Word Vectors - II

Pawan Goyal

CSE, IIT Kharagpur

January 27th, 2017

Word Vectors

- At one level, it is simply a vector of weights.

Word Vectors

- At one level, it is simply a vector of weights.
- In a simple 1-of-N (or ‘one-hot’) encoding every element in the vector is associated with a word in the vocabulary.

Word Vectors

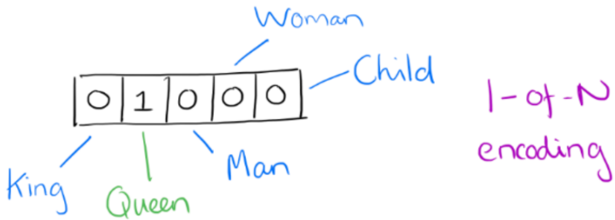
- At one level, it is simply a vector of weights.
- In a simple 1-of-N (or ‘one-hot’) encoding every element in the vector is associated with a word in the vocabulary.
- The encoding of a given word is simply the vector in which the corresponding element is set to one, and all other elements are zero.

One-hot representation

motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0

Word Vectors - One-hot Encoding

- Suppose our vocabulary has only five words: King, Queen, Man, Woman, and Child.
- We could encode the word 'Queen' as:



Limitations of One-hot encoding

Limitations of One-hot encoding

Word vectors are not comparable

Using such an encoding, there is no meaningful comparison we can make between word vectors other than equality testing.

Word2Vec – A distributed representation

Distributional representation – word embedding?

Any word w_i in the corpus is given a distributional representation by an embedding

$$w_i \in R^d$$

i.e., a $d-$dimensional vector, which is mostly learnt!

Word2Vec – A distributed representation

Distributional representation – word embedding?

Any word w_i in the corpus is given a distributional representation by an embedding

$$w_i \in R^d$$

i.e., a $d-$dimensional vector, which is mostly learnt!

$$\textit{linguistics} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

Distributional Representation

- Take a vector with several hundred dimensions (say 1000).
- Each word is represented by a distribution of weights across those elements.
- So instead of a one-to-one mapping between an element in the vector and a word, the representation of a word is spread across all of the elements in the vector, and
- Each element in the vector contributes to the definition of many words.

Distributional Representation: Illustration

If we label the dimensions in a hypothetical word vector (there are no such pre-assigned labels in the algorithm of course), it might look a bit like this:

Such a vector comes to represent in some abstract way the 'meaning' of a word

Word Embeddings

- d typically in the range 50 to 1000
- Similar words should have similar embeddings

Word Embeddings

- d typically in the range 50 to 1000
- Similar words should have similar embeddings

SVD can also be thought of as an embedding method

Reasoning with Word Vectors

- It has been found that the learned word representations in fact capture meaningful syntactic and semantic regularities in a very simple way.

Reasoning with Word Vectors

- It has been found that the learned word representations in fact capture meaningful syntactic and semantic regularities in a very simple way.
- Specifically, the regularities are observed as constant vector offsets between pairs of words sharing a particular relationship.

Reasoning with Word Vectors

- It has been found that the learned word representations in fact capture meaningful syntactic and semantic regularities in a very simple way.
- Specifically, the regularities are observed as constant vector offsets between pairs of words sharing a particular relationship.

Case of Singular-Plural Relations

If we denote the vector for word i as x_i, and focus on the singular/plural relation, we observe that

Reasoning with Word Vectors

- It has been found that the learned word representations in fact capture meaningful syntactic and semantic regularities in a very simple way.
- Specifically, the regularities are observed as constant vector offsets between pairs of words sharing a particular relationship.

Case of Singular-Plural Relations

If we denote the vector for word i as x_i, and focus on the singular/plural relation, we observe that

$$x_{apple} - x_{apples} \approx x_{car} - x_{cars} \approx x_{family} - x_{families} \approx x_{car} - x_{cars}$$

and so on.

Reasoning with Word Vectors

Perhaps more surprisingly, we find that this is also the case for a variety of semantic relations.

Good at answering analogy questions

a is to b, as c is to ?

man is to *woman* as *uncle* is to ? (*aunt*)

Reasoning with Word Vectors

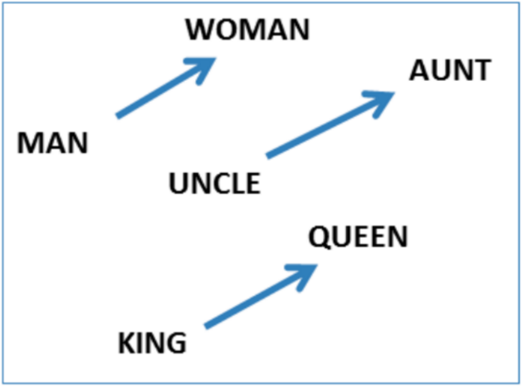
Perhaps more surprisingly, we find that this is also the case for a variety of semantic relations.

