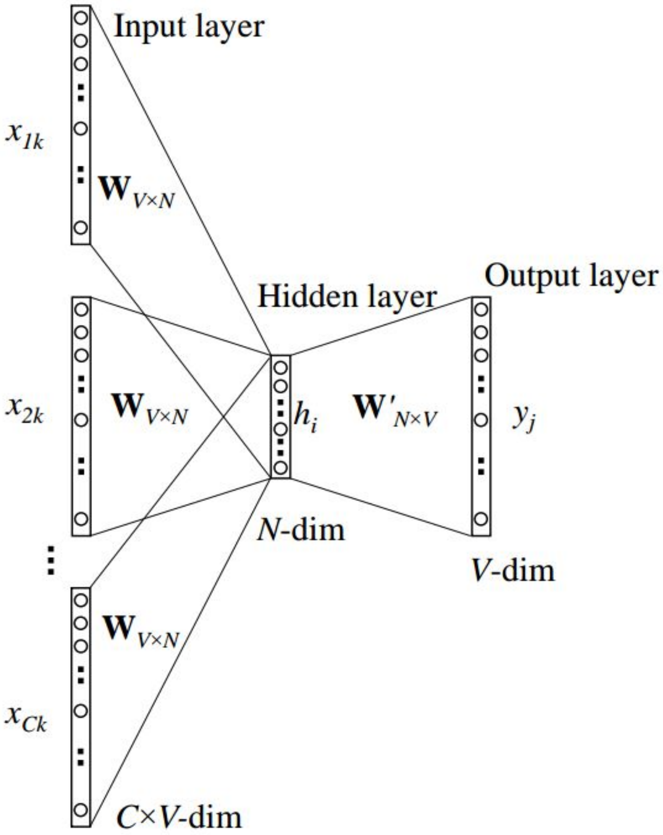
Rong, X. (2014). word2vec Parameter Learning Explained. ArXiv.Org:1411.2738v4 [Cs.CL], 1–21.

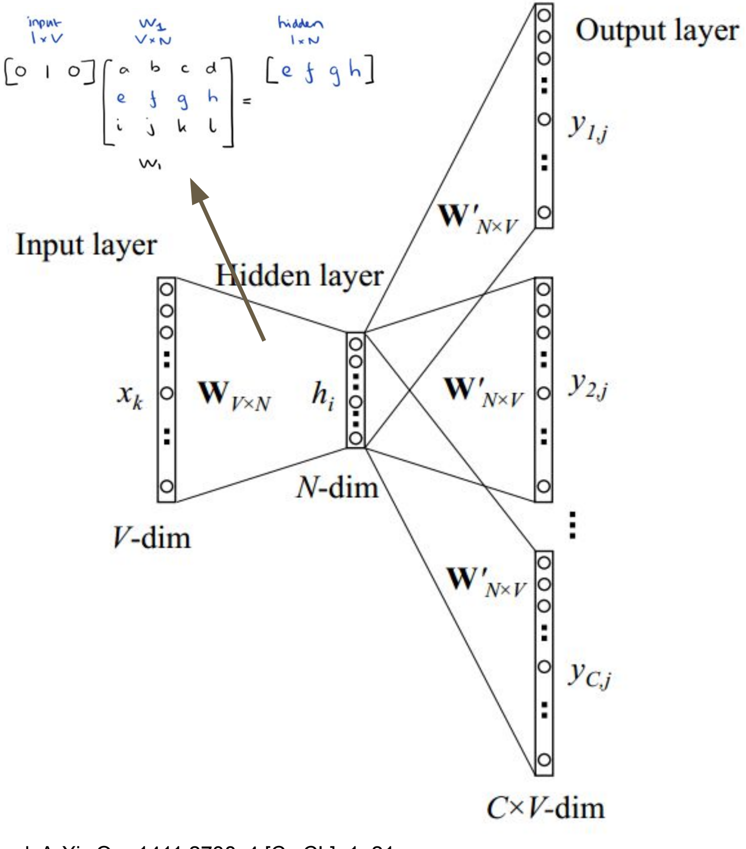
word2vec



Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. ArXiv:1301.3781v3 [Cs.CL]

