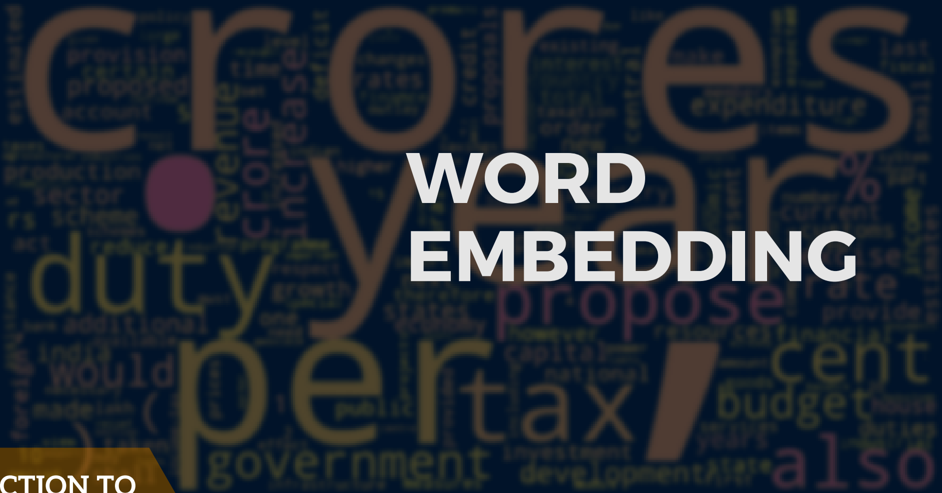


WORD EMBEDDING

Ramaseshan Ramachandran

INTRODUCTION TO LANGUAGE MODELS

July 17-21, 2023

WHAT IS CONTEXT?

- All words within a window or ideally within a sentence
- All content words within a window or sentence that fall in a certain frequency range
- All content words which stand in closest proximity to the word in question in the grammatical schema of each window or sentence

CONTEXT

- Context influences the word meaning
- Small boy, small car, small house, small island
- Words that occur in similar contexts will tend to have similar meanings
- Semantic similarity beween two words (w_x, w_y) is a function of how frequently they appeared in similar linguistic contexts
  - $\vec{w}_x \approx \vec{w}_y$ when the frequency of the context $(f_{C_{xy}(k)})$ with a window of size k in which both words w_x and w_y appeared is higher
- If $f_{C_{xy}}(k)$ is higher, then the semantic relationship of (w_x, w_y) is stronger
- Extending to multiple similar words for w_x:

  $\vec{w}_x \approx \vec{w}_{y_i}$ when the frequency of $C_{xy_i}(k)$ is higher, where $i = 1 \dots n$

  Note: Here $approx$ represents similarity

SEMANTIC SPACE

- A space where the similar words (synonyms, hyponyms, hypernyms) are classified and arranged in various axes
  - Colour (hypernym) - $\underbrace{Red, Green, Orange}_{co-hyponyms}$ (hyponym) - Attributional Similarity
- A space where the similar words (synonyms, hyponyms, hypernyms) are classified and arranged in various axes

- A model or models that automatically find similar words are known as Distributed Semantic Models (DSM)
- Semantically similar words are found automatically using co-occurrences/co-locations/context

  OR
- Words connected by similar patterns are probably semantically similar

DISTRIBUTED SEMANTIC MODELS

- Extract the meaning of the words using distributed linguistic properties
- Compute lexical co-occurrence of every word (co-locates with certain distance) with every other word in the Vocabulary
  - Linear proximity of words within a window is considered
  - They need not represent any relations

Example He drove the car through a *red* bridge. The verb *drove* relates to *red* and *bridge* only through the proximity, but carries no relations with *red* and bridge in terms of semantics

- Build a co-occurrence matrix using co-occurrence statistics
- Rows/columns in the matrix represent distributed semantic information of words

DISTRIBUTED SEMANTIC MODELS

I cook dinner every Sunday

...

I cooked dinner last Sunday

...

I am cooking dinner today

...

My son cooks dinner every Sunday

...

