Natural Language Processing Introduction

Unit 3: Distributed representations

March 2020

Word embeddings

Definition

Word embeddings serve as core representations of text in deep learning approaches. They constitute semantic representations of words.

Why are they relevant?

- It's the state of the art for complex problems

Why are they relevant?

- It's the state of the art for complex problems
- Can deal with huge amount of data and find complex patterns

Why are they relevant?

- It's the state of the art for complex problems
- Can deal with huge amount of data and find complex patterns
- In some tasks can achieve super human performance

Why are they relevant?

- It's the state of the art for complex problems
- Can deal with huge amount of data and find complex patterns
- In some tasks can achieve super human performance
- Can model complex data using sequence modeling.

Why are they relevant?

- It's the state of the art for complex problems
- Can deal with huge amount of data and find complex patterns
- In some tasks can achieve super human performance
- Can model complex data using sequence modeling.
- Can perform memory tasks by design adding memory cells to the network.

Distributional semantics

Distributional semantics is a subfield of natural language processing predicated on the idea that word meaning is derived from its usage. The distributional hypothesis states that words used in similar contexts have similar meanings. That is, if two words often occur with the same set of words, then they are semantically similar in meaning.

Distributional semantics

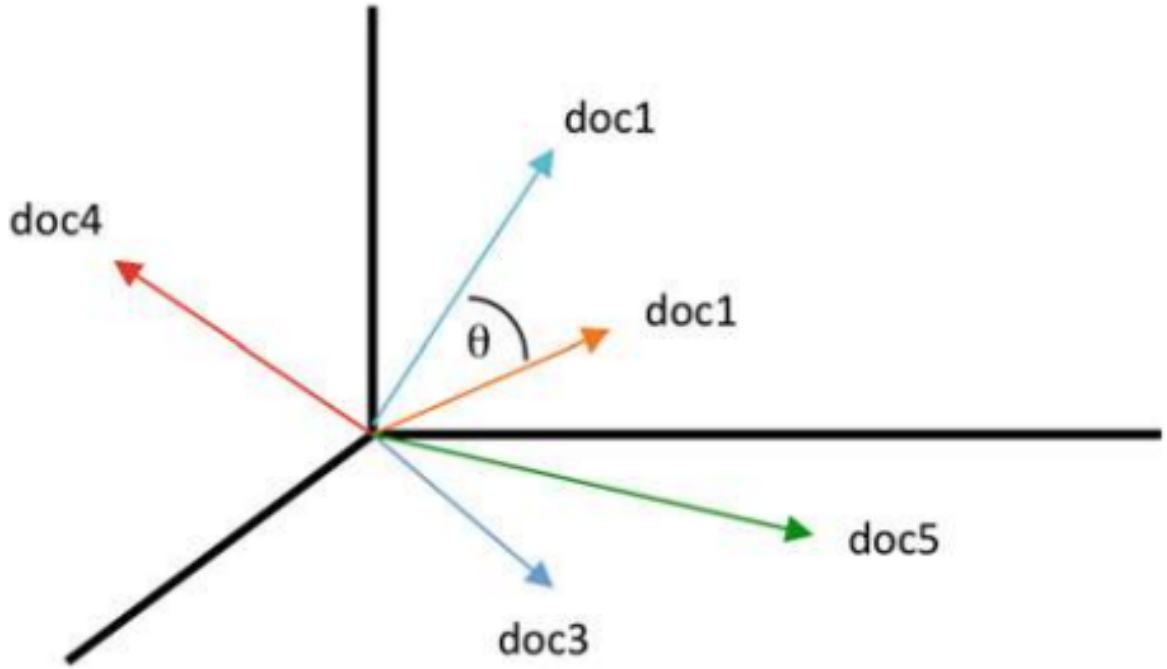
A broader notion is the statistical semantic hypothesis, which states that meaning can be derived from statistical patterns of word usage. Distributional semantics serve as the fundamental basis for many recent computational linguistic advances.