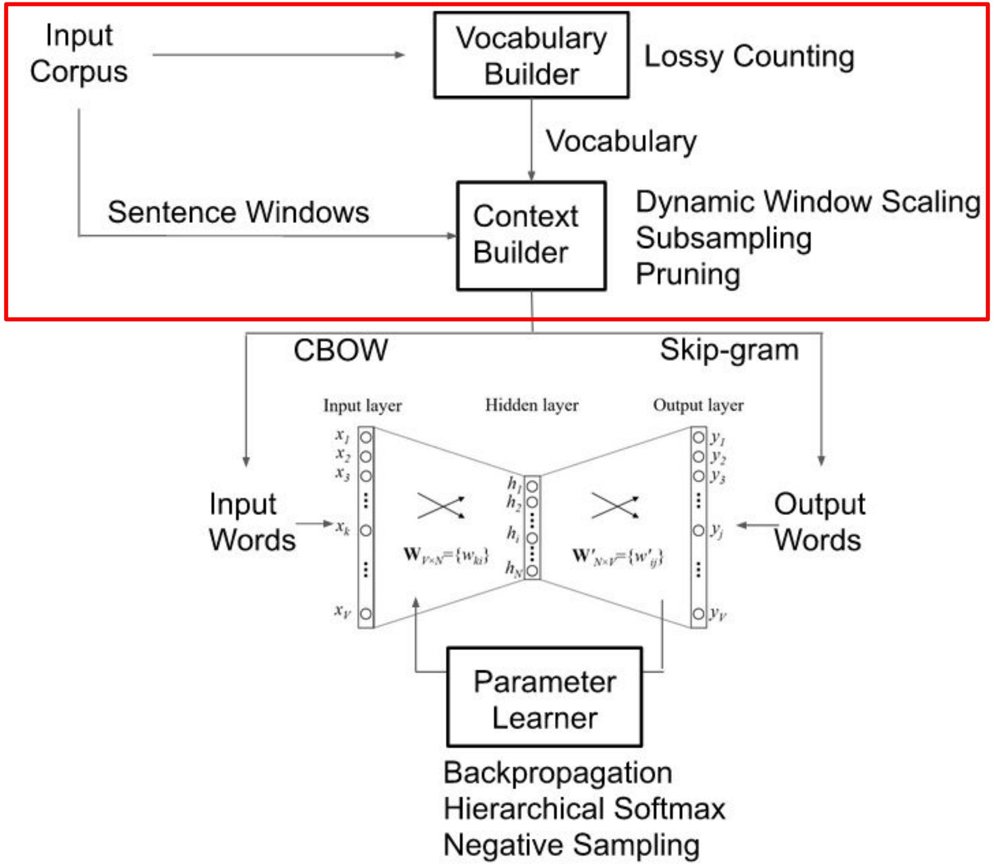
word2vec





Rong, X. (2014). word2vec Parameter Learning Explained. ArXiv.Org:1411.2738v4 [Cs.CL], 1–21.

word2vec (context builder)



Rong, X. (2014). word2vec Parameter Learning Explained. ArXiv.Org:1411.2738v4 [Cs.CL], 1–21.

Context Builder

Jay was hit by a _____ bus in...

by	a	red	bus	in

CBOW

Jay was hit by a red bus in...

by	a	red	bus	in

Skip-gram

input	output
red	by
red	a
red	bus
red	in

Jay Alammar, The Illustrated Word2vec

Context Builder (skip-gram)

Thou shalt not make a machine in the likeness of a human mind

thou	shalt	not	make	a	machine	in	the	…

thou	shalt	not	make	a	machine	in	the	…

thou	shalt	not	make	a	machine	in	the	…

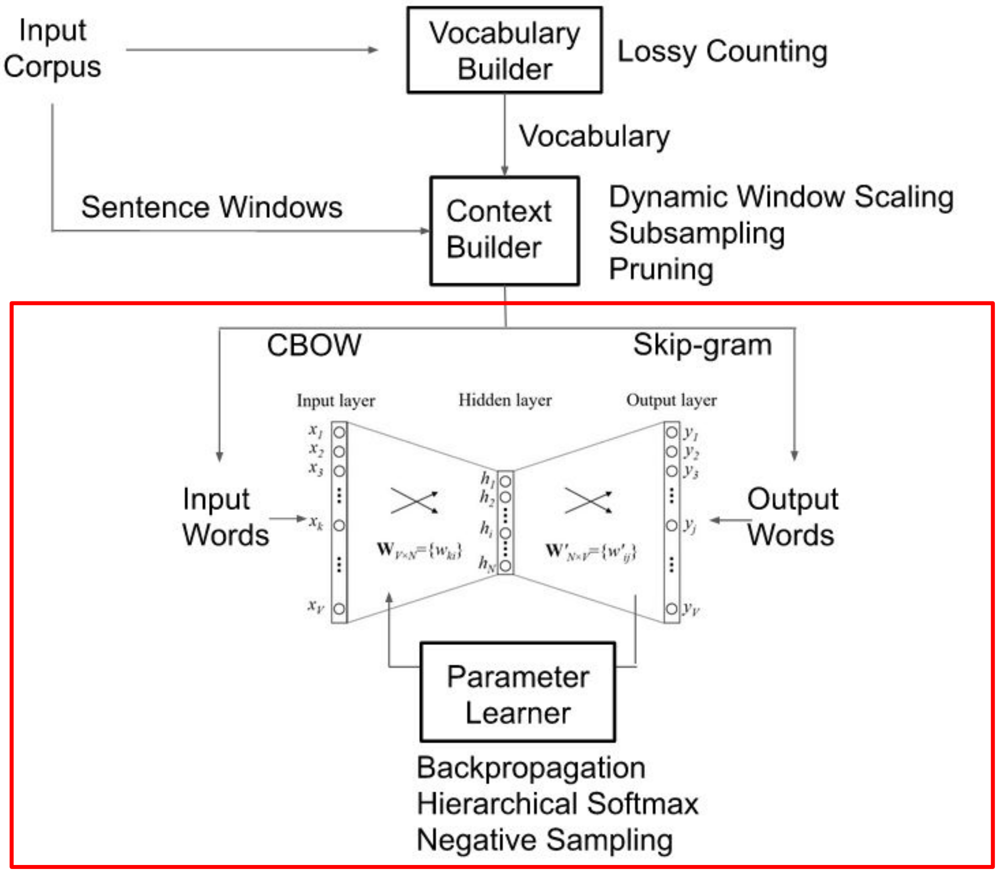
thou	shalt	not	make	a	machine	in	the	…

thou	shalt	not	make	a	machine	in	the	…

input word	target word
not	thou
not	shalt
not	make
not	a
make	shalt
make	not
make	a
make	machine
a	not
a	make
a	machine
a	in
machine	make
machine	a
machine	in
machine	the
in	a
in	machine
in	the
in	likeness