- The words in this corpus are related by association
- The verb cook, cooked, cooks and cooking are related due to its co-occurrence statistics - semantic relationship
- The words dinner and Sunday are similar due to associative relationship and due to co-occurrence

- In the COVID19 corpus it iis difficult to search and find *needle in a haystack*
- You will find needle related to $\underbrace{pain, illness, blood, drugs, syringe}_{Associative\, relationship}$

and not to thread, knitting, cloth

WORD SIMILARITY

You shall know a word by the company it keeps

- Firth, 1957

VECTOR REPRESENTATION OF WORDS

Let V be the unique set of terms and $|V|$ be the size of the vocabulary. Then every vector representing the word $\mathscr{R}^{|V|x1}$ would point to a vector in the V-dimensional space

ONE-HOT VECTOR - 1

Consider all the ≈39000 words (estimated tokens in English is ≈ 13M) in the Oxford Learner's pocket dictionary. We can represent each word as an independent vector quantity as follows in the real space $\mathscr{R}^{|V|X1}$

$$t^{a} = \begin{pmatrix} 1 \\ 0 \\ \dots \\ 0 \\ \dots \\ 0 \\ 0 \end{pmatrix} \quad t^{aback} = \begin{pmatrix} 0 \\ 1 \\ \dots \\ 0 \\ \dots \\ 0 \\ 0 \end{pmatrix} \quad \dots \; t^{zoom} = \begin{pmatrix} 0 \\ 0 \\ \dots \\ 0 \\ \dots \\ 1 \\ 0 \end{pmatrix} \quad t^{zucchini} = \begin{pmatrix} 0 \\ 0 \\ \dots \\ 0 \\ \dots \\ 0 \\ 1 \end{pmatrix}$$

This is a very simple codification scheme to represent words independently in the vector space. This is known as ***one-hot vector***.

ONE-HOT VECTOR - 2

In one-hot vector, every word is represented independently. The terms, *home*, *house*, *apartments*, *flats* are independently coded. With one-hot vector based model, the dot product

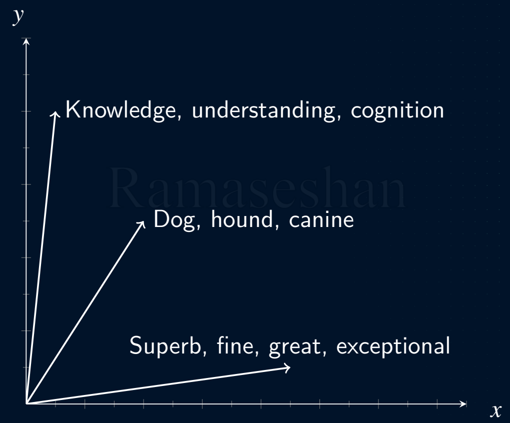
$$\left(t^{House}\right)^T \cdot t^{Apartment} = 0 \tag{1}$$

$$\left(t^{Home}\right)^T \cdot t^{House} = 0 \tag{2}$$

With one-Hot vector, there is no notion of similarity or synonyms.

RELATIONSHIP AMONG TERMS - SYNONYMS

We could represent all the synonyms of a word in one axis



POLYSEMOUS WORD - BANK

Synset('bank.n.01')	sloping land (especially the slope beside a body of water)
Synset('depository-financial-institution.n.01')	a financial institution that accepts deposits and channels the money into lending activities
Synset('bank.n.03')	a long ridge or pile
Synset('bank.n.10')	a flight maneuver; aircraft tips laterally about its longitudinal axis (especially in turning)
Synset('trust.v.01')	have confidence or faith in

Bank appears in different word senses - or the meaning of the word is determined by the context in which appears

POLYSEMOUS WORD - PROGRAM

Synset('plan.n.01')	a series of steps to be carried out or goals to be accomplished
Synset('program.n.02')	a system of projects or services intended to meet a public need
Synset('broadcast.n.02')	a radio or television show
Synset('platform.n.02')	a document stating the aims and principles of a political party
Synset('program.n.05')	an announcement of the events that will occur as part of a theatrical or sporting event
Synset('course_of_study.n.01')	an integrated course of academic studies
Synset('program.n.07')	(computer science) a sequence of instructions that a computer can interpret and execute
Synset('program.n.08')	a performance (or series of performances) at a public presentation
Synset('program.v.01')	arrange a program of or for
Synset('program.v.02')	write a computer program

SYNONYMS

small.a.01	['small', 'little']
minor.s.10	['minor', 'modest', 'small', 'small-scale', 'pocket-size', 'pocket-sized']
humble.s.01	['humble', 'low', 'lowly', 'modest', 'small']
little.s.07	['little', 'minuscule', 'small']
belittled.s.01	['belittled', 'diminished', 'small']
potent.a.03	['potent', 'strong', 'stiff']
impregnable.s.01	['impregnable', 'inviolable', 'secure', 'strong', 'unassailable', 'hard'] He has such an impregnable defense (Cricket-Very hard to find the gap between the bat and the pad)
solid.s.07	['solid', 'strong', 'substantial']
strong.s.09	['strong', 'warm']
firm.s.03	['firm', 'strong'] - firm grasp of fundamentals

CONTEXTUAL UNDERSTANDING OF TEXT

You shall know a word by the company it keeps - (Firth, J. R. 1957)

