What is word2vec?

- Word2vec is a tool which computes vector representations of words.
- Word meaning and relationships between words are encoded spatially
- learns from input texts
- Developed by Mikolov, Sutskever, Chen, Corrado and Dean in 2013 at Google Research

Word Embedding &Word2Vec

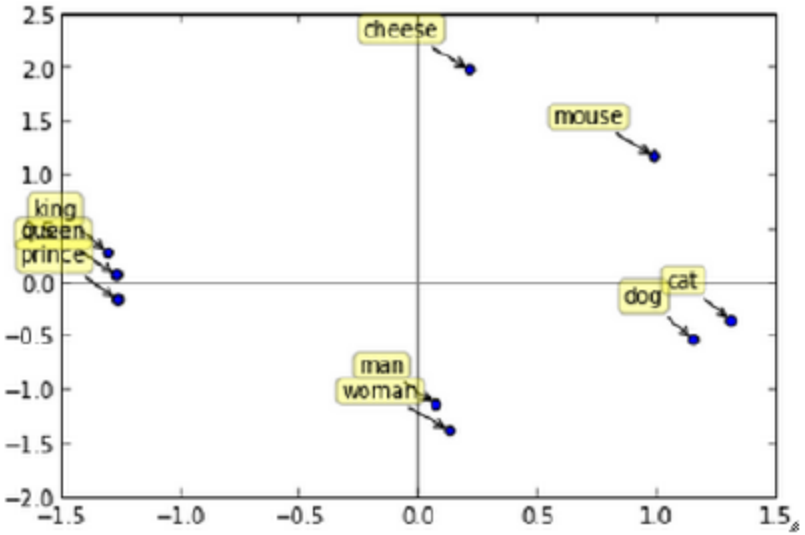
- In NLP we often map words into vectors that contains numerical value that machine can understand
- ***Word embedding*** is a type of mapping that allows the words with similar meaning to have similar vector representation.
- Word2vec is a shallow neural net that processes text.
- Its input is a text corpus and its output is a set of vectors: feature vectors for words in that corpus
- Word2vec is not a deep neural network, it turns text into a numerical form that deep nets can understand

One-Hot encoding vs Word Embedding

Traditional Method - Bag of Words Model	Word Embeddings
• Uses one hot encoding	• Stores each word in as a point in space, where it is represented by a vector of fixed number of dimensions (generally 300)
• Each word in the vocabulary is represented by one bit position in a HUGE vector.	• Unsupervised, built just by reading huge corpus
• For example, if we have a vocabulary of 10000 words, and "Hello" is the 4th word in the dictionary, it would be represented by: 0 0 0 1 0 0 . . . . . 0 0 0 0	• For example, "Hello" might be represented as : [0.4, -0.11, 0.55, 0.3 . . . 0.1, 0.02]
• Context information is not utilized	• Context information is utilized

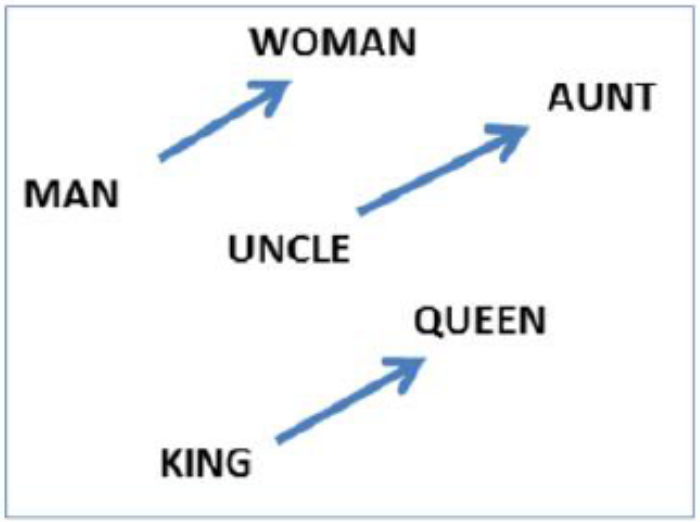
Similar words are closer together

- spatial distance corresponds to word similarity
- words are close together ⇔ their "meanings" are similar
- notation: word w -> vec[w] its point in space, as a position vector.
- e.g. vec[woman] = (0.1, -1.3)