Jay Alammar, The Illustrated Word2vec

word2vec (neural networks)

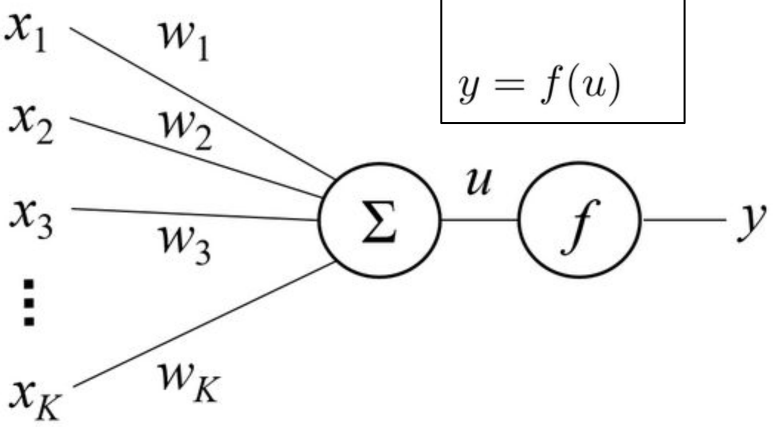


Rong, X. (2014). word2vec Parameter Learning Explained. ArXiv.Org:1411.2738v4 [Cs.CL], 1–21.

Basic Neuron

$$u = \sum_{i=0}^{K} w_i x_i$$

$$y = f(u)$$



$$f(u) = \frac{1}{1+e^{-u}}$$

$$E = \frac{1}{2}(t-y)^2$$

$$f(-u) = 1 - f(u)$$

$$\frac{\partial f(u)}{\partial u} = f(u)f(-u)$$

$$\frac{\partial E}{\partial y} = y - t$$

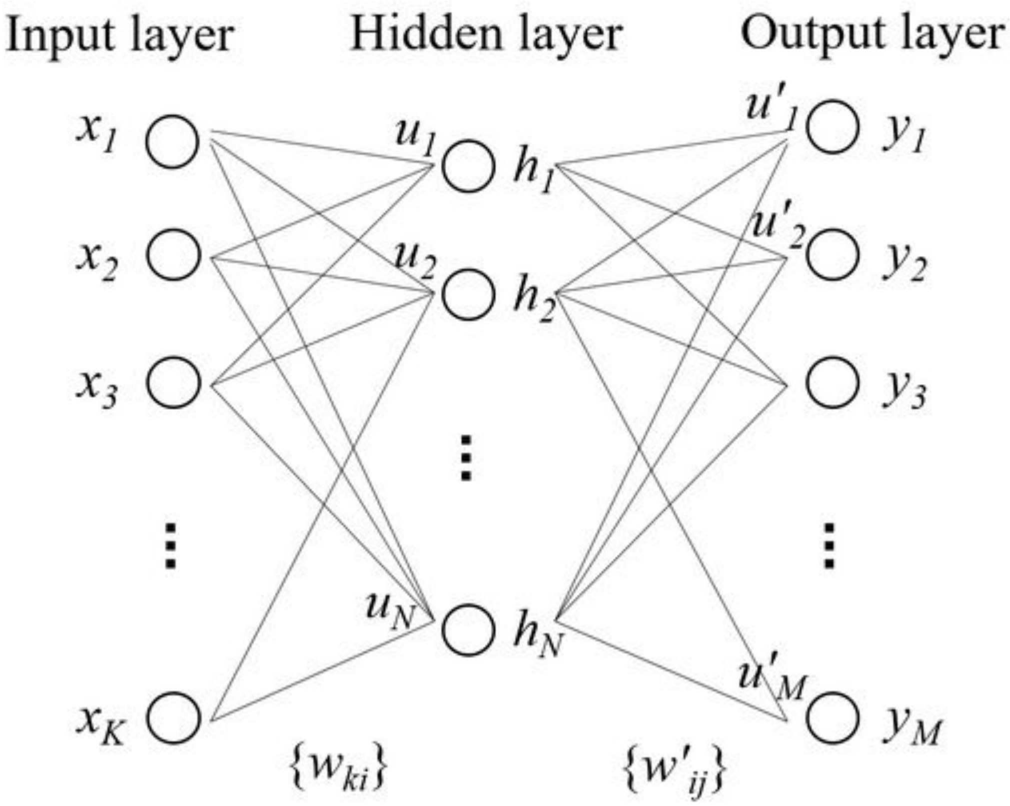
$$\begin{aligned} \frac{\partial y}{\partial u} &= \frac{\partial f(u)}{\partial u} \\ &= f(u)f(-u) \\ &= f(u)(1-f(u)) \\ &= y(1-y) \end{aligned}$$

$$\begin{aligned} \frac{\partial E}{\partial w_i} &= \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial u} \cdot \frac{\partial u}{\partial w_i} \\ &= (y-t) \cdot y(1-y) \cdot x_i \end{aligned}$$

$$\mathbf{w}^{(\text{new})} = \mathbf{w}^{(\text{old})} - \eta \cdot (y-t) \cdot y(1-y) \cdot \mathbf{x}.$$

Rong, X. (2014). word2vec Parameter Learning Explained. ArXiv.Org:1411.2738v4 [Cs.CL], 1–21.