- In order to understand the word and its meaning, it not enough if we consider only the individual word
- The *meaning* and *context* should be central in understanding word/text
- Exploit the context-dependent nature of words
- Language patterns cannot be accounted for in terms of a single entity
- The *collocation*, a particular word consistently co-occurs with the other words, gives enough clue to understand a word and its meaning

UNDERSTANDING A WORD FROM ITS CONTEXT

The view from the top of the mountain was
The view from the summit was
La vue du sommet de la montagne *é*tait
Mtazamo wa juu wa mlima huo ulikuwa

- awesome/(*impressionnante*, *impressionnant*)
- breathtaking
- amazing
- stunning/(*superbe*)
- astounding
- astonishing
- awe-inspiring
- extraordinary
- incredible/(*incroyable*)
- unbelievable
- magnificent
- wonderful/(*ajabu*)
- spectacular
- remarkable/(*yakuvutia*)

SEMANTICALLY CONNECTED VECTORS

- Identify a model that enumerates the relationships between terms
- Identify a model that tries to put similar items closer to each other in some space or structure
- Build a model that discovers/uncovers the semantic similarity between words and documents in the latent semantic domain
- Develop a distributed word vectors or dense vectors that captures the linear combination of word vectors in the transformed domain
- Transform the term-document space into a synonymy and a semantic space

METHODS TO CREATE WORD VECTORS

- Brown clustering - statistical algorithms for assigning words to classes based on the frequency of their co-occurrence with other words
- Hyperspace Analogue to Language - HAL
- Correlated Occurrence Analogue to Lexical Semantic - COALS
- Latent Semantic Analysis or Latent Semantic Indexing
- Global Vectors - GloVe
- Neural networks using skip grams and CBOW
  - CBOW - uses surrounding words to predict the center of words
  - Skip grams use center of words to predict the surrounding words

WORD SIMILARITY

- Sparse vectors are too long and not very convenient as features machine learning
- Abstracts more than just frequency counts
- It captures neighborhood words that are connected by synonyms

You shall know a word by the company it keeps

- Firth, 1957

WORD VECTOR EXAMPLES

Similar words for apple

```
'apple',  0
'iphone',  0.266
'ipad',  0.287
'apples',  0.356
'blackberry',  0.361
'ipod',  0.365
'macbook',  0.383
'mac',  0.391
'android',  0.391
'google',  0.395
'microsoft',  0.418
'ios',  0.433
'iphones',  0.445
'touch',  0.446
'sony',  0.447
```

Similar words for - american

```
'american', 0
'america', 0.255
'americans', 0.312
'u.s.', 0.320
'british', 0.323
'canadian', 0.329
'history', 0.356
'national', 0.364
'african', 0.374
'society', 0.375
'states', 0.386
'european', 0.387
'world', 0.394
'nation', 0.399
'us', 0.399
```

VECTOR DIFFERENCE BETWEEN TWO WORDS

$$vec(apple) - vec(iphone)$$

```
('raisin', 0.5744591153088133)
('pecan', 0.5760617374141159)
('cranberry', 0.5840016172254104)
('butternut', 0.5882322018694753)
('cider', 0.5910795032086132)
('apricot', 0.6036644437522422)
('tomato', 0.6073715970323961)
('rosemary', 0.6150986936477657)
('rhubarb', 0.6157884153793192)
('feta', 0.6183016129045151)
('apples', 0.6226003361980218)
('avocado', 0.6235366677962004)
('fennel', 0.6306016018912576)
('chutney', 0.6312524337590703)
('spiced', 0.6327632200841328)
```

Hyperspace Analogue To Language[1] (HAL)

[1] Lund, K., Burgess, C. Producing high-dimensional semantic spaces from lexical co-occurrence. Behavior Research Methods, Instruments, & Computers 28, 203-208 (1996).

HAL ALGORITHM

Require a big corpus >5 GB for a reasonable similarity measures

1. Preprocess to limit the vocabulary size
2. Perform two scans using a ramping window of size 11 - first $\rightarrow$ direction and later in the $\leftarrow$ direction
3. Use the first word as the key word and the rest as context words
4. Use the last word as the key and rest as the context words, during the $\leftarrow$ scanning
5. The nearest neighbor of the key gets the weight 10 and the 10th word gets the weight 1
6. Construct an incidence matrix using the co-occurrence values
7. Concatenate two word vectors found for every word (row and column) in the matrix Concatenate them to get the word vectors for all the words in the vocabulary.
8. The number of elements in the word vecord will be $2||V|$

HAL SCANNING

Left2Right Scanning

the	horse	raced	past	the	barn	fell
K	5	4	3	2	1	0
	K	5	4	3	2	1
		K	5	4	3	2
			K	5	4	3
				K	5	4
					K	5

