4c: Word Vectors

Statistics of Language

Definitions, synonyms and antonyms

How do we capture the true meaning of a word, or its relationship to other words?

Words are sometimes arranged into a taxonomy — for example, a bird is a mammal is an animal, etc.

We can also look at dictionary definitions, or lists of synonyms and antonyms. Here is a list of synonyms for the word 'elegant':

> stylish, graceful, tasteful, discerning, refined, sophisticated, dignified, cultivated, distinguished, classic, smart, fashionable, modish, decorous, beautiful, artistic, aesthetic, lovely; charming, polished, suave, urbane, cultured, dashing, debonair; luxurious, sumptuous, opulent, grand, plush, high-class, exquisite

Human effort is required to compile taxonomies, dictionary definitions and lists of synonyms and antonyms. Some important relationships may be missing, nuances may be lost. Words like 'king' and 'queen' are not quite synonyms nor antonyms because they are similar in some attributes but opposite in others.

Is it possible to extract statistical properties and relationships between words automatically, without human involvement?

We will discuss a number of techniques for statistical language processing, using this children's rhyme as an example.

There was a Crooked Man
who walked a crooked mile
and found a crooked sixpence
upon a crooked stile.
He bought a crooked cat,
who caught a crooked mouse,
and they all lived together
in a little crooked house.

http://www.kearley.co.uk/images/uploads/JohnPatiencePJ03.gif

Counting word frequencies

Perhaps the simplest thing we can do to analyse a text is just to count the frequency with which each word occurs.

word	doc 1	doc 2	doc X
a	.	.	7
all	.	.	1
and	.	.	2
bought	.	.	1
cat	.	.	1
caught	.	.	1
crooked	.	.	7
found	.	.	1
he	.	.	1
house	.	.	1
in	.	.	1
little	.	.	1
lived	.	.	1
man	.	.	1
mile	.	.	1
mouse	.	.	1
sixpence	.	.	1
stile	.	.	1
there	.	.	1
they	.	.	1
together	.	.	1
upon	.	.	1
walked	.	.	1
was	.	.	1
who	.	.	2

This can be useful for document classification. Each column of the matrix becomes a vector representing the corresponding document. Some words like 'a', 'and' occur frequently in all (or most) documents; other words occur frequently only in certain types of documents, so are more distinctive. For example, words like 'cat', 'mouse', 'house' tend to occur in children's books or rhymes, while other groups of words might be characteristic of legal documents, political news, sporting results, etc.

Words occurring many times in one document may skew the vector. Sometimes it is better if the matrix entries are just '1' or '0' indicating whether the word occurs at all.

Counting consecutive words

Another thing we can do is to count the frequency of two words occurring consecutively.

word	a	all	and	bought	cat	caught	crooked	found	he	house	in	little	lived	man	mile	mouse	sixpence	stile	there	they	together	upon	walked	was	who
a							6					1													
all													1												
and								1												1					
bought	1																								
cat																									1
caught	1																								
crooked					1					1				1	1	1	1	1							
found	1																								
he				1																					
house																									
in	1																								
little							1																		
lived																					1				
man																									1
mile			1																						
mouse			1																						
sixpence																						1			
stile									1																
there																								1	
they		1																							
together											1														
upon	1																								
walked	1																								
was	1																								
who						1																		1	

Predictive n-Gram Word Model

If we normalise so that the sum of the entries in each row is equal to 1, we can use this matrix to estimate the probability prob($w_j|w_i$) of word w_j occurring after w_i

word	a	all	and	bought	cat	caught	crooked	found	he	house	in	little	lived	man	mile	mouse	sixpence	stile	there	they	together	upon	walked	was	who
a							$\frac{6}{7}$					$\frac{1}{7}$													
all													1												
and								$\frac{1}{2}$												$\frac{1}{2}$					
bought	1																								
cat																									1
caught	1																								
crooked					$\frac{1}{7}$					$\frac{1}{7}$				$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{7}$							
found	1																								
he				1																					
house																									
in	1																								
little							1																		
lived																					1				
man																									1
mile			1																						
mouse			1																						
sixpence																						1			
stile									1																
there																								1	
they		1																							
together												1													
upon	1																								
walked	1																								
was	1																								
who						$\frac{1}{2}$																		$\frac{1}{2}$	

In practice, we need to aggregate over a large corpus, so that unusual words like 'crooked' will not dominate.

