Word embeddings

Prof. Luca Cagliero
Dipartimento di Automatica e Informatica
Politecnico di Torino

Politecnico
di Torino

Lecture goal

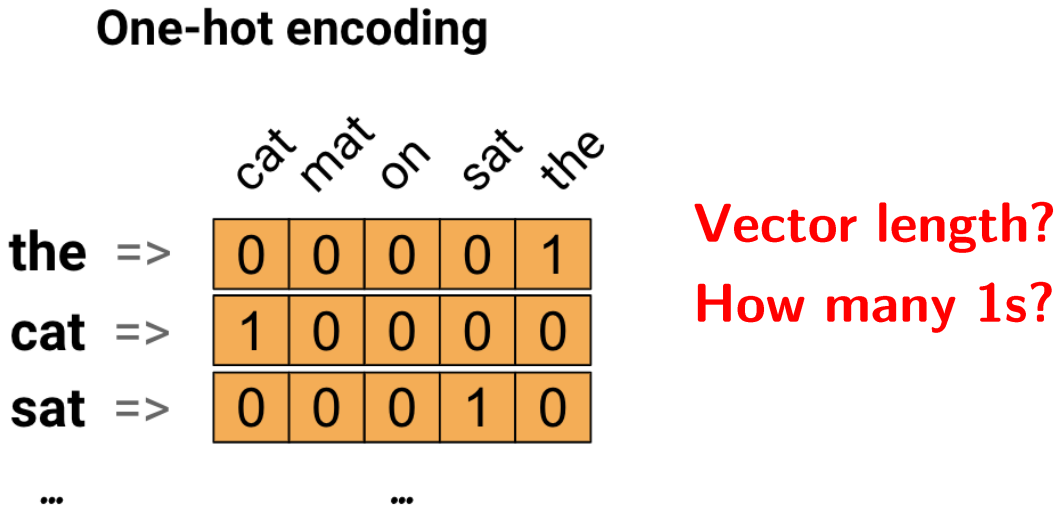
- Introduction to vector representations of text
  - Distributional hypothesis
- Word-level embedding models
  - Word2Vec, FastText, GloVe
- Word embedding visualization

Occurrence-based text representations

- Feature-value data representation
- Each feature is mapped to a specific textual unit
  - Word, n-gram, multiword group, entity, paragraph, ...
- Samples are documents or portions of documents
  - E.g., sections, paragraphs, sentences
- The encoded information is derived from the frequency of occurrence of the textual units in the documents
  - E.g., word-document representation

One-hot encoding

- Each word in the dictionary is regarded as discrete symbol and mapped to a distinct feature
- Each document is represented as a tuple in a relation whose schema consists of the word-level set of features
- The feature takes binary values (0/1) indicating whether a particular word occurs in the given document

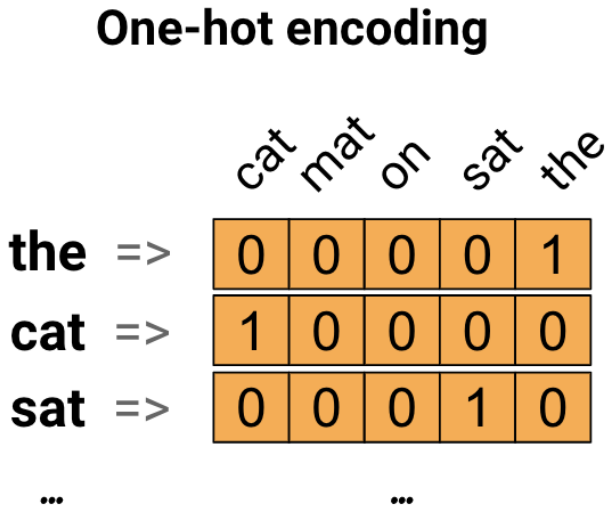


Vector length?

How many 1s?