Right2Left Scanning

fell	barn	the	past	raced	horse	the
K	5	4	3	2	1	0
	K	5	4	3	2	1
		K	5	4	3	2
			K	5	4	3
				K	5	4
					K	5

Incidence Matrix

	the	horse	raced	past	the	barn	fell
the	2						
horse	5						
raced	4						
past	3						
barn	3						
fell	0						

HAL EXPERIMENT

160 million words from Usenet news groups

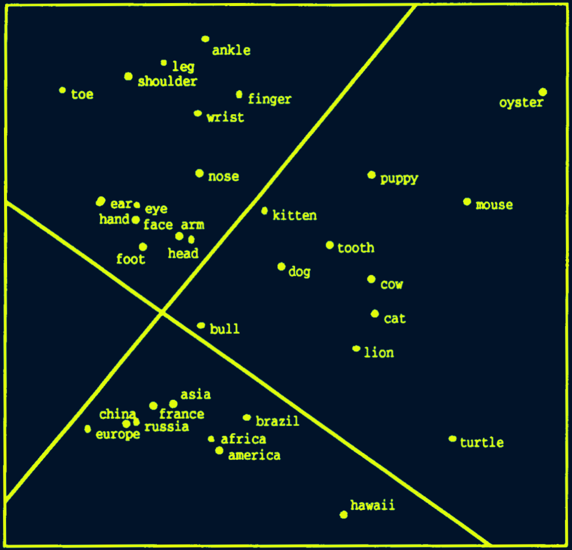
Window size = 10

- Vocabulary - Words with a frequency ¿ 50
- Zipf's law is used to eliminate most common and rare words
- Two word vectors are obtained for every word
- Minkowski distance measure is used for computing word similarities

$$d_{x_i y_j} = \sqrt[r]{|x_i - y_i|^r}$$

- The word vectors produce high dimensional semantic space - associative

HAL WORD VECTORS - SIMILARITY CHART



Correlated Occurrence Analogue to Lexical Semantic[2] (COALS)

[2] Rhode et al, "An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence", CACM, 2006, 8, 627-633

COALS ALGORITHM

1. Gather co-occurrence counts, typically ignoring closed-class neighbors and using a ramped, size 4 window:

$$1\ 2\ 3\ 4\ \boxed{0}\ 4\ 3\ 2\ 1$$

2. Discard all but the m (14,000, in this case) columns reflecting the most common open-class words.
3. Convert counts to word pair correlations, set negative values to 0, and take square roots of positive ones.
4. The semantic similarity between two words is given by the correlation of their vectors.

COALS RESULTS

Nearest neighbours and their percent correlation similarities for a set of nouns

	gun		point		mind		monopoly	
1)	46.4	handgun	32.4	points	33.5	minds	39.9	monopolies
2)	41.1	firearms	29.2	argument	24.9	consciousness	27.8	monopolistic
3)	41.0	firearm	25.4	question	23.2	thoughts	26.5	corporations
4)	35.3	handguns	22.3	arguments	22.4	senses	25.0	government
5)	35.0	guns	21.5	idea	22.2	subconscious	23.2	ownership
6)	32.7	pistol	20.1	assertion	20.8	thinking	22.2	property
7)	26.3	weapon	19.5	premise	20.6	perception	22.2	capitalism
8)	24.4	rifles	19.3	moot	20.4	emotions	21.8	capitalist
9)	24.2	shotgun	18.9	distinction	20.1	brain	21.6	authority
10)	23.6	weapons	18.7	statement	19.9	psyche	21.3	subsidies

SOME INSIGHTS

- The majority of the correlations are negative
- Words with negative correlations do not contribute well to finding similarity than the ones with positive correlation
- Closed-class words (147) convey syntactic information than semantic - could be removed from the correlation table punctuation marks, she, he, where, after, ...

Word2Vec[3]

[3] Mikolov, Tomas, Kai Chen, Gregory S. Corrado and Jeffrey Dean, "Efficient Estimation of Word Representations in Vector Space." International Conference on Learning Representations (2013)

GOAL

- Process each word in a Vocabulary of words to obtain a respective numeric representation of each word in the Vocabulary
- Reflect semantic similarities, Syntactic similarities, or both, between words they represent
- Map each of the plurality of words to a respective vector and output a single merged vector that is a combination of the respective vectors

CONTEXT WORDS AND CENTRAL WORD

- **Continuous Bag of Words (CBOW)** Models – A central word is surrounded by context words. Given the context words identify the central word
  - Wish you many more happy returns of the day
- **Skip Gram Model**- Given the central word, identify the surrounding words
  - Wish you many more happy returns of the day

CONTINUOUS BAG OF WORDS (CBOW)