The above model captures some common combinations like “there was”, “man who”, “and found”, “he bought”, “who caught”, “and they”, “they all”, “lived together”, etc.

This **unigram** model can be generalised to a bi-gram, tri-gram, . . . , n-gram model by considering the n preceding words. If the vocabulary is large, we need some tricks to avoid exponential use of memory.

Unigram Text Generator

Here is an example of text generated from a unigram model based on the frequencies of word pairs in Shakespeare's Hamlet.

“Rashly – Good night is very liberal – it is easily said there is – gyved to a sore distraction in wrath and with my king may choose but none of shapes and editing by this , and shows a sea And what this is miching malhecho; And gins to me a pass , Transports his wit , Hamlet , my arms against the mind impatient , by the conditions that would fain know ; which , the wicked deed to get from a deed to your tutor .”

We see that the local structure looks plausible, but the global structure seems like gibberish.

Co-occurrence Matrix

Sometimes, we don’t necessarily predict the next word, but simply a “nearby word” (e.g. a word occurring within an n-word window centered on that word).

For this case, we can build a matrix in which each row represents a word, and each column a nearby word.

Co-occurrence Matrix (2-word window)

word	a	all	and	bought	cat	caught	crooked	found	he	house	in	little	lived	man	mile	mouse	sixpence	stile	there	they	together	upon	walked	was	who
a				1		1	6	1			1	1										1	1	1	
all													1							1					
and								1							1	1				1					
bought	1								1																
cat							1																		1
caught	1																								1
crooked	6				1					1		1		1	1	1	1	1							
found	1		1																						
he				1														1							
house							1																		
in	1																				1				
little	1						1																		
lived		1																			1				
man							1																		1
mile			1				1																		
mouse			1				1																		
sixpence							1															1			
stile							1		1																
there																								1	
they		1	1																						
together											1		1												
upon	1																1								
walked	1																								1
was	1																		1						
who					1	1								1									1		

Co-occurrence Matrix (10-word window)

word	a	all	and	bought	cat	caught	crooked	found	he	house	in	little	lived	man	mile	mouse	sixpence	stile	there	they	together	upon	walked	was	who
a	10	2	3	2	2	2	13	3	2	1	1	1	1	2	2	1	2	2	1	2	1	2	2	1	4
all	2		1				1				1	1	1			1				1	1				
and	3	1				1	3	1			1		1		1	1	1			1	1	1	1		2
bought	2				1	1	2		1									1				1			1
cat	2			1		1	2		1							1		1							1
caught	2		1	1	1		2									1				1					1
crooked	13	1	3	2	2	2	10	2	2	1	1	1	2	2	2	1	2	3	1	1	1	2	2	1	4
found	3		1				2								1		1					1	1		
he	2			1	1		2										1	1				1			1
house	1						1				1	1									1				
in	1	1	1				1			1		1	1							1	1				
little	1	1					1			1	1		1								1				
lived	1	1	1				2				1	1				1				1	1				
man	2						2								1				1				1	1	1
mile	2		1				2	1						1			1						1		1
mouse	1	1	1		1	1	1						1							1	1				1
sixpence	2		1				2	1	1						1			1				1			
stile	2			1	1		3		1								1					1			
there	1						1							1										1	1
they	2	1	1			1	1				1		1			1					1				
together	1	1	1				1			1	1	1	1			1				1					
upon	2		1	1			2	1	1								1	1							
walked	2		1				2	1						1	1									1	1
was	1						1							1					1				1		1
who	4		2	1	1	1	4		1					1	1	1			1				1	1	

By aggregating over many documents, pairs (or groups) of words emerge which tend to occur near each other (but not necessarily consecutively). For example:

- 'cat', 'caught', 'mouse'
- 'walked', 'mile'
- 'little', 'house'

This analysis raises a number of questions:

1. Common words tend to dominate the matrix. Could we sample common words less often, in order to reveal the relationships of less common words?
2. Each row of this matrix could be considered as a vector representation for the corresponding word, but the number of dimensions is equal to the size of the vocabulary, which could be very large ($\sim 10^5$). Is there a way to reduce the dimensionality while still preserving the relationships between words?

Optional video

Singular Value Decomposition

Word Vectors

"Words that are used and occur in the same contexts tend to purport similar meanings." Z. Harris (1954)

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

We would like to assign a vector to each word or document, in such a way that words with nearby representations are likely to occur in similar contexts (or, similar documents have close vectors).