Vector space model

Vector space models (VSMs) represent a collection of documents as points in a hyperspace, or equivalently, as vectors in a vector space. They are based on the key property that the proximity of points in the hyperspace is a measure of the semantic similarlity of the documents. In other words, documents with similar vector representations imply that they are semantically similar. VSMs have found widespread adoption in information retrieval applications, where a search query is achieved by returning a set of nearby documents sorted by distance.

Vector space model

We have already seen VSMs in the form of the bag-of-words term-frequency or TFIDF example:



Word representations

One of the earliest use of word representations dates back to 1986. Word vectors explicitly encode linguistic regularities and patterns. Distributional semantic models can be divided into two classes, co-occurrence based and predictive models. Co-occurrence based models must be trained over the entire corpus and capture global dependencies and context, while predictive models capture local dependencies within a (small) context window.

Word representations

The most well-known of these models, word2vec and GloVe, are known as word models since they model word dependencies across a corpus. Both learn high-quality, dense word representations from large amounts of unstructured text data. These word vectors are able to encode linguistic regularities and semantic patterns, which lead to some interesting algebraic properties.

Co ocurrence

The distributional hypothesis tells us that co-occurrence of words can reveal much about their semantic proximity and meaning. Computational linguistics leverages this fact and uses the frequency of two words occurring alongside each other within a corpus to identify word relationships. Pointwise Mutual Information (PMI) is a commonly used information theoretic measure of co-occurrence between two words w_1 and w_2:

$$PMI(w_1, w_2) = log \frac{p(w_1, w_2)}{p(w_1)p(w_2)}$$

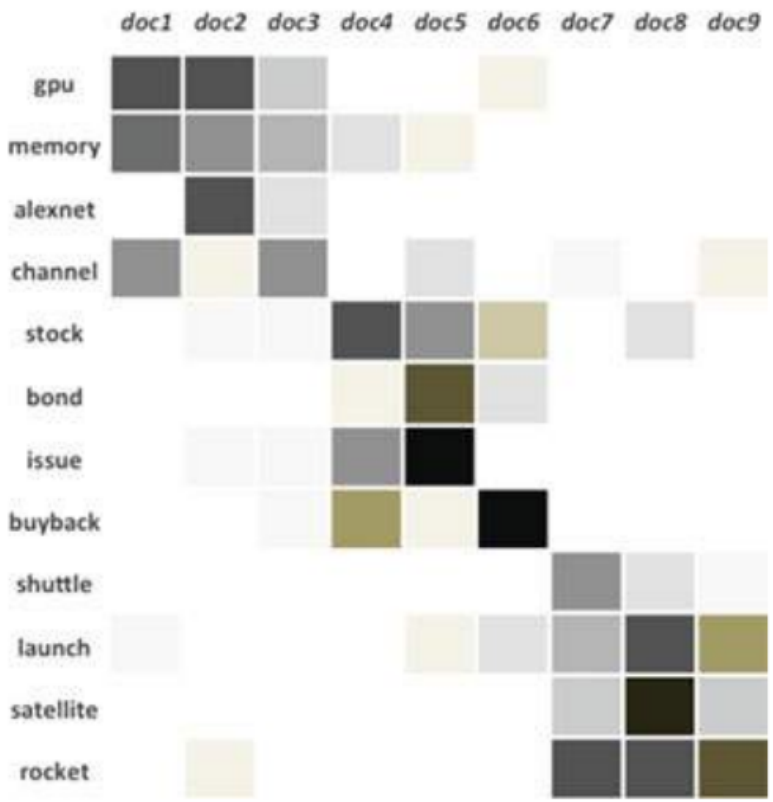
Co ocurrence

Where $p(w)$ is the probability of the word occurring, and $p(w_1, w_2)$ is joint probability of the two words co-occurring. High values of PMI indicate collocation and coincidence (and therefore strong association) between the words. It is common to estimate the single and joint probabilities based on word frequency and co-occurrence within the corpus. PMI is a useful measure for word clustering and many other tasks.