One-hot encoding

- Each word in the dictionary is regarded as discrete symbol and mapped to a distinct feature
- Each document is represented as a tuple in a relation whose schema consists of the word-level set of features
- The feature takes binary values (0/1) indicating whether a particular word occurs in the given document



Vector length? 5

It is equal to the vocabulary size.

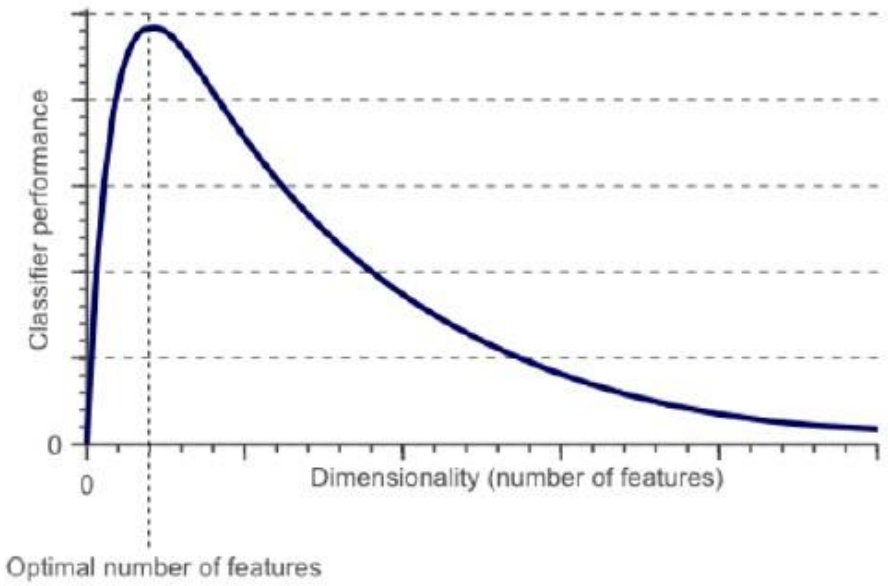
How many 1s?

Just one corresponding to the index of the word in the vocabulary.

One-hot encoding: drawbacks

- High dimensionality
  - Number of words in the dictionary >> Number of documents
- Data sparsity
  - Each document is likely to include a limited number of words in the dictionary
- Inability to encode semantic relations

Curse of dimensionality



Inability to encode semantic relations

- Suitable for evaluating syntactical text similarities
- Unsuitable for capturing semantic text similarities

$$king = [0000000000\mathbf{1}0000]$$

$$queen = [0000000\mathbf{1}0000000]$$

Inability to encode semantic relations

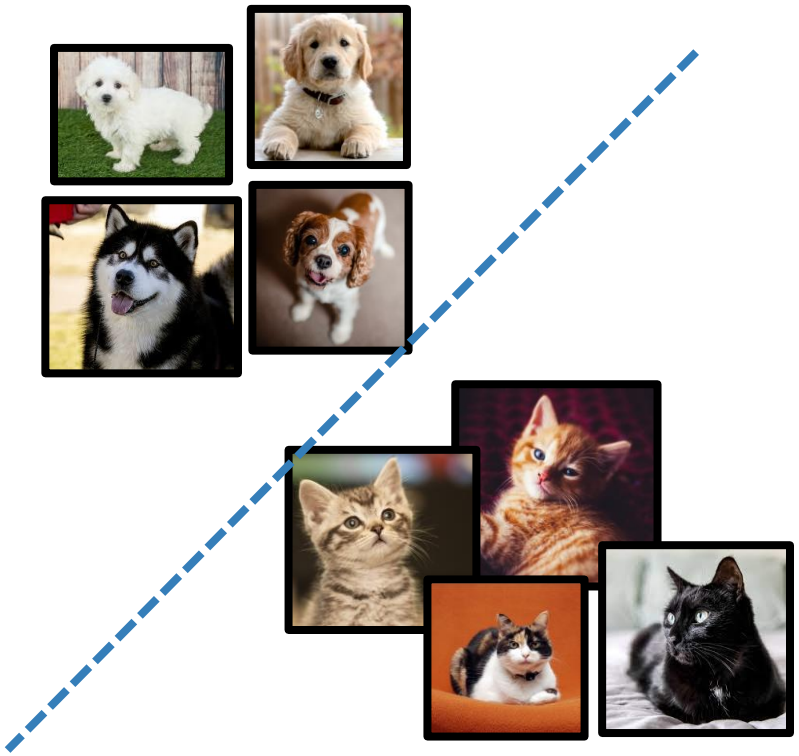
- Vectors corresponding to semantically related text snippets are orthogonal
- Need for ad hoc semantic models to capture semantic models
  - Neither efficient nor flexible