Multilayer Neural Network



$$u_i = \sum_{k=1}^{K} w_{ki} x_k$$

$$h_i = f(u_i)$$

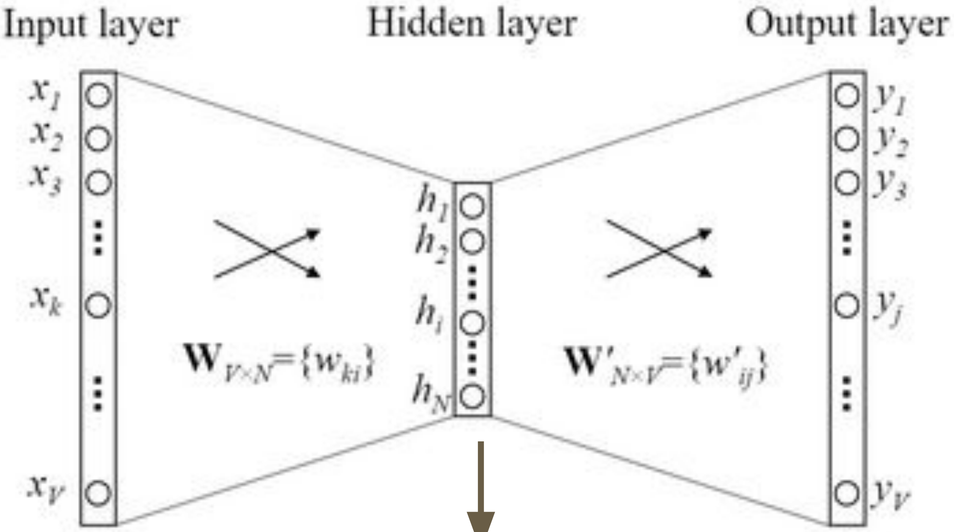
$$u'_j = \sum_{i=1}^{N} w'_{ij} h_i$$

$$y_j = f(u'_j)$$

$$E = \frac{1}{2} \sum_{j=1}^{M} (y_j - t_j)^2$$

Rong, X. (2014). word2vec Parameter Learning Explained. ArXiv.Org:1411.2738v4 [Cs.CL], 1–21.

word2vec



One-hot vector

Linear

Softmax

Minimize

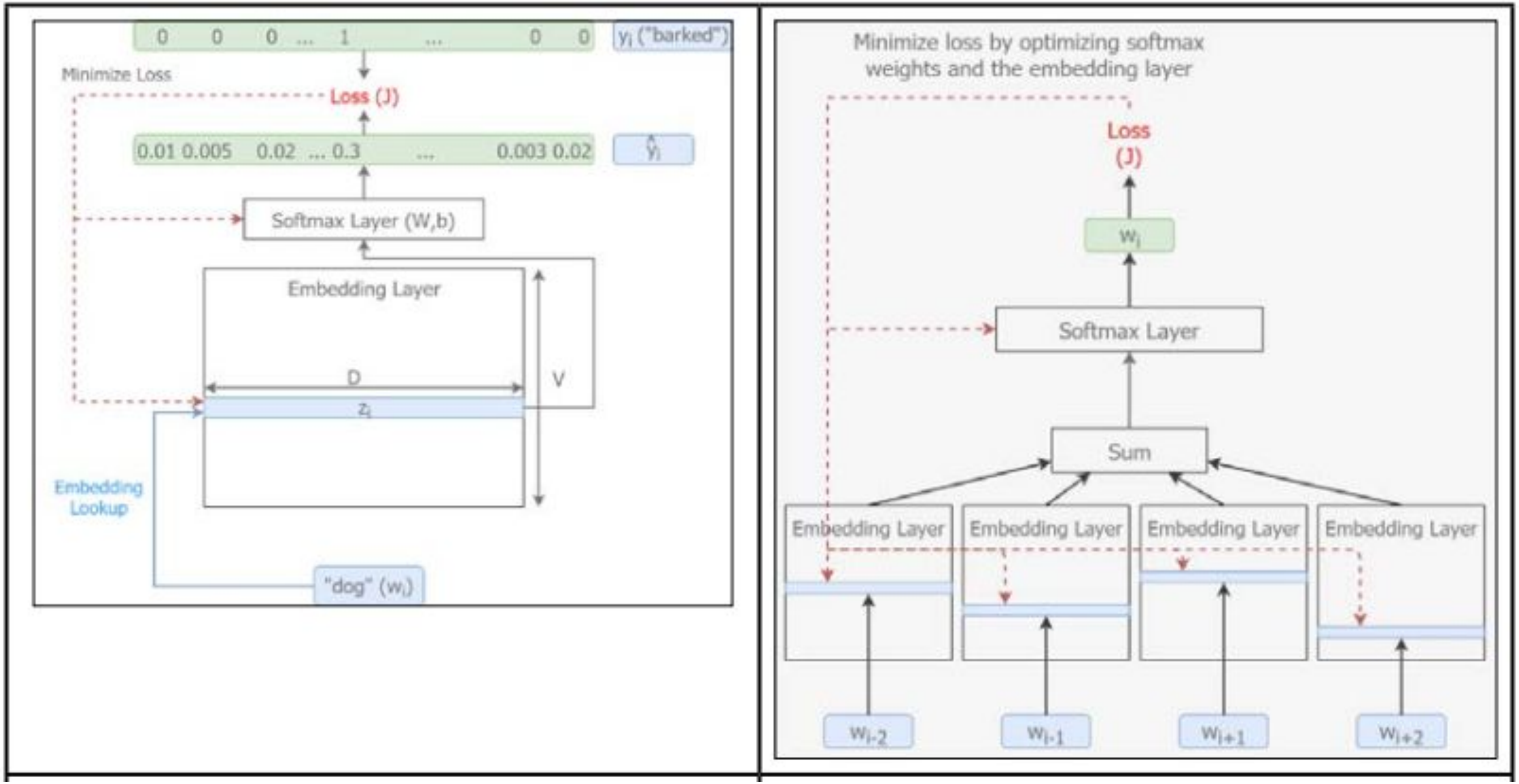
$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0} \log P(w_{t+j}|w_t;\theta)$$