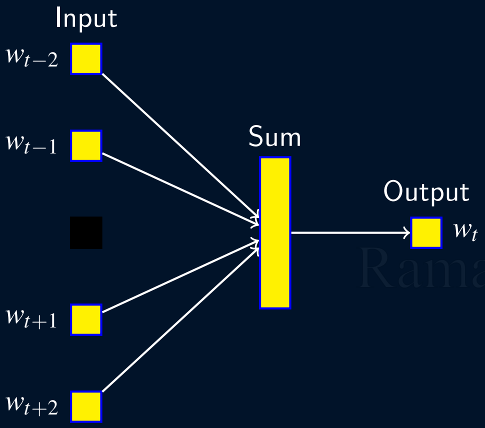


Figure: The CBOW architecture predicts the current word based on the context words of length n. Here the window size is 5

CBOW uses the sequence of words "Wish", "you", "a", "happy", "year" as a context and predicts or generates the central word "new"

- CBOW is used for learning the central word
- Maximize probability of word based on the word co-occurrences within a distance of n

SKIP GRAM MODEL

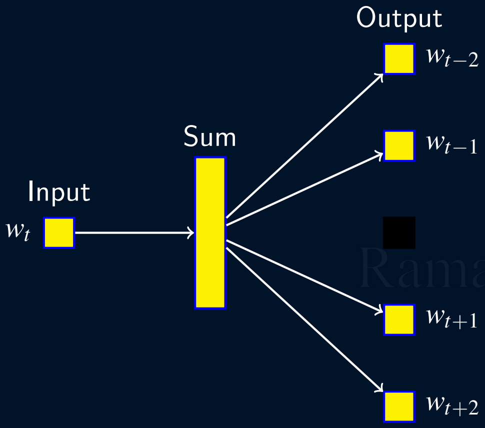


Figure: The SG architecture predicts the one context word at a time based on the center word. Here the window size is 5

SG uses the central word "new" and predicts the context words "Wish", "you", "a", "happy", "year"

- SG is used to learn the context words given the central word
- Maximize probability of word based on the word co-occurrences within a distance of $[-n, +n]$ from the center word

SOURCE PREPARATION FOR TRAINING

Source Text	Training Samples
[**Wish** | you | many] more happy returns of the day →	$(wish, you)$ $(wish, many)$
[Wish | **you** | more | happy] returns of the day →	$(you, Wish)$ $(you, more), (you, happy)$
[Wish | you | **many** | more | happy] returns of the day →	$(many, Wish), (many, you)$ $(many, more), (many, happy)$
Wish [you | many | **more** | happy | returns] of the day →	$(more, many), (more, you)$ $(more, happy), (more, returns)$
Wish you [many | more | **happy** | returns | of] the day →	$(happy, many), (happy, more)$ $(happy, returns), (happy, of)$
Wish you many [more | happy | **returns** | of | the] day →	$(returns, more), (returns, happy)$ $(returns, of), (returns, the)$
Wish you many more [happy | returns | **of** | the | day] →	$(of, happy), (of, returns)$ $(of, the), (of, day)$

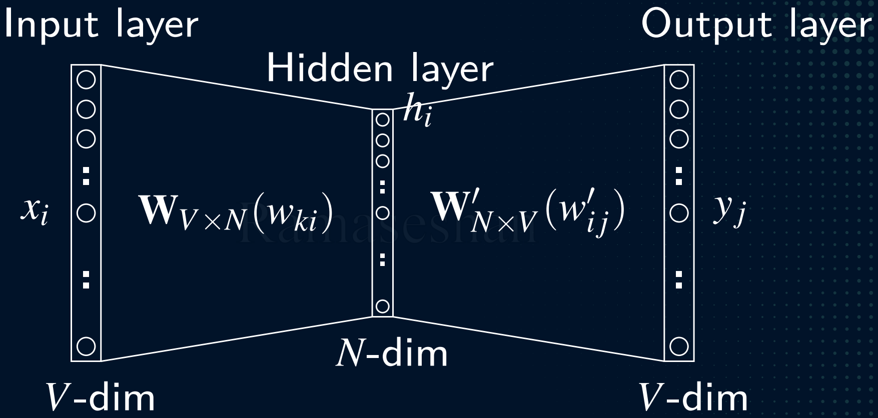


Figure: A CBOW model with only one word as input[**DBLP:journals/corr/Rong14**]. The layers are fully connected

Here the W and W' are learned

INPUT LAYER

$$t^{aback} = \begin{pmatrix} 0 \\ 1 \\ \dots \\ 0 \\ \dots \\ 0 \\ 0 \end{pmatrix} \dots t^{zoom} = \begin{pmatrix} 0 \\ 0 \\ \dots \\ 0 \\ \dots \\ 1 \\ 0 \end{pmatrix} t^{zucchini} = \begin{pmatrix} 0 \\ 0 \\ \dots \\ 0 \\ \dots \\ 0 \\ 1 \end{pmatrix}$$