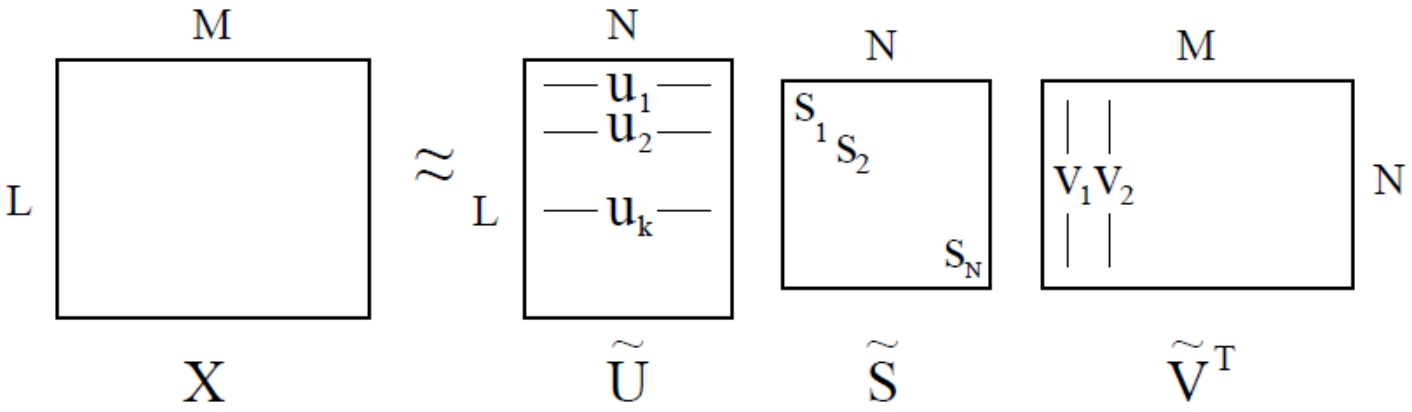
Each column in the co-occurrence matrix could be considered as a vector representation of a word or document. However, the number of dimensions in this vector would be very large (equal to the size of the vocabulary, which is typically about 60,000).

Singular Value Decomposition (SVD) is a way of projecting these vectors from 60,000 dimensions to about 200 dimensions, in such a way that the relationship between words (or documents) is preserved. SVD itself is too computationally expensive, so methods like Word2Vec and Glove are introduced which provide a good approximation to SVD with reasonable computing resources.

Singular Value Decomposition

Let X be a (co-)occurence matrix where each row represents a word, and each column represents either a word (in which case L=M) or a document (in which case L, M might not be equal).

This matrix $\mathrm{X}_{(\mathrm{L}\times\mathrm{M})}$ can be decomposed as $\mathrm{X} = \mathrm{USV}^{\mathrm{T}}$ where $\mathrm{U}_{(\mathrm{L}\times\mathrm{L})}$, $\mathrm{V}_{(\mathrm{M}\times\mathrm{M})}$ are unitary (all columns have unit length) and $\mathrm{S}_{(\mathrm{L}\times\mathrm{M})}$ is diagonal, with diagonal entries $s_1 \geq s_2 \geq \ldots \geq s_M \geq 0$



We can obtain an approximation for X of rank $\mathrm{N} < \mathrm{M}$ by truncating U to $\tilde{\mathrm{U}}_{(\mathrm{L}\times\mathrm{N})}$, S to $\tilde{\mathrm{S}}_{(\mathrm{N}\times\mathrm{N})}$ and

V to $\tilde{\mathrm{V}}_{(\mathrm{M}\times\mathrm{N})}$. The kth row of $\tilde{\mathrm{U}}$ then provides an N-dimensional vector representing the k^{th} word in the vocabulary. The kth column of $\tilde{\mathrm{V}}^{\mathrm{T}}$ also provides an N-dimensional vector, which can be seen either as a representation for documents, or as an alternative representation for words, depending on the origin of the (co-)occurrence matrix.

Word2vec and GloVe

Typically, L is the number of words in the vocabulary (about 60,000) and M is either equal to L or, in the case of document classification, the number of documents in the collection. SVD is computationally expensive, proportional to $\mathrm{L} \times \mathrm{M}^2$ if $\mathrm{L} \geq \mathrm{M}$.

word2vec (Mikolov, 2013) and GloVe (Pennington, 2014) are two common methods for computing an approximation to SVD incrementally, with less computation. **word2vec** is a predictive model which aims to maximise the probability of a word, based on the surrounding words. **GloVe** is a count-based model which tries to directly reconstruct a close approximation to the co-occurrence matrix X.