Latent semantic analysis

Latent semantic analysis (LSA) is a technique that effectively leverages word co occurrence to identify topics within a set of documents. Specifically, LSA analyzes word associations within a set of documents by forming a document-term matrix, where each cell can be the frequency of occurrence or TFIDF of a term within a document. As this matrix can be very large (with as many rows as words in the vocabulary of the corpus), a dimensionality reduction technique such as singular-value decomposition is applied to find a low-rank approximation. This low-rank space can be used to identify key terms and cluster documents or for information retrieval

Latent semantic analysis



LSA document term matrix

Neural language models

Recall that language models seek to learn the joint probability function of sequences of words. As stated above, this is difficult due to the curse of dimensionality, the size of the vocabulary used in the English language implies that there could be an impossibly huge number of sequences over which we seek to learn. A language model estimates the conditional probability of the next word w_T given all previous words w_t:

$$p(w_t) = \prod_{t=1}^{T} p(w_t | w_1, \cdots, w_{t-1})$$

Neural language models

Many methods exist for estimating continuous representations of words, including latent semantic analysis (LSA) and latent Dirichlet allocation (LDA). The former fails to preserve linear linguistic regularities while the latter requires huge computational expense for anything beyond small datasets. In recent years, different neural network approaches have been proposed to overcome these issues, which we introduce below. The representations learned by these neural network models are termed neural embeddings or simply embeddings.

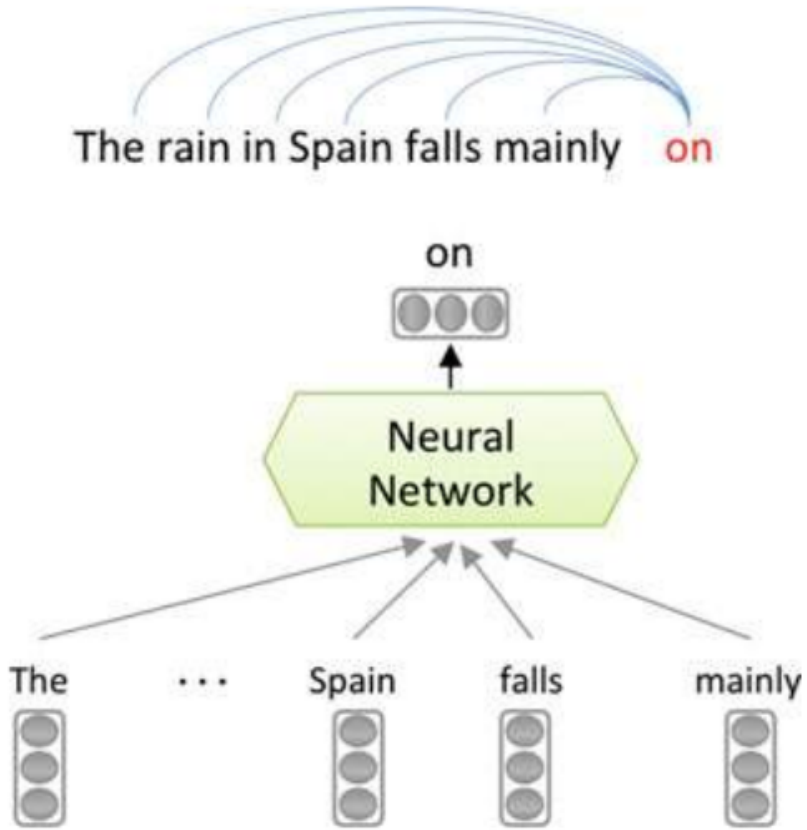
Bengio

Turing award winners 2019

Bengio

In 2003, Bengio et al. presented the paper "A neural probabilistic language model" where described a probabilistic model for learning a distributed representation of words. Instead of sparse, high-dimensional representations, the Bengio model proposed representing words and documents in lower-dimensional continuous vector spaces by using a multilayer neural network