$$king = [0000000000\mathbf{1}0000]$$

$$queen = [0000000\mathbf{1}0000000]$$

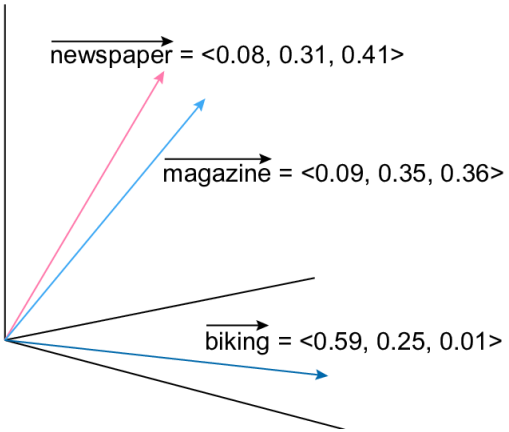
Representation learning

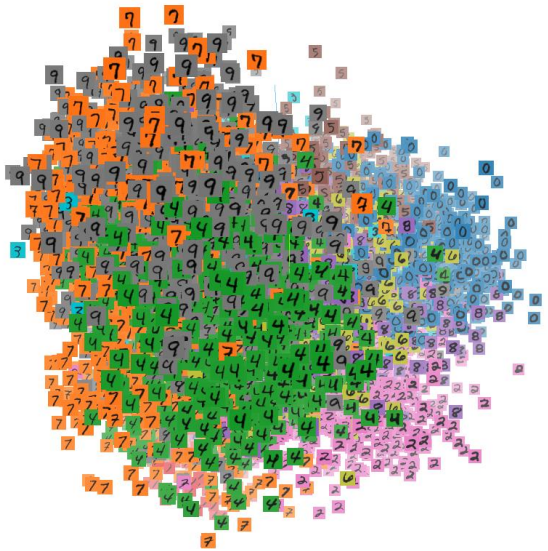
- Has the goal of learning general representation from raw data
- Each sample is mapped to a data point that can semantically represent the input



Representation learning

- Automatic learning of actionable text representations
  - The obtained representations captures the underlying text semantics
- Opposite of feature engineering
- Directly modeling data distribution



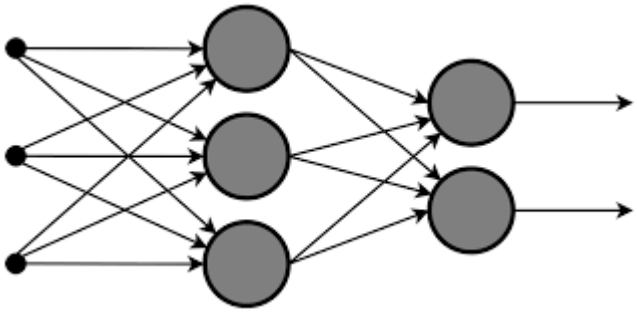


Kenter, Tom. (2017). Text Understanding for Computers.