Word relationships are displacements

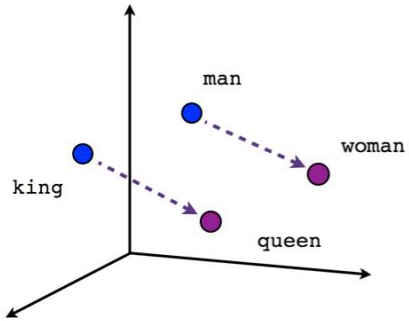
- The displacement (vector) between the points of two words represents the word relationship.
- Same word relationship => same vector



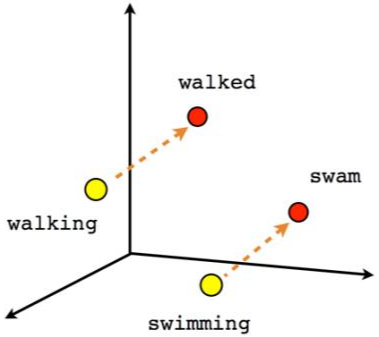
Source: *Linguistic Regularities in Continuous Space Word Representations*, Mikolov et al, 2013

- E.g. vec[queen] - vec[king] = vec[woman]- vec[man]

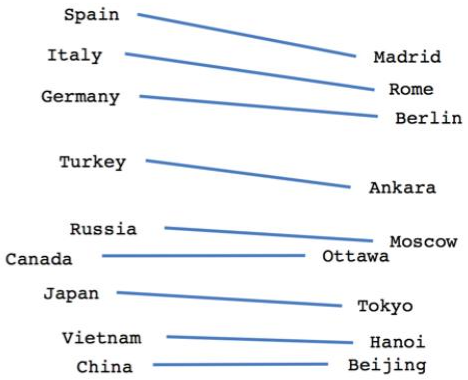
Examples



Male-Female



Verb tense



Country-Capital

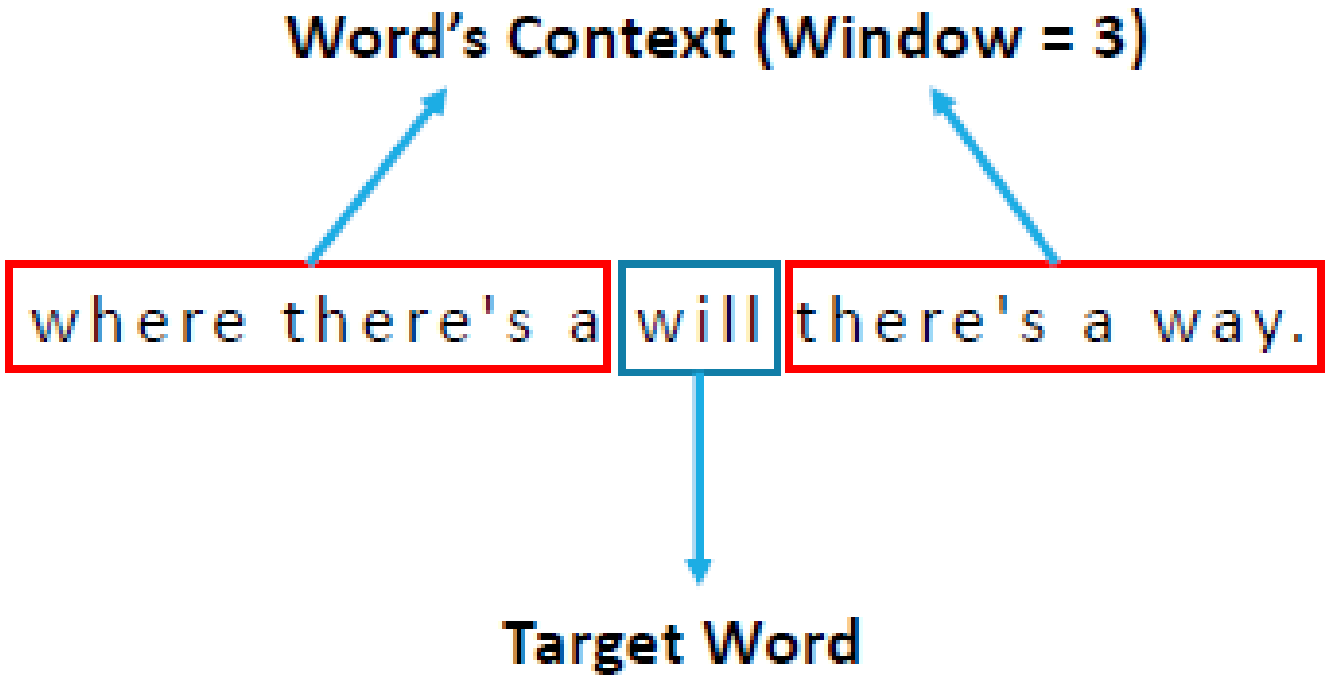
vector[Queen] = vector[King] - vector[Man] + vector[Woman]