Bengio



Neural language model

Bengio

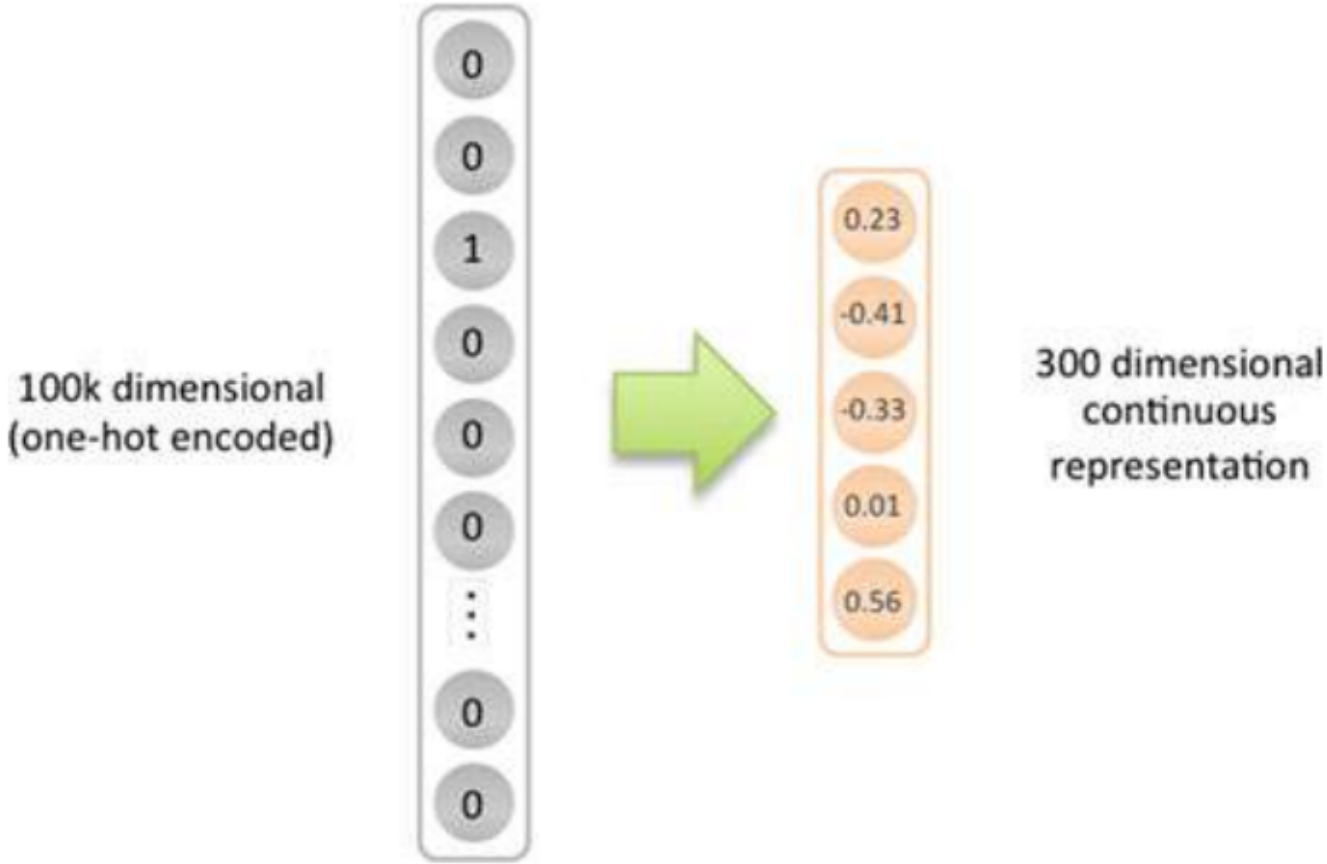
To predict the next word given the previous ones. This network is iteratively trained to maximize the conditional log-likelihood J over the training corpus using back propagation:

$$J = \frac{1}{T}\sum_{t=1}^{T} logf(v(w_t), v(w_{t-1}), \cdots, v(w_{t-n+1}); \theta) + R(\theta)$$

Neural language model

Where $v(w_t)$ is the feature vector for word w_t , f is the mapping function representing the neural network, and $R(\theta)$ is the regularization penalty applied to weights θ of the network. In doing so, the model concurrently associates each word with a distributed word feature vector as well as learning the joint probability function of word sequences in terms of the feature vectors of the words in the sequence. For instance, with a corpus of vocabulary size of 100,000, a one-hot encoded 100,000 dimensional vector representation, the Bengio model can learn a much smaller 300 dimensional continuous vector space representation.