https://projector.tensorflow.org/

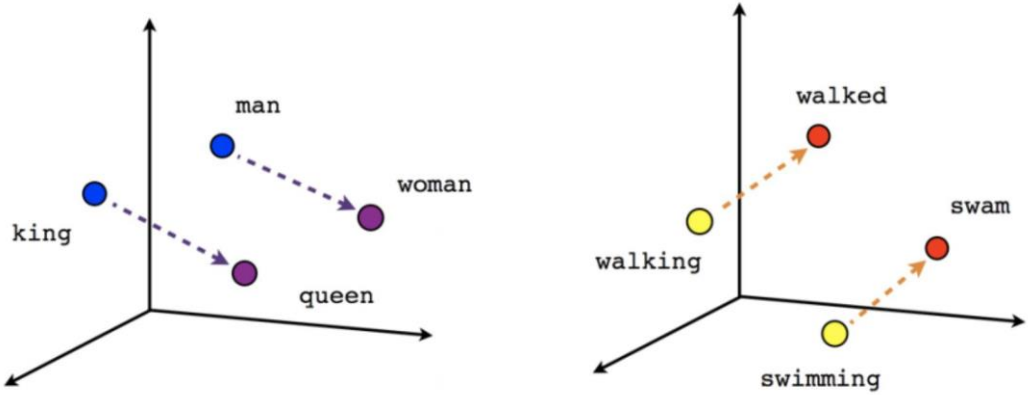
Distributed representations of text

- High-dimensional text representation
  - Extracted by means of Neural Networks



Distributed representations of text

- Dense representation
  - Concepts are no longer localized in a single feature
  - They are distributed across multiple features
- We can learn new concepts without adding new features
  - Derived from the latent space

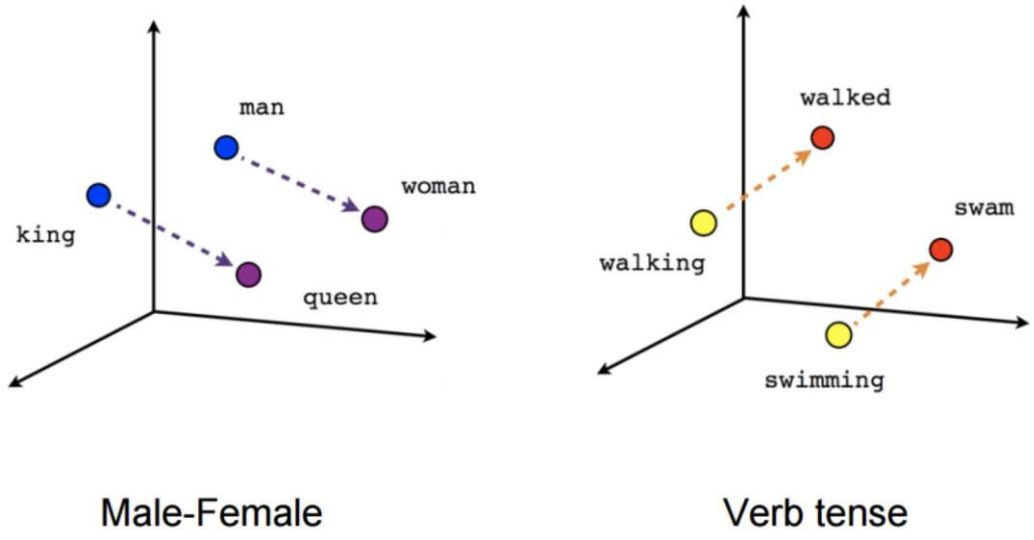


Male-Female

Verb tense

Distributed representations of text

- Features are not interpretable
  - They do not correspond to any textual unit
- They take continuous values instead of 0s and 1s