Singular Value compared to Eigenvalue Decomposition

Those familiar with Eigenvalue Decomposition will see that it is similar in some ways to Singular Value Decomposition. But, we must note a number of important differences, as illustrated in these examples:

Eigenvalue Decomposition:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \Omega\mathrm{D}\Omega^{-1}, \quad \text{where} \quad \Omega = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \mathrm{D} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} = \Omega\mathrm{D}\Omega^{-1}, \quad \text{where} \quad \Omega = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ -i & i \end{bmatrix}, \quad \mathrm{D} = \begin{bmatrix} i & 0 \\ 0 & -i \end{bmatrix}$$

Singular Value Decomposition:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \mathrm{USV}^{\mathrm{T}}, \quad \mathrm{U} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \mathrm{S} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathrm{V} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

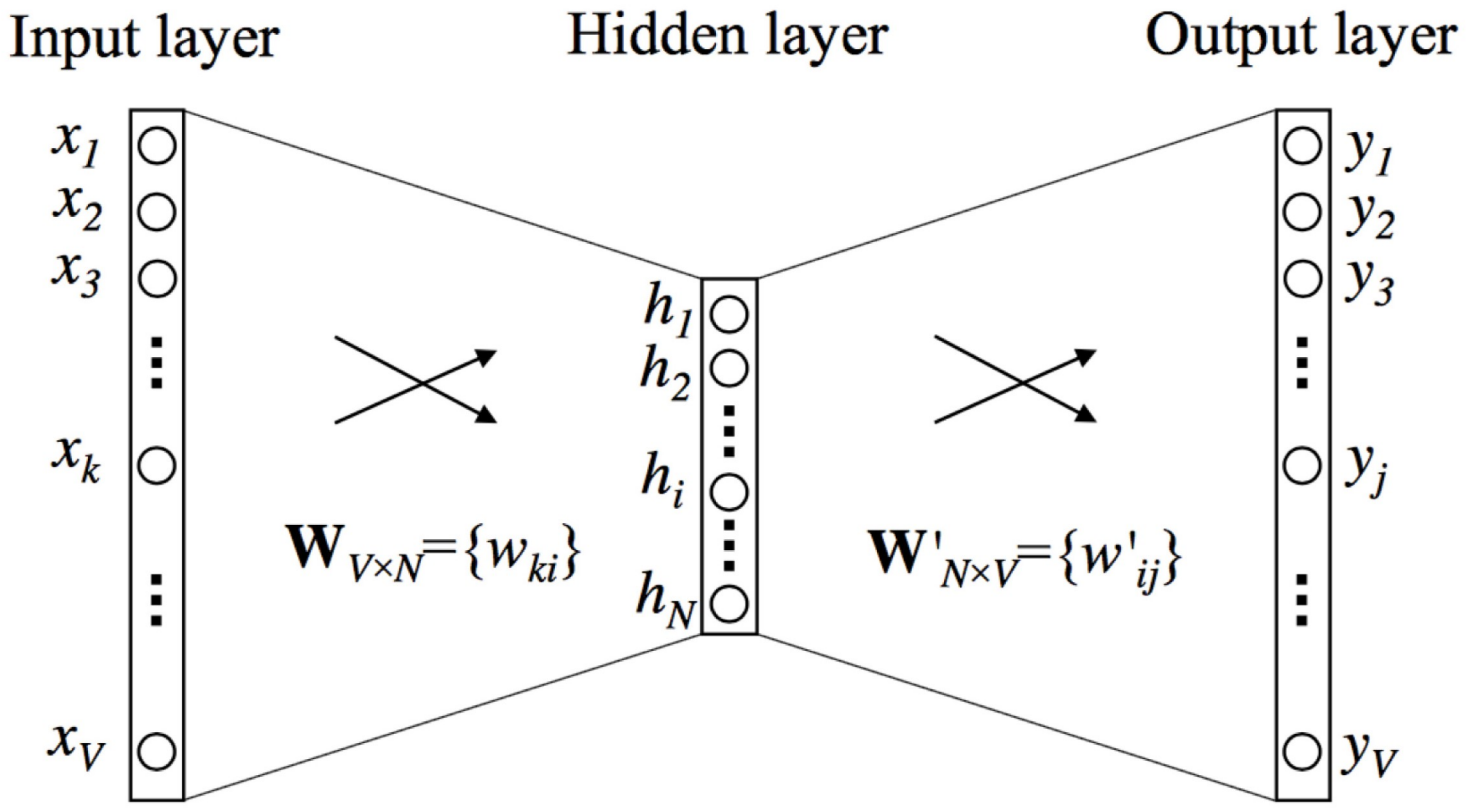
$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} = \mathrm{USV}^{\mathrm{T}}, \quad \mathrm{U} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \mathrm{S} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathrm{V} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