$x_k = 1$ and $x'_k = 0, \forall k' \neq k$

HIDDEN LAYER

This neural network is fully connected. Input to the network is a one-hot vector. W is the N-dimensional vector representation of the word, v_w^T, presented as input

$$\mathbf{h} = \mathbf{W^T X} \tag{3}$$

Now v_{wI} of the matrix (W) is the vector representation of the input one-hot vector w_I. From (3), h is a linear combination of input and weights.
In the same way. we get a score for u_j

$$u_j = \mathbf{v}'^{\mathbf{T}}_{\mathbf{w_j}}\mathbf{h} = \mathbf{v}'^{\mathbf{T}}_{\mathbf{w_j}}\mathbf{v_{w_I}} \tag{4}$$

where v_{w_I} is the vector representation of the input word w_I and v'_{w_j} is the j^{th} column of (W')

OUTPUT LAYER

At the output layer, we apply the softmax to get the posterior distribution of the word(s). It is obtained by,

$$p(w_j|w_I) = y_j \tag{5}$$

where y_j is the output of the j^{th} unit in the output layer

$$y_j = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u'_j)} \tag{6}$$

$$= \frac{\exp(\mathbf{v'^T_{wj}} v_{wI})}{\sum_{j'=1}^{V} \exp((\mathbf{v'^T_{w_{j'}}}) v_{w_I})} \tag{7}$$

where $\mathbf{v}_w$, $\mathbf{v}'_w$ are the input vector (word vector) and output vector (feature vector) representations, of w_j and $w_{j'}$, respectively

UPDATE WEIGHTS - HIDDEN-OUTPUT LAYERS

The learning/training objective is to maximize (7) or minimize the error between the target and the computed value of the target which is $y_j^* - t$ and t is same as the input vector, in this case. We use cross-entropy as it provides us with a good measure of "error distance"

$$\begin{aligned} max\, p(w_o|w_I) &= max(\log(y_j*)) - -\text{Maximize} && (8) \\ -E &= u_j - \log(y_{j*}) - -\text{minimize} && (9) \\ &= u_j * - \log \sum_{j'=1}^{V} \exp(u'_j) && (10) \end{aligned}$$

where
w_o is the output word and E is the loss function.
It is the special case of cross-entropy measurement between two probabilistic distributions u_{j*} and $u_{j'}$

- $\log p(x)$ is well scaled
- Selection of step size is easier
- With $p(x)$ multiplication may yield to near zero causing *underflow*
- For better optimization, $\log p(x)$ is considered (multiplication $\rightarrow$ addition)

UPDATE WEIGHTS (HO) - MINIMIZATION OF E

To minimize E,take the partial derivative of E with respect to j^{th} unit of u_j

$$\frac{\partial E}{\partial u_j} = y_j - t_j = e_j \tag{11}$$

where e_j is the prediction error. Taking partial derivative with respect to the hidden-output weights, we get,

$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial w'_{ij}} = e_j \cdot h_i \tag{12}$$

Using the above equation (12),

$$w'^{\,new}_{ij} = w'^{\,old}_{ij} - \eta e_j \cdot h_i \text{ or} \tag{13}$$

$$\mathbf{v}^{(\mathbf{new})}_{\mathbf{w_j}} = \mathbf{v}'^{(\mathbf{old})}_{\mathbf{w_j}} - \eta e_j \cdot \mathbf{h} \quad \text{for } j = 1, 2, 3, \ldots, V \tag{14}$$

UPDATE INPUT TO HIDDEN WEIGHTS

Taking the derivative with respect to h_i, we get

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j \cdot w'_{ij} = \mathbf{EH_i} \tag{15}$$

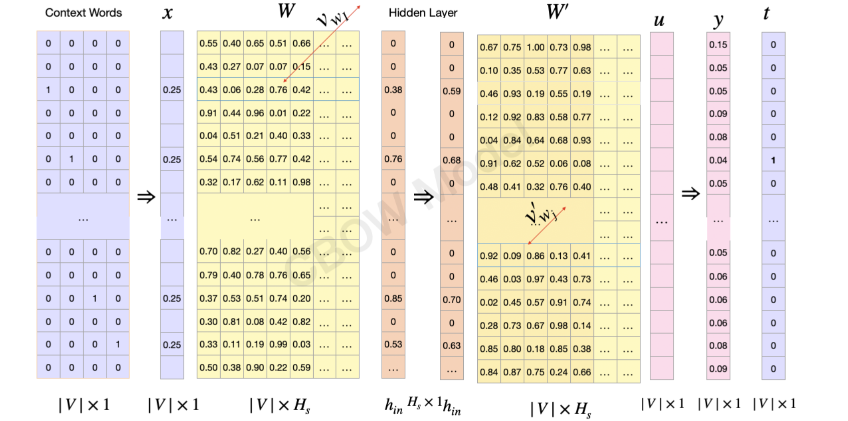
Taking the derivative with respect to w_{ki}, we get