Sparse vs. dense representations



Sparse vs. dense representations

Neural models

Neural language models can be trained by stochastic gradient descent and thereby avoid the heavy computational and memory burden of storing co occurrence matrices in memory.

word2vec

In 2013, Mikolov proposed a set of neural architectures could compute continuous representations of words over large datasets. Unlike other neural network architectures for learning word vectors, these architectures were highly computationally efficient, able to handle even billion-word vocabularies, since they do not involve dense matrix multiplications.

word2vec

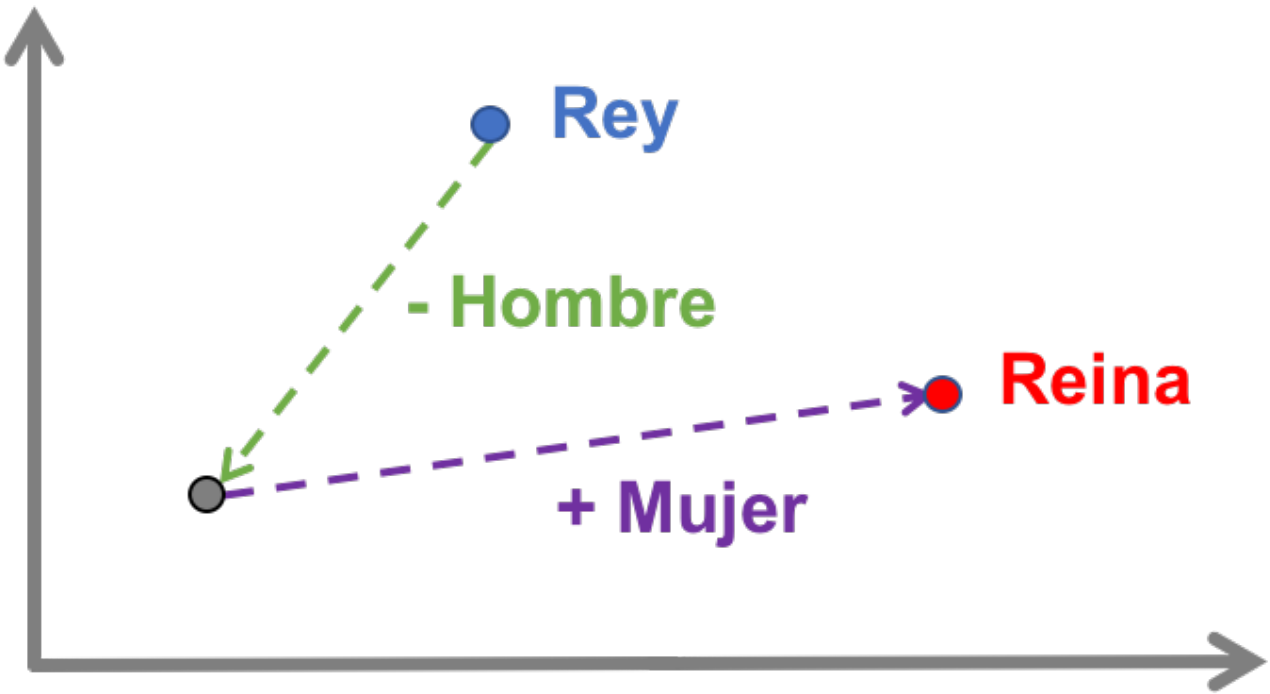
The proposed architectures consisted of the continuous bag-of-words (CBOW) model and the skip-gram model. They termed the group of models word2vec. They also proposed two methods to train the models based on a hierarchical softmax approach or a negative-sampling approach.