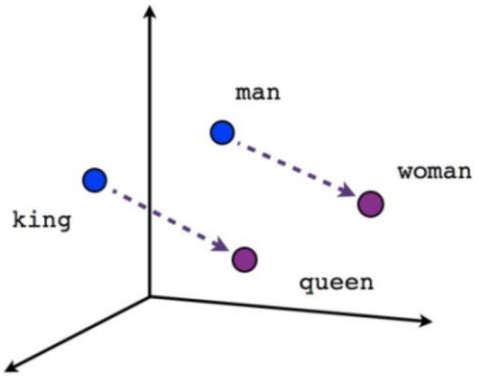


Male-Female

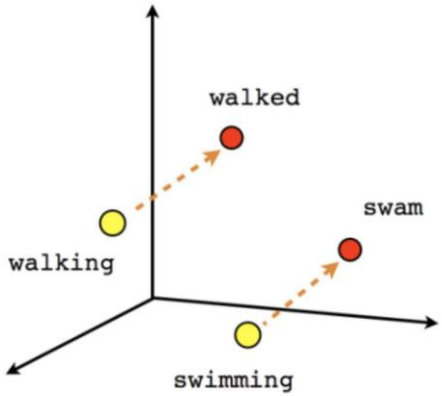
Verb tense

Distributed representations of text

- Each textual unit is mapped to a dense vector of real numbers
  - Overcome data sparsity
  - Features are generated automatically
- The vector representation of text embeds textual similarities into the vector representation
  - Semantically related units are mapped to similar vectors



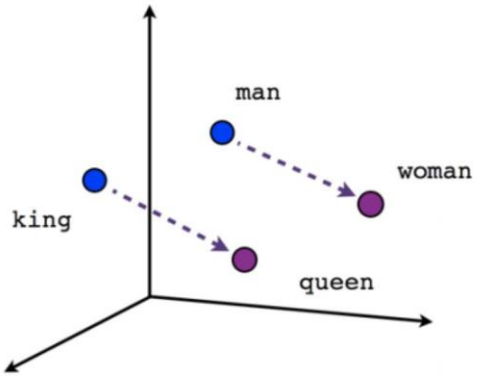
Male-Female



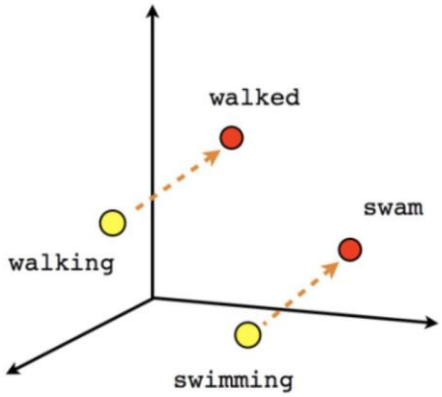
Verb tense

Word-level representations of text

- Also known as "word embedding"
- Each word in a input vocabulary is mapped to a dense high-dimensional vector
  - Typically, 200-300 dimensions
- It encodes both syntactic and semantic word similarities
  - Similar words have similar coordinates in the latent space



Male-Female



Verb tense

Embeddings

- Self-supervised text representations enable better supervised learning on large datasets
  - Not limited to NLP
- In the NLP domain, input data could be
  - Characters
  - Words
  - Phrases
  - Sentences
  - Paragraphs
  - Documents

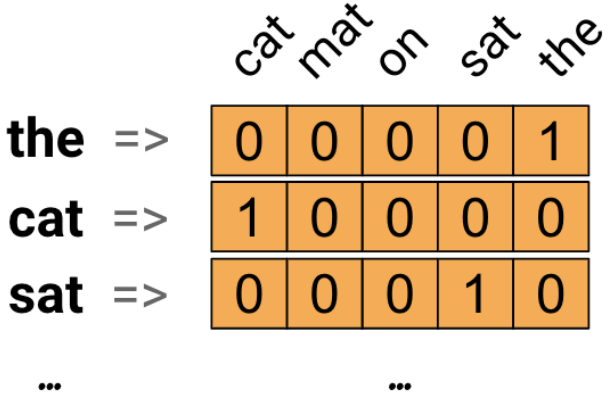
Embeddings

- Vector semantics
  - learning representations of the mearning of words, called embeddings, directly from their distributions in text
- Here we focus on static embeddings
  - The (more powerful) dynamic embeddings will be introduced later on