Good at answering analogy questions

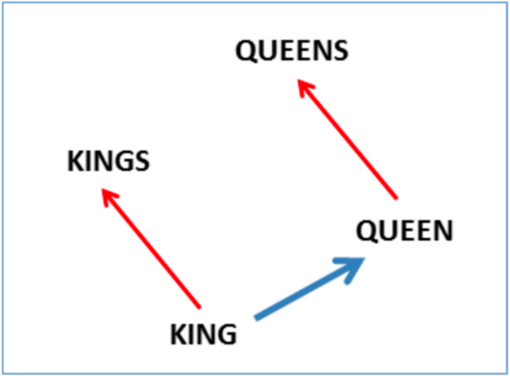
a is to b, as c is to ?
man is to *woman* as *uncle* is to ? (*aunt*)

A simple vector offset method based on cosine distance shows the relation.

Vcctor Offset for Gender Relation



Vcctor Offset for Singular-Plural Relation



Encoding Other Dimensions of Similarity

Analogy Testing

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

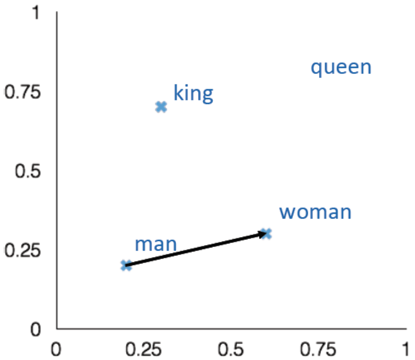
Analogy Testing

a:b :: c:?

$$d = \arg\max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}$$

man:woman :: king:?

+	king	[0.30 0.70]
-	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]



Country-capital city relationships

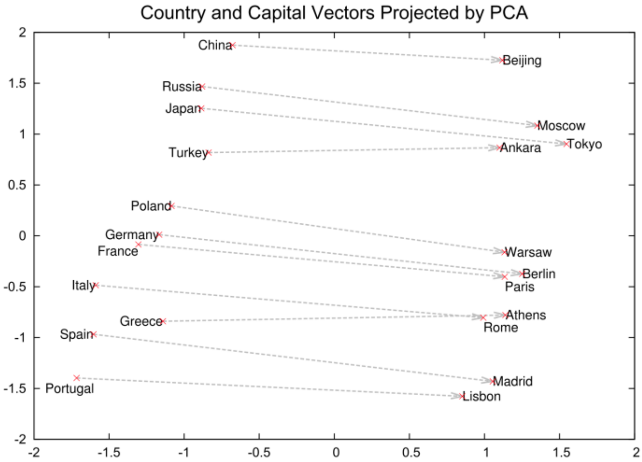


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

More Analogy Questions

Newspapers			
New York	New York Times	Baltimore	Baltimore Sun
San Jose	San Jose Mercury News	Cincinnati	Cincinnati Enquirer
NHL Teams			
Boston	Boston Bruins	Montreal	Montreal Canadiens
Phoenix	Phoenix Coyotes	Nashville	Nashville Predators
NBA Teams			
Detroit	Detroit Pistons	Toronto	Toronto Raptors
Oakland	Golden State Warriors	Memphis	Memphis Grizzlies
Airlines			
Austria	Austrian Airlines	Spain	Spainair
Belgium	Brussels Airlines	Greece	Aegean Airlines
Company executives			
Steve Ballmer	Microsoft	Larry Page	Google
Samuel J. Palmisano	IBM	Werner Vogels	Amazon

Table 2: Examples of the analogical reasoning task for phrases (the full test set has 3218 examples). The goal is to compute the fourth phrase using the first three. Our best model achieved an accuracy of 72% on this dataset.

Element Wise Addition

We can also use element-wise addition of vector elements to ask questions such as 'German + airlines' and by looking at the closest tokens to the composite vector come up with impressive answers:

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

Table 5: Vector compositionality using element-wise addition. Four closest tokens to the sum of two vectors are shown, using the best Skip-gram model.