Word Context/Meaning

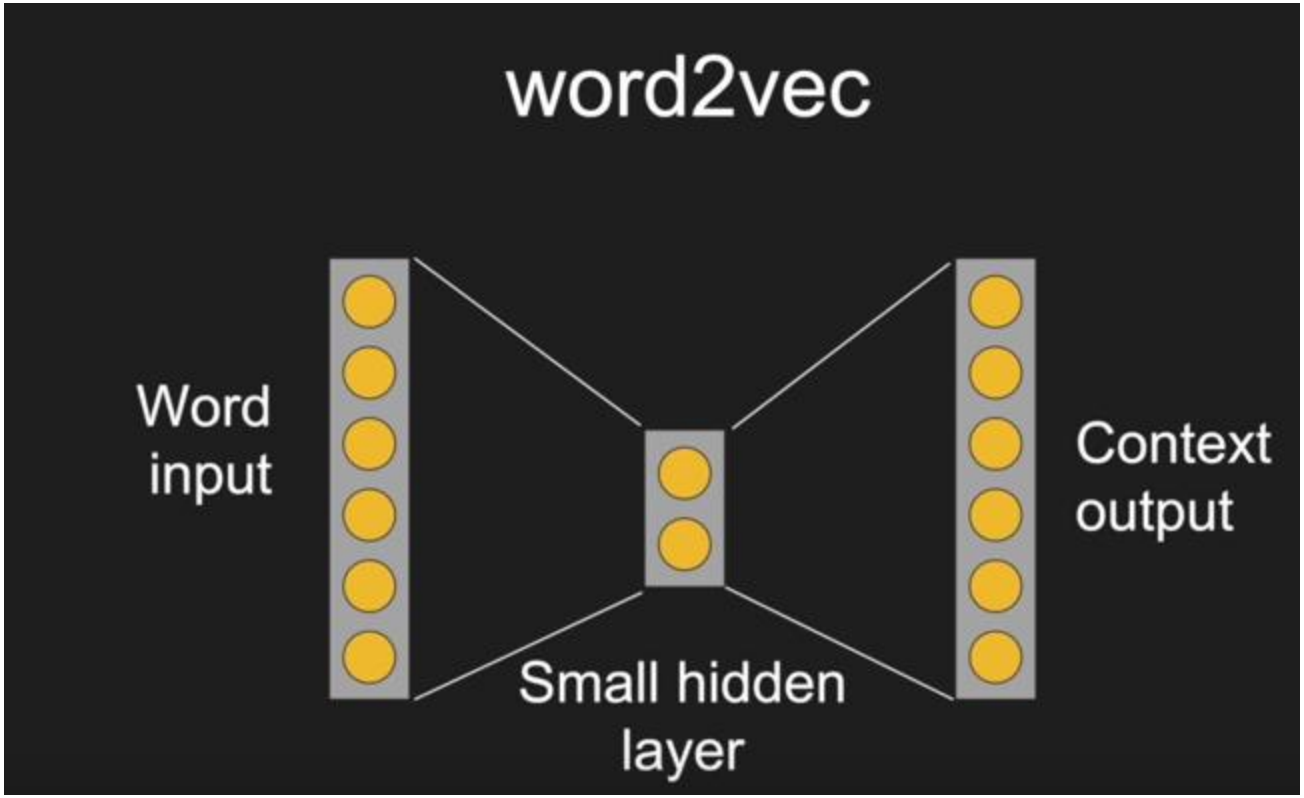
- Word is represented as continuous levels of activation
- Meaning of a word by neighbours it keeps with
- The words used in same context are related
- Word is represented by context in use
  - I eat an apple everyday
  - I eat an orange everyday
  - I like driving my car everyday