In general, eigenvalues can be negative or even complex, but singular values are always real and non-negative. Even if X is a square matrix, singular value decomposition treats the source and target as two entirely different spaces. The only case where the two decompositions are the same is when X is symmetric and positive semi-definite. The word co-occurrence matrix is symmetric but not positive semi-definite. For example, if the text consisted entirely of two alternating letters

..ABABABABABABAB.. then A would be the context for B, and vice-versa.

Optional video

Word2Vec



[source: Rong, 2014]

The k^{th} row $\mathbf{v}_k$ of $\mathbf{W}$ is a representation of word k.

The j^{th} column $\mathbf{v}'_j$ of $\mathbf{W}'$ is an (alternative) representation of word j.

If the (1-hot) input is k, the linear sum at each output will be $u_j = \mathbf{v}'^{\mathrm{T}}_j \mathbf{v}_k$

Note that word2vec is a linear model in the sense that there is no activation function at the hidden nodes.

Cost function

Softmax can be used to turn these linear sums u_j into a probability distribution estimating the probability of word j occurring in the context of word k

$$\mathrm{prob}(j \mid k) = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u_{j'})} = \frac{\exp\left(\mathbf{v}'^{\mathrm{T}}_j \mathbf{v}_k\right)}{\sum_{j'=1}^{V} \exp\left(\mathbf{v}'^{\mathrm{T}}_{j'} \mathbf{v}_k\right)}$$

We can treat the text as a sequence of numbers $w_1, w_2, ..., w_T$ where $w_i = j$ means that the i^{th}

word in the text is the j^{th} word in the vocabulary. We then seek to maximize the log probability

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq r\leq c, r\neq 0}\log \text{prob}\left(w_{t+r}\mid w_t\right)$$

where c is the size of training context (which may depend on w_t).

word2vec differentials

If we assume the probabilities are computed by full softmax, and the correct output is j^*, then the cost function is

$$E=-u_{j^*}+\log\sum_{j'=1}^{V}\exp\left(u_{j'}\right)$$

the output differentials are

$$e_j=\frac{\partial E}{\partial u_j}=-\delta_{jj^*}+\frac{\partial}{\partial u_j}\log\sum_{j'=1}^{V}\exp\left(u_{j'}\right)$$

where

$$\delta_{jj^*}=\begin{cases}1, & \text{if} \quad j=j^*\\ 0, & \text{otherwise}\end{cases}$$

word2vec weight updates

hidden-to-output differentials

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

hidden unit differentials

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

input-to-hidden differentials

$$\frac{\partial E}{\partial w_{ki}}=\frac{\partial E}{\partial h_i}\frac{\partial h_i}{\partial w_{ki}}=\sum_{j=1}^{V}e_j w'_{ij}x_k$$

Continuous Bag of Words and Skip-Gram

word2vec can be implemented either as Continuous Bag of Words (CBOW) or Skip-Gram, as illustrated in these figures from [Rong, 2014].