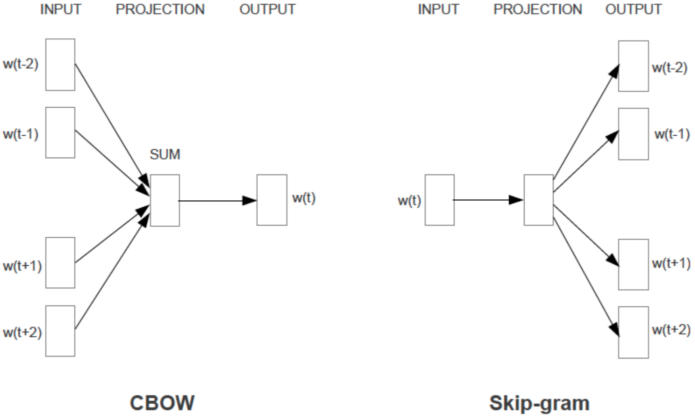
Learning Word Vectors

Basic Idea

Instead of capturing co-occurrence counts directly, predict (using) surrounding words of every word.

Code as well as word-vectors: *https://code.google.com/p/word2vec/*

Two Variations: CBOW and Skip-grams



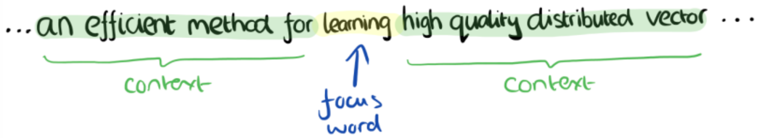
- Consider a piece of prose such as:
  "The recently introduced continuous Skip-gram model is an efficient method for learning high-quality distributed vector representations that capture a large number of syntatic and semantic word relationships."

CBOW

- Consider a piece of prose such as:
  "The recently introduced continuous Skip-gram model is an efficient method for learning high-quality distributed vector representations that capture a large number of syntatic and semantic word relationships."
- Imagine a sliding window over the text, that includes the central word currently in focus, together with the four words that precede it, and the four words that follow it:

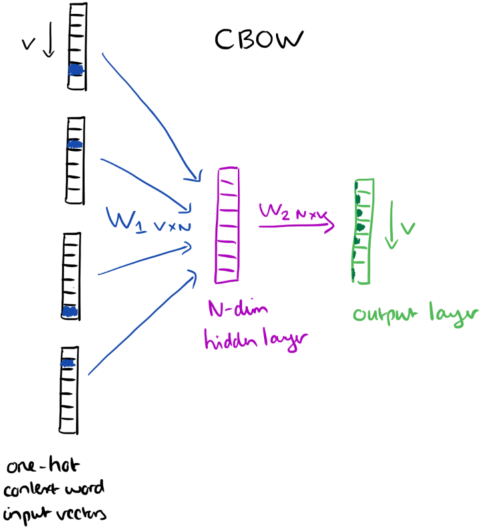
CBOW

- Consider a piece of prose such as:
  "The recently introduced continuous Skip-gram model is an efficient method for learning high-quality distributed vector representations that capture a large number of syntatic and semantic word relationships."
- Imagine a sliding window over the text, that includes the central word currently in focus, together with the four words that precede it, and the four words that follow it:



CBOW

The context words form the input layer. Each word is encoded in one-hot form. A single hidden and output layer.



CBOW: Training Objective

- The training objective is to maximize the conditional probability of observing the actual output word (the focus word) given the input context words, with regard to the weights.
- In our example, given the input (“an”, “efficient”, “method”, “for”, “high”, “quality”, “distributed”, “vector”), we want to maximize the probability of getting “learning” as the output.

CBOW: Input to Hidden Layer

Since our input vectors are one-hot, multiplying an input vector by the weight matrix W_1 amounts to simply selecting a row from W_1.

$$\begin{array}{cccc} \text{input} & W_1 & & \text{hidden} \\ 1\times V & V\times N & & 1\times N \\ \begin{bmatrix}0 & 1 & 0\end{bmatrix} & \begin{bmatrix} a & b & c & d\\ e & f & g & h\\ i & j & k & l\end{bmatrix} & = & \begin{bmatrix}e & f & g & h\end{bmatrix} \\ & W_1 & & \end{array}$$

Given C input word vectors, the activation function for the hidden layer h amounts to simply summing the corresponding 'hot' rows in W_1, and dividing by C to take their average.

CBOW: Hidden to Output Layer

From the hidden layer to the output layer, the second weight matrix W_2 can be used to compute a score for each word in the vocabulary, and softmax can be used to obtain the posterior distribution of words.