Probability of the **central word c** from a **context word o**

$$P(o|c) = \frac{exp(u_o^T v_c)}{\sum_{w\in V} exp(u_w^T v_c)}$$

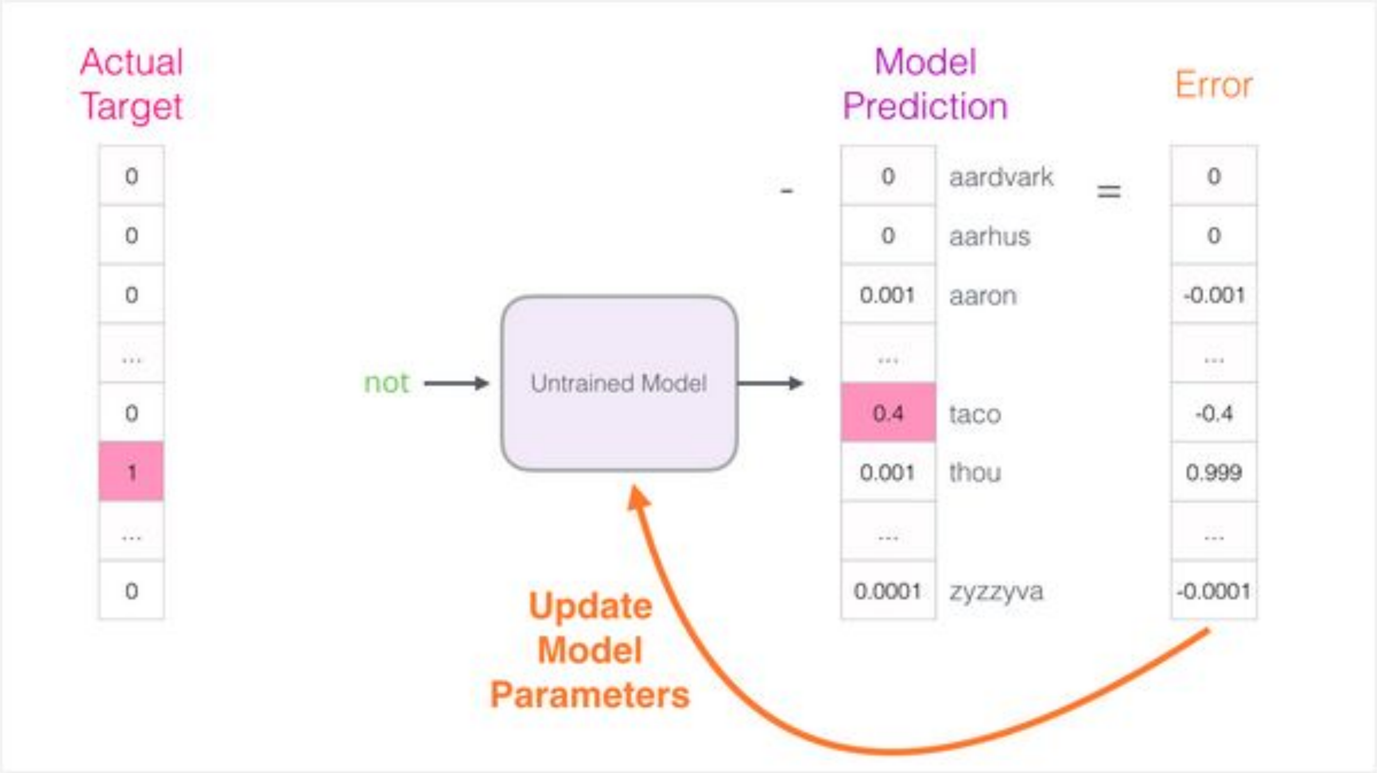
Rong, X. (2014). word2vec Parameter Learning Explained. ArXiv.Org:1411.2738v4 [Cs.CL], 1–21.

word2vec



Ganegedara, T. (2018). Natural Language Processing with TensorFlow. Packt Publishing Ltd.