A word is known by the company it keeps

Word Context



word2vec



Characteristic

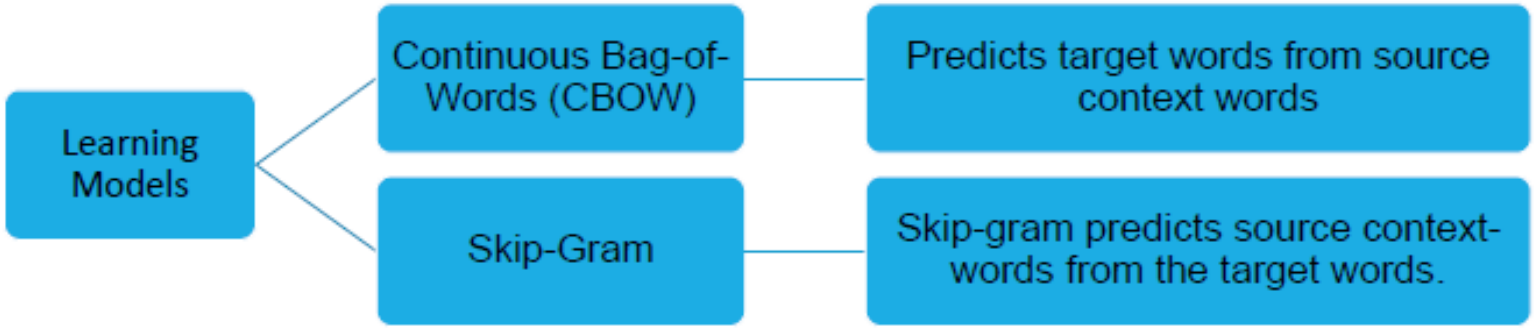
Word2vec is a computationally efficient model for learning word embeddings.

Word2vec is a successful example of “shallow” learning.

Very simple Feedforward neural network with single hidden layer, backpropagation

Supervised learning with unlabeled input data!