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial w_{ki}} = \mathbf{EH_i}.x_k \tag{16}$$

Now the weights are updated using

$$v_{w_i}{}^{(new)} = v_{wi}{}^{(old)} - \eta \mathbf{EH^T} \tag{17}$$

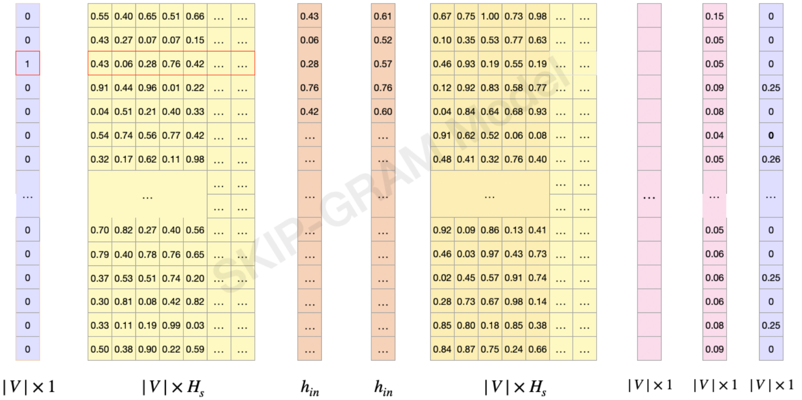
CBOW IN MATRIX FORM



Forward Pass	$x = \frac{1}{C}\sum_{c=1}^{C} x_c$	$h = \sigma(W^T x)$	$y = softmax(W'h)$ is a probability distribution	Maximise $(y - t)$

The numbers shown in the tables are for representation only. They do not represent actual computations.

SG IN MATRIX FORM



$X, W^e, W^c, h,$ and $\hat{Y}$ represent the input, embedding matrix, context matrix and output vector, respectively. h is the copy of the vector corresponding to the index of the input word in the vocabulary for skip-gram model. For CBOW model, it represents a vector whose elements are obtained using the the average linear sum of the vectors corresponding to the context words. x is represented by

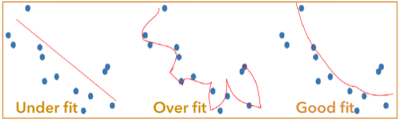
$$x = \frac{1}{C}(x_1 + x_2 + \ldots + x_n), \tag{1}$$

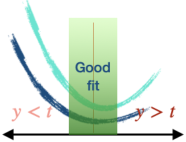
where n represents the number of context words. For skip-gram, $n = 1$ and for CBOW model $n > 1$.

UNDER FIT, OVER FIT AND GOOD FIT

During the forward pass, h_{in} is obtained using the linear combination the initial weight matrix and the one-hot vector of the corresponding context word (skip-gram model) or corresponding context words (CBOW model). Note that the index of the word in the vocabulary is represented by the position of 1 in the OHV. During the computations of Y, the propagated properties of the word vector, h, further passes on the properties of the word vector to the predicted output, Y, through the W^c.

y		t		e
0.15		0		0.15
0.05		0		0.05
0.05		0		0.05
0.09		0		0.09
0.08		0		0.08
0.04		1		-0.96
0.05	$-$	0	$=$	0.05
...		...		
0.05		0		0.05
0.06		0		0.06
0.06		0		0.06
0.06		0		0.06
0.08		0		0.08
0.09		0		0.09
$\lvert V \rvert \times 1$		$\lvert V \rvert \times 1$		$\lvert V \rvert \times 1$





If $y - t$ is lower, then the predicted word is not close to the target word. In other words, the context vectors which are supposed to be similar to the target vector needs to be adjusted into to minimise the differences.

If $y - t$ is higher, then the predicted word moved further away from the target, again the context vectors are further away from the expected result.

If $y - t$ is very small, then we are close to achieving the desired result. The learning becomes slow, the weight updates do not yield any significant change In such case, learning is complete and the context vectors are now closer to the target vector.

Since the target and the estimated values are probability distributions, it is a good idea to use the cross-entropy loss for the propagation of error. In other words, the movement of the word vector towards the expected one is proportional to the error.