word2vec

The translational properties of the vectors learned through word2vec models can provide highly useful linguistic and relational similarities. In particular, Mikolov et al. revealed that vector arithmetic can yield high-quality word similarities and analogies.

word2vec

They showed that the vector representation of the word queen can be recovered from representations of king, man, and woman by searching for the nearest vector based on cosine distance to the vector sum:



CBOW

The CBOW architecture is based on a projection layer that is trained to predict a target word given a context window of c words to the left and right side of the target word. The input layer maps each context word through an embedding matrix W to a dense vector representation of dimension k, and the resulting vectors of the context words are averaged across each dimension to yield a single vector of k dimension.

CBOW

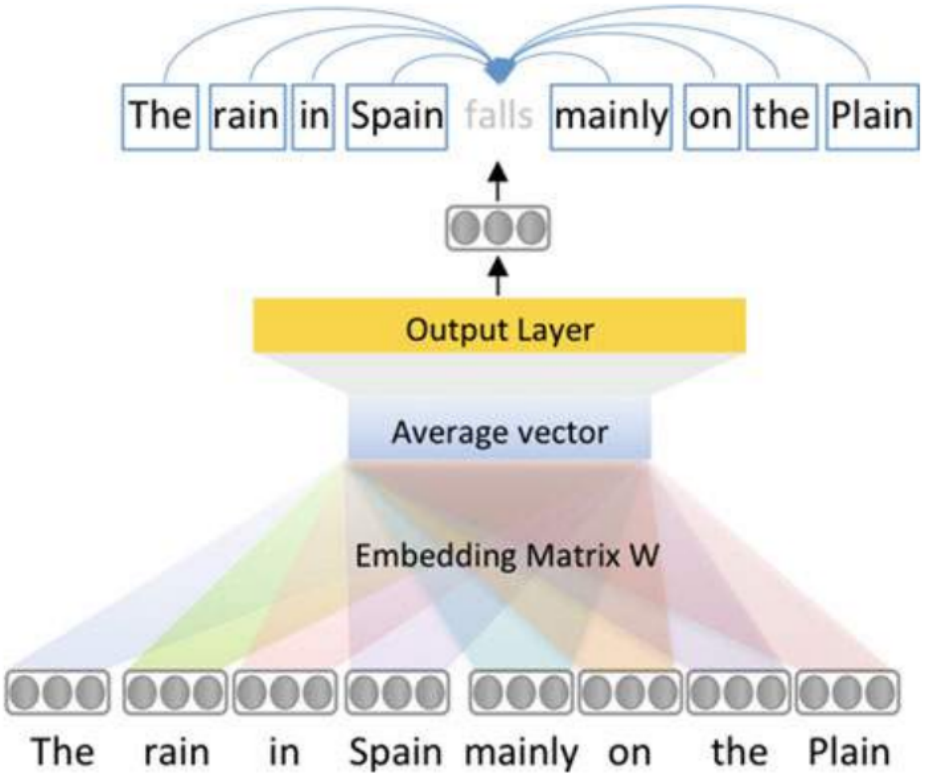
The CBOW model objective seeks to maximize the average log probability:

$$\frac{1}{T} \sum_{t=1-c<j<c, j\neq 0}^{T} \sum log(p(w_t|w_{t+j}))$$

where c is the number of context words to each side of the target word. For the simple CBOW model, the average vector representation from the output of the projection layer is fed into a softmax that predicts over the entire vocabulary of the corpus, using backpropagation to maximize the log probability objective:

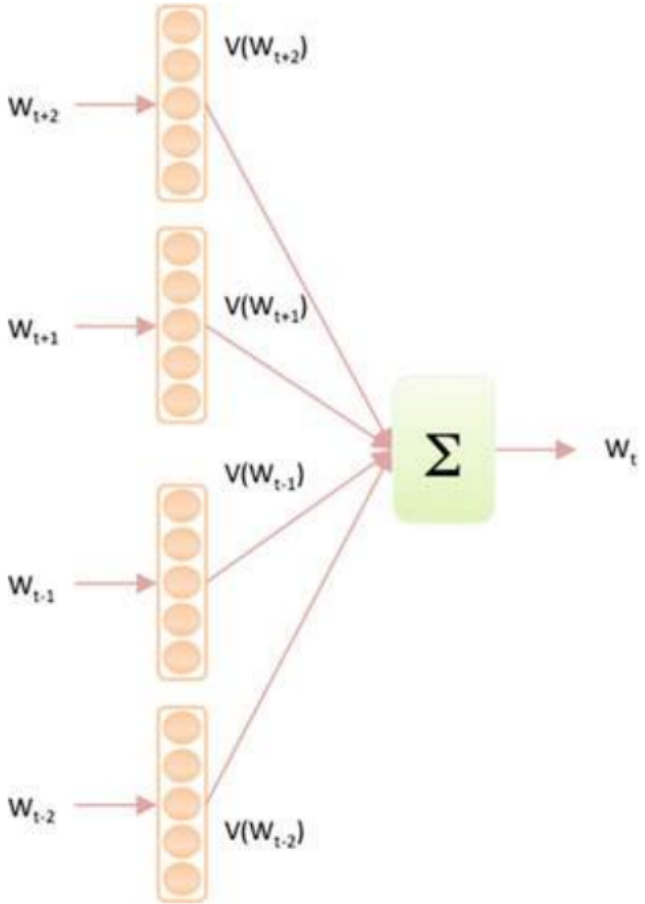
$$p(w_t|w_{t+j}) = \frac{exp(v'_{w_t} v^T_{w_{t+j}})}{\sum_{w=1}^{V} exp(v'_{w_t} v^T_{w_{t+j}})}$$

CBOW



Continuous bag of words model (context window = 4)

CBOW



CBOW vector construction (context window = 2)

Skip gram

Whereas the CBOW model is trained to predict a target word based on the nearby context words, the skip-gram model is trained to predict the nearby context words based on the target word. Once again, word order is not considered. For a context size c, the skip-gram model is trained to predict the c words around the target word. The objective of the skip-gram model is to maximize the average log.

Skip gram

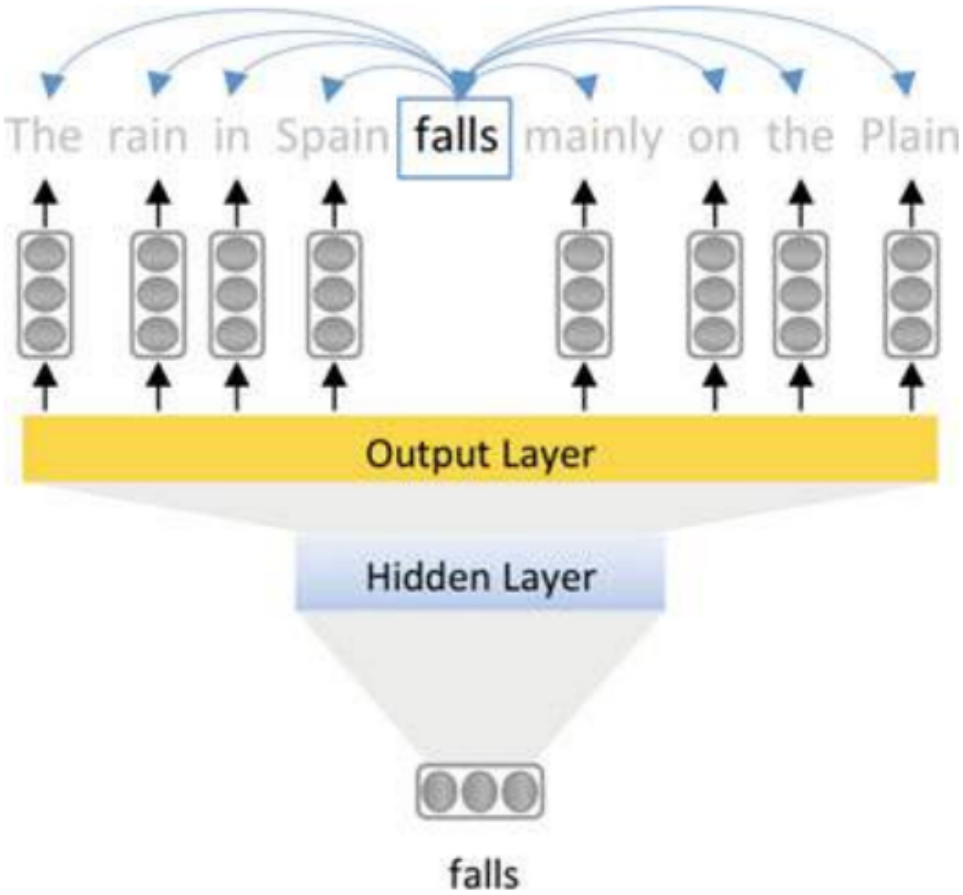
The skip gram model objective seeks to maximize the average log probability:

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{-c<j<c, j\neq 0} log(p(w_{t+j}|w_t))$$

where c is the number of context words to each side of the target word. For the simple CBOW model, the average vector representation from the output of the projection layer is fed into a softmax that predicts over the entire vocabulary of the corpus, using backpropagation to maximize the log probability objective:

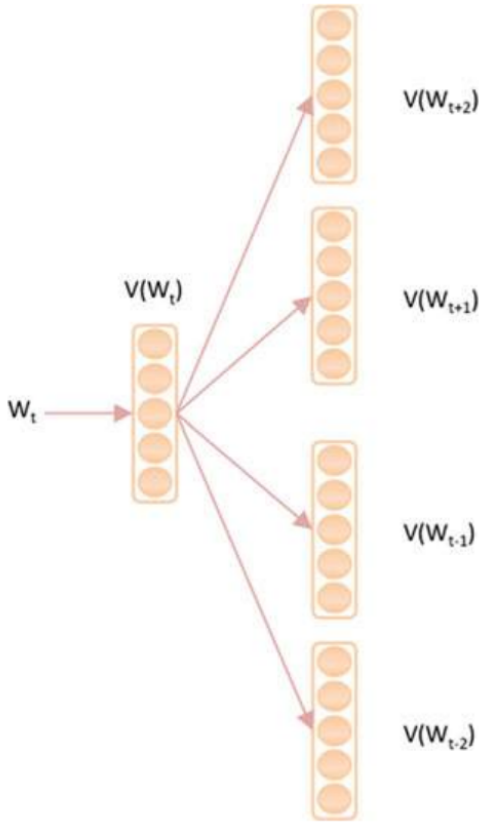
$$p(w_t|w_{t+j}) = \frac{exp(v'_{w_t+j} v^T_{w_t})}{\sum_{w=1}^{V} exp(v'_{w_t} v^T_{w_t})}$$

Skip gram



Skip gram model (context window = 4)

Skip gram



Skip gram vector construction (context window = 2)

Let's code

References

[1] Jurafsky D., Martin J.: Speech and Language Processing 2nd. ed. (2009).

[2] Bird S., Klein E., Loper, E.: Natural Language Processing with Python, (2009). ISBN: 978-0-596-51649-9

[3] Indurkhya N., Damerau F. Handbook of Natural Language Processing, Second Edition (2010). ISBN: 978-1-4200-8593-8

[4] Kao, A., Poteet S. Natural Language Processing and Text Mining (2007). ISBN: 78-1-84628-175-4

[5] Sipser, M. Introduction to the theory of computation (2013). ISBN: 978-1-133-18779-0

[6] https://www.sketchengine.eu/corpora-and-languages/corpus-types/