Learning Models

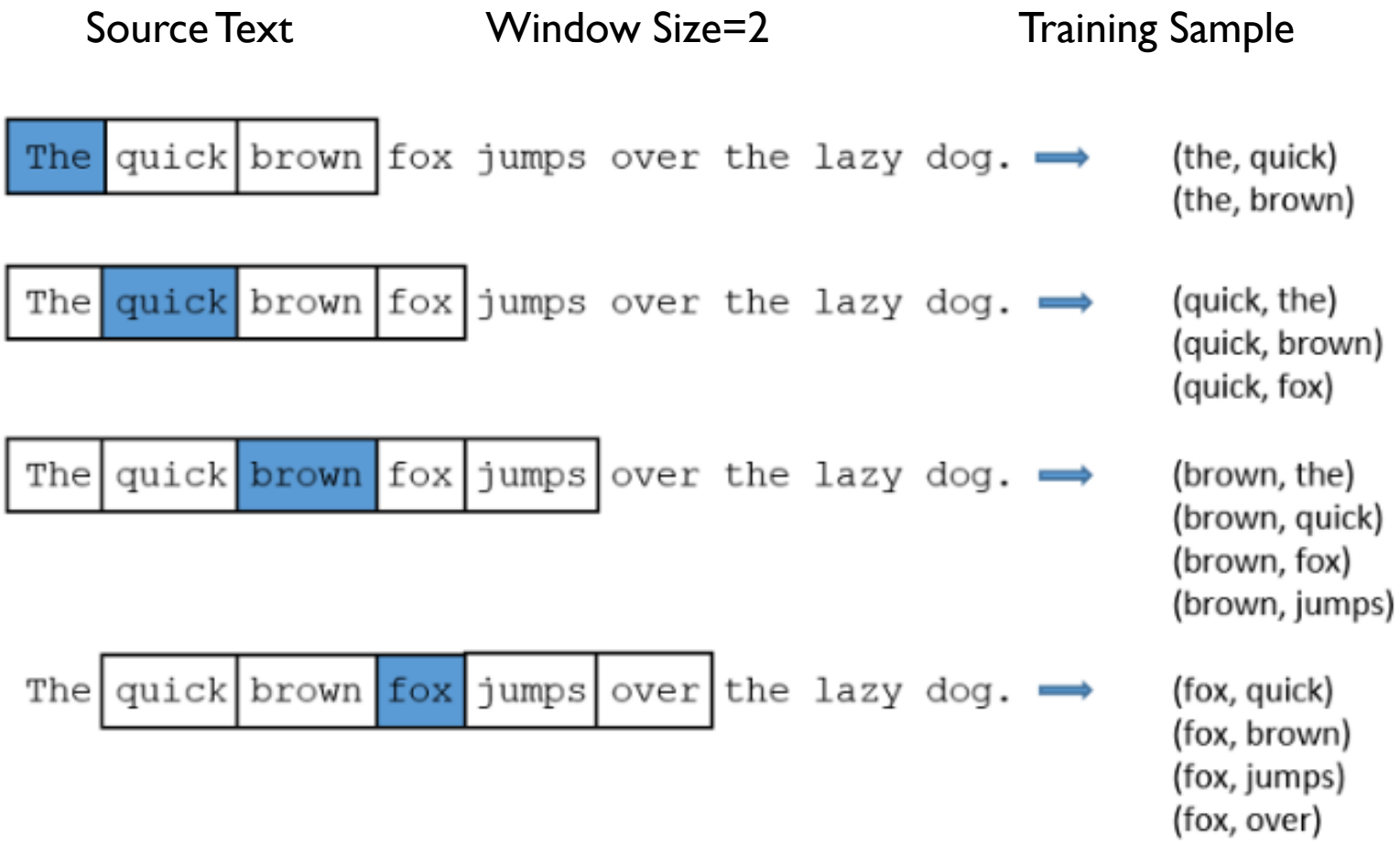


Learning Models

"the quick brown fox jumped over the lazy dog"

- define 'context' as the window of words to the left and to the right of a target word. Using a window size of 1, we then have the dataset using CBOW
- CBOW predicts target word from context word
- ([the, brown], **quick**), ([quick, fox], **brown**), ([brown, jumped], **fox**), ...
- (context, target) pairs
- *Skip-gram* inverts contexts and targets, and tries to predict each context word from its target word, so the task becomes to predict 'the' and 'brown' from 'quick', 'quick' and 'fox' from 'brown', etc. Therefore our dataset becomes (target, context)
- (**quick**, the), (**quick**, brown), (**brown**, quick), (**brown**, fox),…