Skip-gram (training process)



Jay Alammar, The Illustrated Word2vec

word2vec (Demo)

https://ronxin.github.io/wevi/

Optimizing Computational Efficiency

CONACYT
Consejo Nacional de Ciencia y Tecnología

- There exist **two vector** representations for each word in the vocabulary: **the input vector** and the **output vector**.
- Learning the **input vectors** is **cheap**; but learning the output vectors is **very expensive**.
- For **each training instance**, we have to iterate through **every word** in the vocabulary, compute their net input, probability prediction, their prediction error and finally use their prediction error to **update their output vector**.

Minimize

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0} \log P(w_{t+j}|w_t;\theta)$$

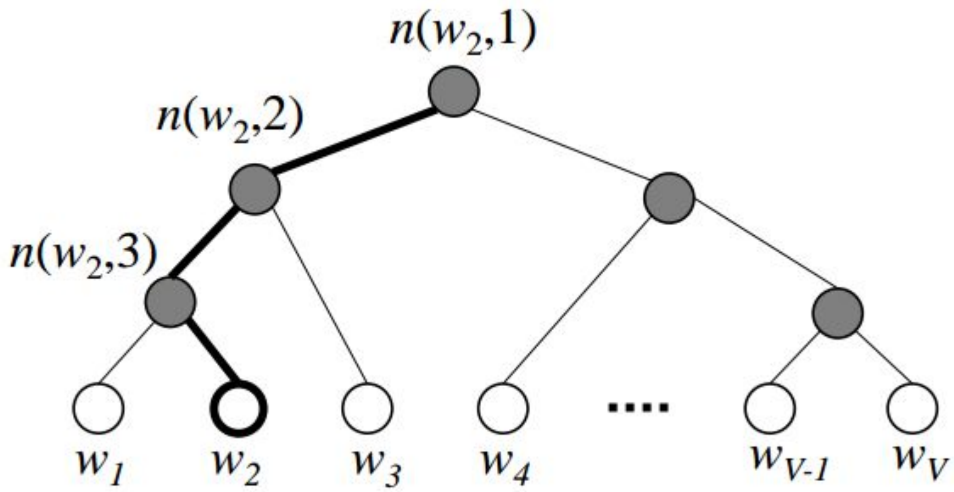
Hierarchical Softmax

Negative Sampling

$$P(o|c) = \frac{exp(u_o^T v_c)}{\sum_{w\in V} exp(u_w^T v_c)}$$

Rong, X. (2014). word2vec Parameter Learning Explained. ArXiv.Org:1411.2738v4 [Cs.CL], 1–21.

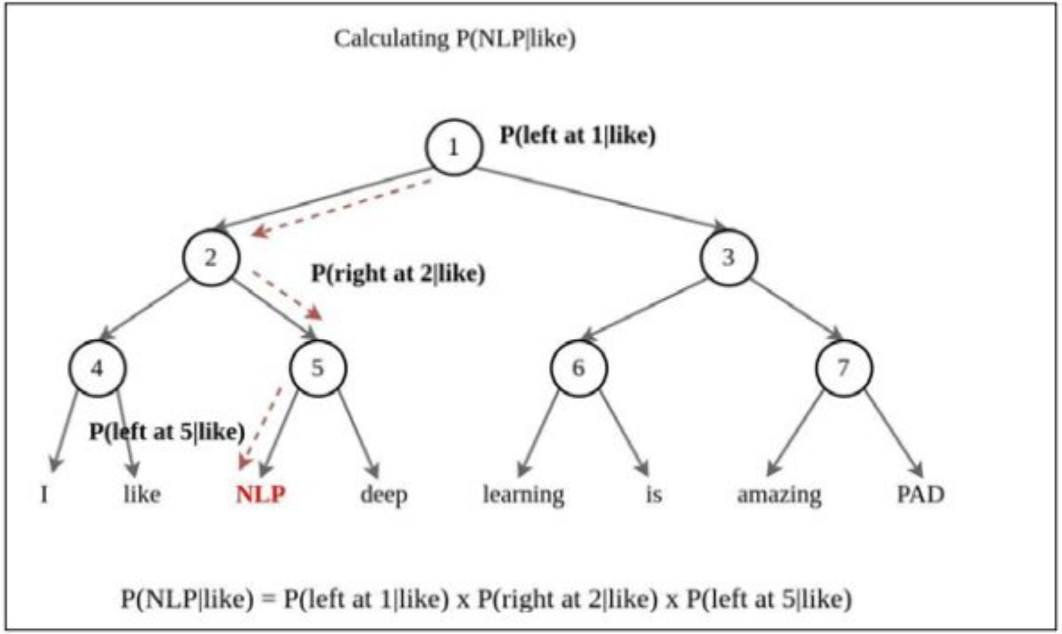
Hierarchical Softmax



- The vocabulary words are on the leafs.
- For each leaf unit, there exists a unique path from the root to the unit.
- This path is used to estimate the probability of the word represented by the leaf unit.
- There is no output vector representation for words.
- Each inner node has an output vector

Rong, X. (2014). word2vec Parameter Learning Explained. ArXiv.Org:1411.2738v4 [Cs.CL], 1–21.