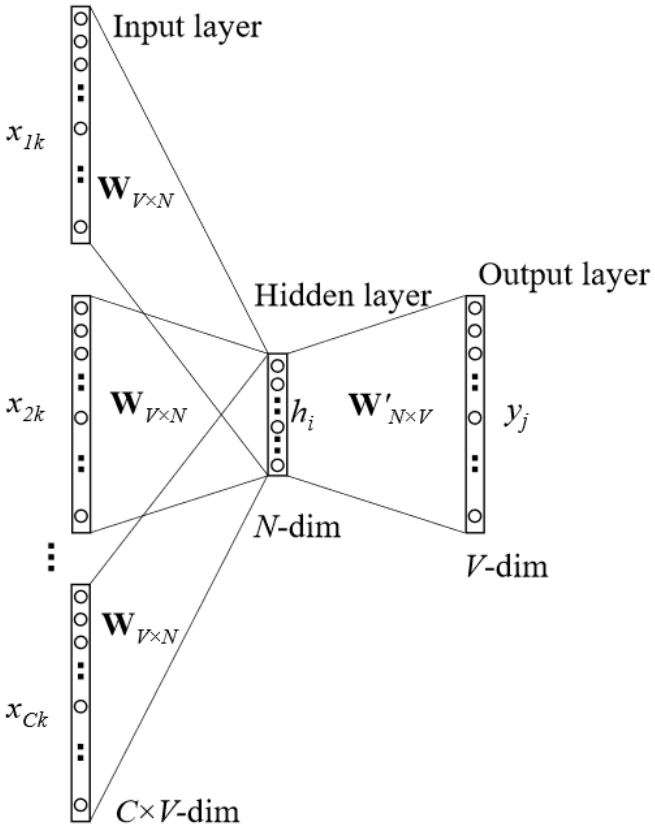


Figure 2: Continuous bag-of-word model

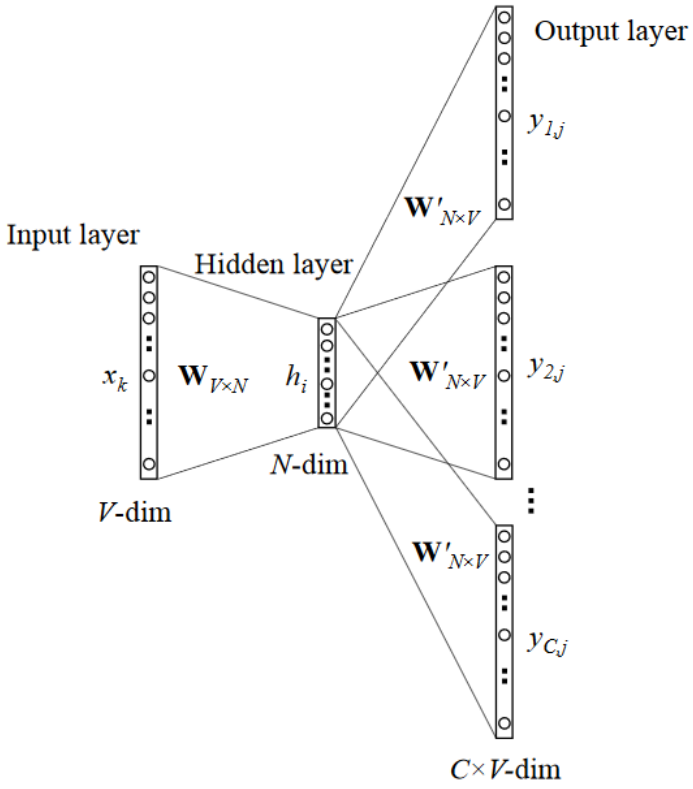


Figure 3: The skip-gram model

For CBOW, several context words are each used independently to predict the centre word, and the hidden activation becomes a sum (or average) over all the context words.

Note the difference between this and NetTalk – in word2vec (CBOW) all context words share the same input-to-hidden weights.

The Skip-Gram version tries to predict the context words, given the centre word. This is similar to CBOW, except that in this case a single input word is used to predict multiple context words, and all context words share the same hidden-to-output weights.

Efficiency issues

Depending on the computational resources available, it may be prohibitively expensive to use the full softmax, which involves computing output values for all 60,000 words in the vocabulary. Hirerarchical Softmax and negative sampling are two alternatives which are computationally more

efficient.

Hierarchical Softmax

For Hierarchical Softmax, the words are arranged in a Huffmann coded tree according to their frequency. One network output (with sigmoid activation) is assigned to each node in the tree, and the output corresponding to each node in the path from the root to the actual word w_j is trained to produce either 0 or 1 depending on whether the left or right branch should be taken. In this way, the total number of outputs is the same, but if the size of the vocabulary is L then only $\log_2(L)$ outputs need to be evaluated, on average, for each word in the training text.

Negative sampling

The idea of negative sampling is that we train the network to increase its estimation of the target word j^* and reduce its estimate not of all the words in the entire vocabulary but instead just a subset of them W_{neg}, drawn from an appropriate distribution.

$$E = -\log \sigma\left({\mathbf{v}'_{j*}}^{\mathrm{T}}\mathbf{h}\right) - \sum_{j \in \mathcal{W}_{\text{neg}}} \log \sigma\left(-{\mathbf{v}'_{j}}^{\mathrm{T}}\mathbf{h}\right)$$

This is a simplified version of Noise Contrastive Estimation (NCE). It is not guaranteed to produce a well-defined probability distribution, but in practice it does produce high-quality word embeddings.