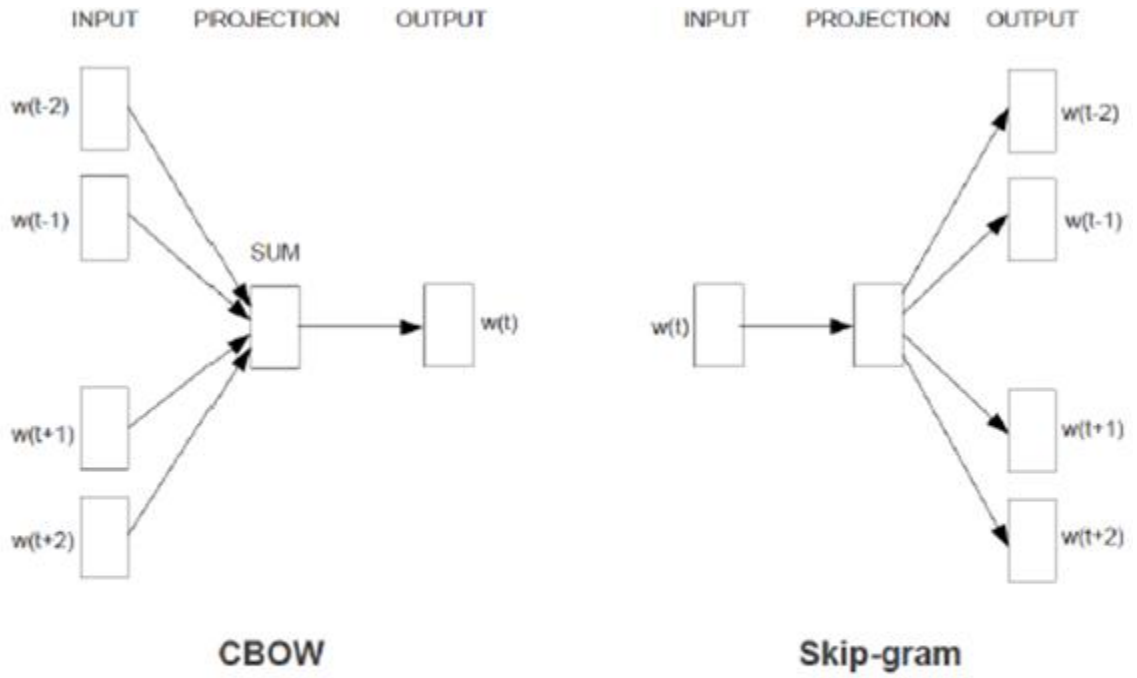
Learning Models (Skip-gram)



Learning Model

- Using context to predict a target word (a method known as continuous bag of words, or CBOW)
- Using a word to predict a target context, which is called skip-gram
- When the feature vector assigned to a word cannot be used to accurately predict that word's context, the components of the vector are adjusted.
- Each word's context in the corpus is the *teacher* sending error signals back to adjust the feature vector.
- The vectors of words judged similar by their context are nudged closer together by adjusting the numbers in the vector.
- Both the n words before and after the target word w_t to predict it