Hierarchical Softmax



$$(NLP \mid like) = P(left\ at\ 1 \mid like)\ x\ P(right\ at\ 2 \mid like)\ x\ P(left\ at\ 5 \mid like)$$

- Since now we know how to calculate **P(w_j|w_i)**, we can use the original **loss function**.
- This method uses **only** the **weights** connected to the **nodes in the path** for calculation, resulting in a **high computational efficiency.**

Ganegedara, T. (2018). Natural Language Processing with TensorFlow. Packt Publishing Ltd.

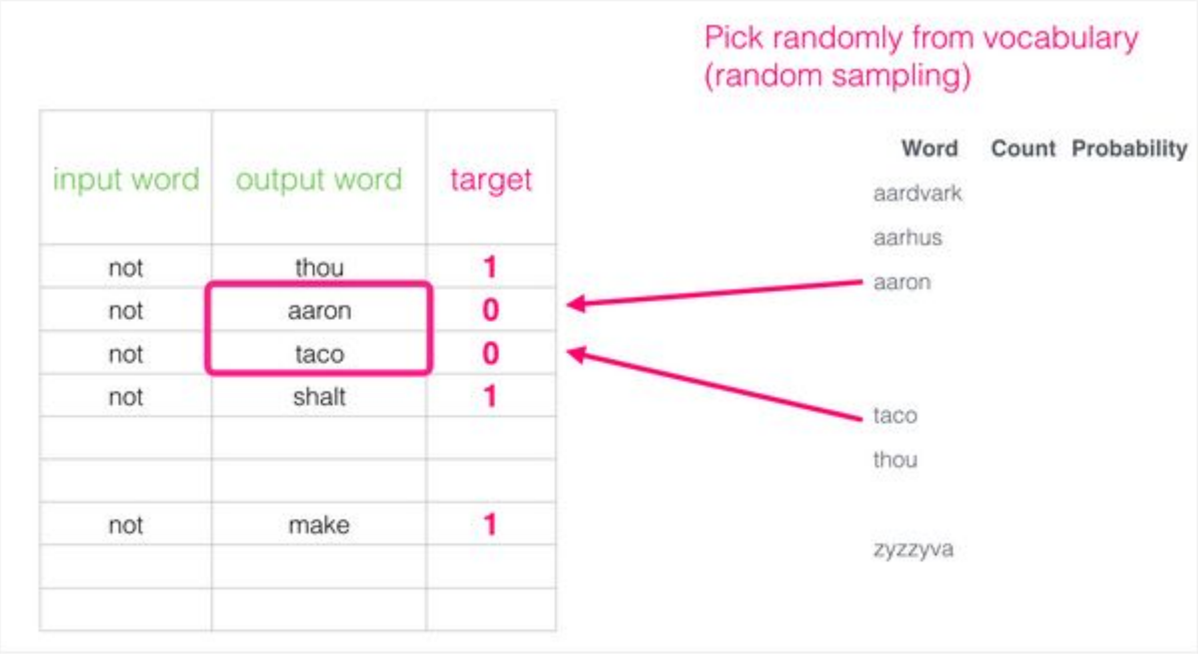
Negative Sampling

input word	target word
not	thou
not	shalt
not	make
not	a
make	shalt
make	not
make	a
make	machine

input word	output word	target
not	thou	1
not	shalt	1
not	make	1
not	a	1
make	shalt	1
make	not	1
make	a	1
make	machine	1