The number of samples is 5-20 for small datasets, 2-5 for large datasets.

Empirically, a good choice of the distribution from which to draw the negative samples is $P(w) = U(w)^{3/4}/Z$ where $U(w)$ is the unigram distribution determined by the previous word, and Z is a normalizing constant.

Subsampling of frequent words

In order to diminish the influence of more frequent words, each word in the corpus is discarded with probability

$$P\left(w_i\right) = 1 - \sqrt{\frac{t}{f\left(w_i\right)}}$$

where $f(w_i)$ is the frequency of word w_i and $t \sim 10^{-5}$ is an empirically determined threshold.

Optional video

References

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient estimation of word representations in vector space*. *arXiv*: 1301.3781.

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Conference on empirical methods in natural language processing* (EMNLP) (pp. 1532–1543).

Xin Rong. (2014). *word2vec parameter learning explained*. *arXiv*: 1411.2738.

Resources

https://nlp.stanford.edu/projects/glove/

Word Relationships

Sentence Completion Task

One way to test the quality of the word embeddings is by using them to answer sentence completion tasks from standardised tests such as the Scholastic Aptitude Test (SAT).

Q1. Seeing the pictures of our old home made me feel _____ and nostalgic.

$$\begin{aligned}&\mathrm{A.\ fastidious}\\&\mathrm{B.\ indignant}\\&\mathrm{C.\ wistful}\\&\mathrm{D.\ conciliatory}\end{aligned}$$

This kind of question can be answered by feeding the surrounding words into the CBOW version of the network and seeing which of $\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}$ is assigned the highest probability by the network.

Q2. Because the House had the votes to override a presidential veto, the President has no choice but to _____ .

$$\begin{aligned}&\mathrm{A.\ object}\\&\mathrm{B.\ abdicate}\\&\mathrm{C.\ abstain}\\&\mathrm{D.\ compromise}\end{aligned}$$

The network might have some difficulty answering questions like Q2, because of the need for logical reasoning or culturally specific knowledge.

Word Relationships

Another way to probe the word vectors is to look for linguistic regularities. For example, if we add the vectors for King and Woman and subtract the vector for Man, we get something very close to the vector for Queen, i.e.

$$\mathrm{King} + \mathrm{Woman} - \mathrm{Man} \simeq \mathrm{Queen}$$

This can be used to answer questions of the form: A is to B as C is to What? For example:

Q3. evening is to **morning** as **dinner** is to _____ .

A. breakfast
B. soup
C. coffee
D. time

Q4. **bow** is to **arrow** as _____ is to **bullet.**

A. defend
B. lead
C. shoot
D. gun

In each case, we look for the word X in the dictionary whose vector v_{X} is closest to $(v_C + v_B - v_A)$, with closeness determined by cosine similarity.

$$X = \arg\max_X \frac{(v_C + v_B - v_A)^{\mathrm{T}} v_X}{\|v_C + v_B - v_A\|}$$

Here are some examples of word relationships from (Mikolov, 2013).

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

The network generally performs very well, but occasionally makes an understandable mistake. For example, did you notice what the network thinks is the first and last name of the President of Russia?

Optional video

References

Mikolov, T., Sutskever, I., Chen, K., Corrado, D.S., & Dean, J. (2013). *Distributed representations of words and phrases and their compositionality*. NIPS: 3111-19.

Exercise - Singular Value Decomposition

Consider the sentence:

> "two flowers grew tall on two tall towers"

Question 1 *Submitted Oct 8th 2022 at 10:18:10 pm*

Write the co-occurrence matrix $\mathbf{X}$ for this sentence, using a 4-word context window (i.e. two context words on either side of the central word).

asdf

Question 2 *Submitted Oct 8th 2022 at 10:18:27 pm*

Use `torch.svd()` to compute the singular value decompositon of this matrix $\mathbf{X} = \mathbf{USV}^{\mathrm{T}}$.

asdf

Question 3 *Submitted Oct 8th 2022 at 10:20:26 pm*

Extract a word representation from the first two columns of $\mathbf{U}$ and use `matplotlib` to plot the words on a 2-dimensional graph.

asdf