OUTPUT GENERATION

The idea of Word2Vec model is to estimate the conditional probability $p(w_o|w_I)$ and if given by

$$p(w_O|w_I) = \hat{y}_j$$

and each element value or neuron value $\hat{Y}$ is computed using Softmax

$$\begin{aligned} y_j &= \frac{\exp(v'_{w_j}h)}{\sum_k \exp(v'_{w_k}h)} \\ &= \frac{\exp(v'_{w_j}v^T_{w_I})}{\sum_k^{|V|} \exp(v'_k v^T_{w_I})} \end{aligned}$$

where w_I is the input word vector (skip-gram model) or the average of the context word vectors for which we want to estimate the word vector of the central word. The idea of using *softmax* is to amplify the maxima in the output layer, U, such that $\forall i, y_j \in (0,1), \in \mathscr{R}^{|V|}$ and $\sum_j (y_j) = 1$. In order to propagate the change in the output layer, y, with respect to the input vector, v_j, we compute the error. There are several ways of computing the change or loss. We choose log-loss or cross-entropy loss as both the predicted output and the target are probability distributions.The changes are computed with respect to all the element of the output layer, context embedding matrix and input embedding matrix . The error is used to change these elements to make sure that the relation between input and the output are well described in the emerging model. During the back propagation of the error, the error when the error is minimum, the context and input vectors moved closer each other.

OUTPUT GENERATION

One word Learning	CBOW	Skip-gram
$h = \mathbf{v^T}.x$	$h = \frac{1}{C}(\mathbf{v_1}^T.x1 + \mathbf{v_2}^T.x2 + ...\mathbf{v_C}^T x_C)$	$h = \mathbf{v}^T.x$
$u_j = \mathbf{v}'^T_{w_j}.\mathbf{h}$	$u_j = \mathbf{v}'^T_{w_j}.\mathbf{h}$	$\mathrm{u}_{c,j} = \mathbf{v}'_{w_j}{}^T.\mathbf{h}$ for $\mathrm{c} = 1,2,3,...\mathrm{C}$
$\mathbf{v}'^{new}_{w_I} = \mathbf{v}'^{old}_{w_I} - \eta e_j.\mathbf{h}$ for $j = 1, 2, 3, ...\|V\|$	$\mathbf{v}'^{new}_{w_I} = \mathbf{v}'^{old}_{w_I} - \eta e_j.\mathbf{h}$	$\mathbf{v}'^{new}_{w_I} = \mathbf{v}'^{old}_{w_I} - \eta \mathbf{EI}_j.\mathbf{h}$ where $\mathrm{EI} = \sum_{c=1}^{C} e_{c,j}$ for $\mathrm{j} = 1,2,3,...\|V\|$
$\mathbf{v}^{new}_{w_I} = \mathbf{v}^{old}_{w_I} - \eta \mathbf{EH}^T$	$\mathbf{v}^{new}_{w_I} = \mathbf{v}^{old}_{w_I} - \frac{1}{C}\eta \mathbf{EH}^T$	$\mathbf{v}^{new}_{w_I} = \mathbf{v}^{old}_{w_I} - \eta \mathbf{EH}^T$ where $\mathrm{EH}_j = \sum_{j=1}^{\|V\|} \mathbf{EI}_i w'_{ij}$
$\mathrm{E} = \text{-}\mathrm{u}_{j^*} + log \sum_{j'=1}^{\|V\|} exp(u_{j'})$	$\mathrm{E} = \text{-}\mathrm{u}_{j^*} + log \sum_{j'=1}^{\|V\|} exp(u_{j'})$	$\mathrm{E} = \text{-}\sum_{j'=1}^{C} u_{j_c^*} +$ $\mathrm{C.}\ \log \sum_{j'=1}^{\|V\|} exp(u_{j'})$

Reference: Xin Rong, word2vec Parameter Learning Explained, https://arxiv.org/abs/1411.2738

SOME INSIGHTS ON OUTPUT-HIDDEN-INPUT LAYER WEIGHT UPDATES

- The prediction error E propagates the weighted sum of all words in the vocabulary to every output vector v'_j
- The change in the input vector is defined by the output vector which in turn is updated due to the prediction error
- The model parameters accumulate the changes until the system reaches a state of equilibrium
- Ideally the $v_j \cdot v'_j$ will result in an identity
- The rows in the Input-Hidden layer (v_j) stores the features of the words in the vocabulary V

GloVe[4]

[4] Glove: Pennington, Jeffrey and Socher, Richard and Manning, Christopher Booktitle, "Global vectors for word representation", Proceedings of the conference on EMNLP, 2014

FastText[5]

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov, "Enriching word vectors with subword information", CoRR abs/1607.04606. http://arxiv.org/abs/1607.04606

SUMMARY OF WORD EMBEDDINGS

- Maps words to vectors of real numbers
- Learned from a corpus of text
- Capture the semantic meaning of words
- Used as input to many downstream applications, such as text classification, sentiment analysis, and machine translation, context embedding

Thank you

Ramaseshan Ramachandran