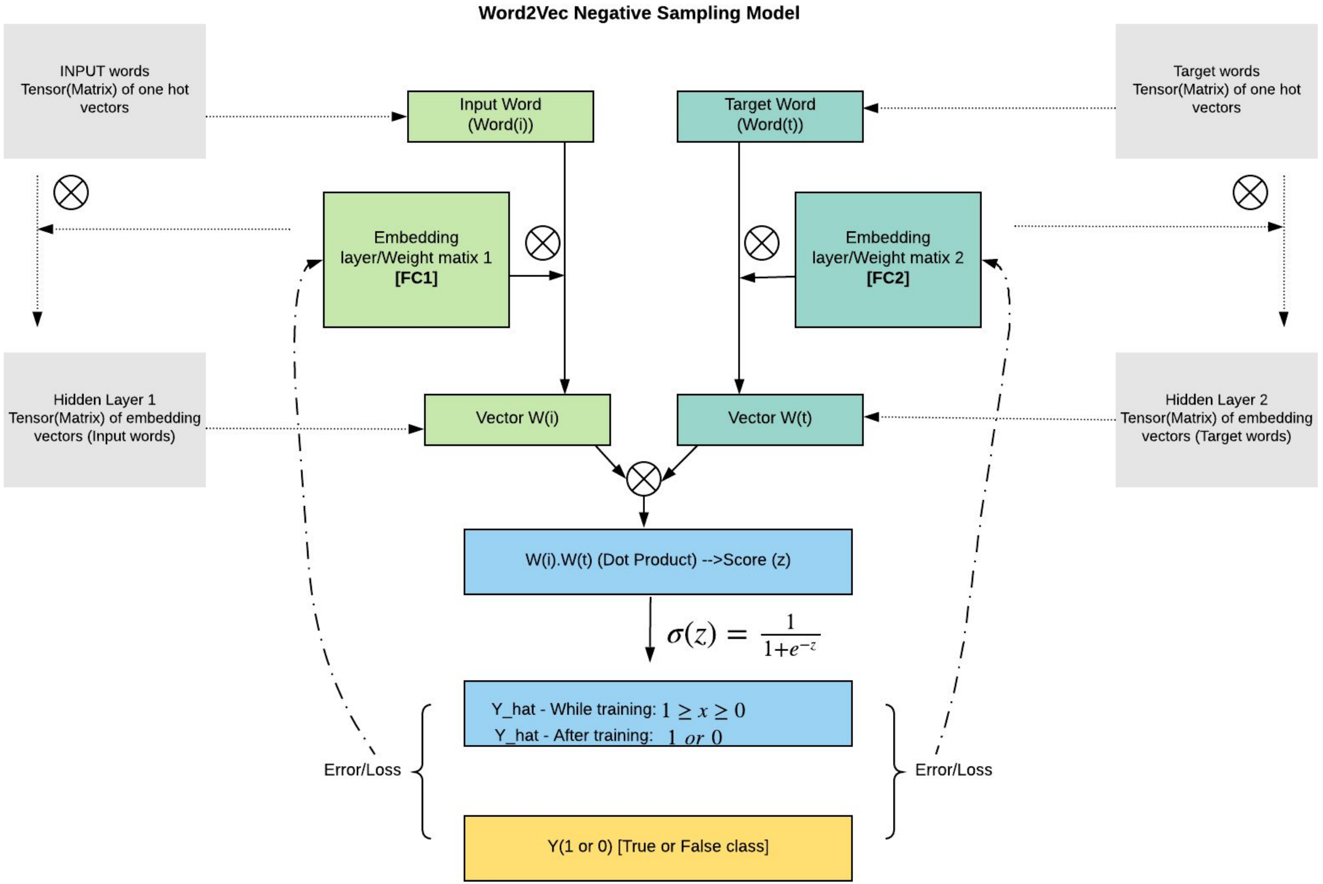
Jay Alammar, The Illustrated Word2vec

Negative Sampling



Jay Alammar, The Illustrated Word2vec

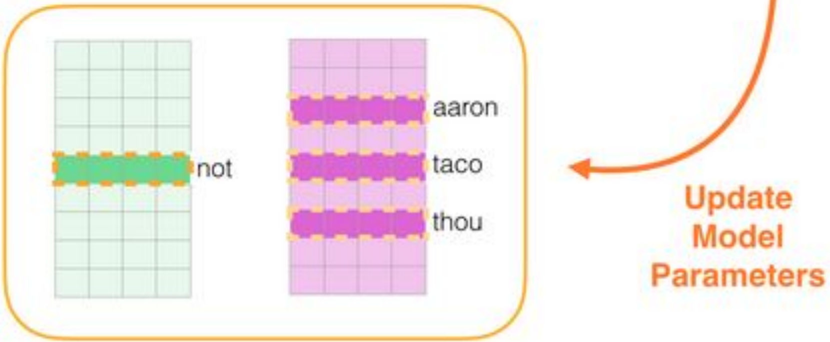
Negative Sampling



Munesh Lakhey, Word2Vec -Negative Sampling made easy

Negative Sampling

input word	output word	target	input • output	sigmoid()	Error
not	thou	1	0.2	0.55	0.45
not	aaron	0	-1.11	0.25	-0.25
not	taco	0	0.74	0.68	-0.68



Jay Alammar, The Illustrated Word2vec

Skip-gram or CBOW?

- Mikolov et al., 2013 suggests that **skip-gram works better in semantic tasks**, whereas **CBOW works better in syntactic tasks.**
- However, **skip-gram** appears to perform better than CBOW in **most tasks.**
- Various empirical evidence suggests that **skip-gram** works well with **large datasets** compared to CBOW.
- **Skip-gram** learn more meticulous representations because there is no averaging effect as in CBOW.

Ganegedara, T. (2018). Natural Language Processing with TensorFlow. Packt Publishing Ltd.

References

- Ganegedara, T. (2018). Natural Language Processing with TensorFlow. Packt Publishing Ltd. https://github.com/PacktPublishing/Natural-Language-Processing-with-TensorFlow
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. ArXiv:1301.3781v3 [Cs.CL]. https://arxiv.org/abs/1301.3781
- Mikolov, T., Yih, W., & Zweig, G. (2013a). Linguistic Regularities in Continuous Space Word Representations. Proceedings of NAACL-HLT 2013, 746–751. https://doi.org/10.3109/10826089109058901
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013b). Distributed Representations of Words and Phrases and their Compositionality. Advances in Neural Information Processing Systems, 3111–3119. https://doi.org/10.18653/v1/d16-1146
- Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 1532–1543. https://www.aclweb.org/anthology/D14-1162
- Rong, X. (2014). word2vec Parameter Learning Explained. ArXiv.Org:1411.2738v4 [Cs.CL], 1–21. http://arxiv.org/abs/1411.2738

Auxiliar Bibliography

- Wevi Demo
- Tensorflow Projector
- Natural Language Processing with Deep Learning CS224N/Ling284 (Stanford Course)
- Jay Alammar, The Illustrated Word2vec
- Munesh Lakhey, Word2Vec -Negative Sampling made easy

Questions?

Exercises

- Queen = King - Man + Woman
- word2vec (training)
- word2vec (fine tuning)
- Tensorflow Projector

Word Embedding

Juan Luis García Mendoza, Mario Ezra Aragón, Adrián Pastor López Monroy, Luis Villaseñor Pineda, Manuel Montes y Gómez