Learning Model

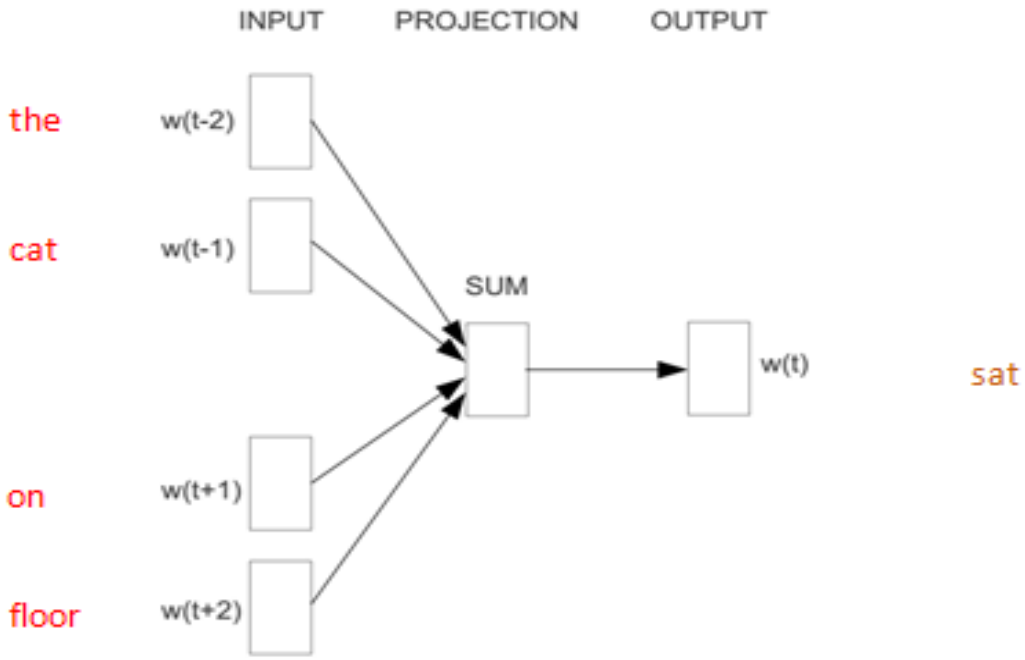


Skip-gram: works well with small amount of the training data, represents well even rare words or phrases.

CBOW: several times faster to train than the skip-gram, slightly better accuracy for the frequent words

Word2vec – Continuous Bag of Word

- E.g. "The cat sat on floor"
  - Window size = 2



Learning Models

- The output of the Word2vec neural net is a vocabulary in which each item has a vector attached to it
- Vectors can be fed into a deep-learning net or simply queried to detect relationships between words.
- The vectors we use to represent words are called *neural word embeddings*
- word2vec trains words against other words that neighbor them in the input corpus.
- **word2vec demonstrates that, for vectorial representations of words, shallow learning can give great results.**

Vector Space Model

- The purpose and usefulness of Word2vec is to group the vectors of similar words together in vector space
- It detects similarities mathematically.
- Word2vec creates vectors that are distributed numerical representations of word features
- Features such as the context of individual words. Is captured
- It does so without human intervention.
- The vector components represent weights or importance of each word in the document

Learning Model

- Word2vec uses a single hidden layer, fully connected neural network as shown below.
- The neurons in the hidden layer are all linear neurons.
- The input layer is set to have as many neurons as there are words in the vocabulary for training.
- The hidden layer size is set to the dimensionality of the resulting word vectors.
- The size of the output layer is same as the input layer.

Learning Model

- If words are used in similar context when these words will be converted to vectors (low dimensional representation) in a vector space they must be close to each other
- Similar things and ideas are shown to be “close”. Their relative meanings have been translated to measurable distance
- Qualities become quantities, and algorithms can do their work.
- But similarity is just the basis of many associations that Word2vec can learn.
- It can gauge relations between words of one language, and map them to another

Why word2vec is so powerful

- Word2vec is a powerful black box.
- Off-the-shelf end-to-end pipeline
- These vectors are the basis of a more comprehensive geometry of words.
- Not only will Rome, Paris, Berlin and Beijing cluster near each other, but they will each have similar distances in vector space to the countries whose capitals they are; i.e. Rome - Italy = Beijing - China.
- Rome was the capital of Italy, and were wondering about the capital of China, then the equation Rome -Italy + China would return Beijing

Applications

- Given enough data, usage and contexts, Word2vec can make highly accurate guesses about a word's meaning based on past appearances.
- Those guesses can be used to establish a word's association /relation with other words (e.g. "man" is to "boy" what "woman" is to "girl")
- cluster documents and classify them by topic. Those clusters can form the basis of search, sentiment analysis and recommendations in such diverse fields as scientific research, legal discovery, e-commerce and customer relationship management

Summary

Word2vec is a computationally efficient model for learning word embeddings.

Its basic idea is words in similar contexts have similar meanings.

The learned vectors can be used as input for so many NLP tasks.

There are good free implementations for it including Python's Gensim library.

References

1. http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/
2. https://iksinc.online/tag/continuous-bag-of-words-cbow/
3. http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/