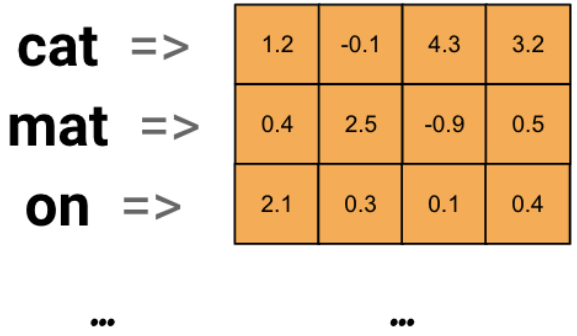
Word embeddings

Multiple architectures have been proposed to embed single words into a fixed-size vector representation

One-hot encoding



A 4-dimensional embedding



https://www.tensorflow.org/text/guide/word_embeddings (latest access: June 2021)

Word embeddings

- Embeddings are dense vectors containing real-valued elements instead of integer/boolean values
- The vectors embed some semantic knowledge for the words they represents
- Vector embeddings are learned using mostly **unsupervised** training procedures

Distributional hypothesis

- Key idea behind feature extraction in embedding models
- Given a target word we look for the semantically related words among those appearing nearby
  - A word is characterized by the company it keeps

A synopsis of linguistic theory 1930-1955. In Studies in Linguistic Analysis, Firth, J.R. (1957). pp. 1-32.

Distributional hypothesis

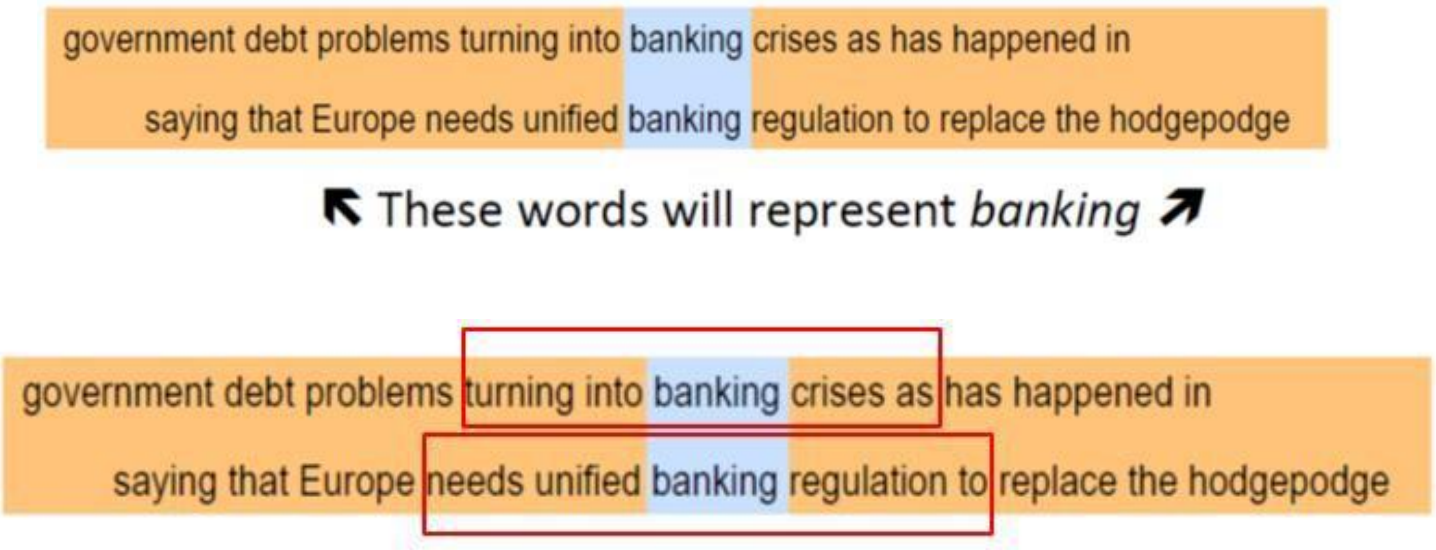
Lenci, A. (2018). Distributional models of word meaning. Annual Review of Linguistics, 4(1), 151–171

Distributional hypothesis

- A fixed-size sliding window denotes the context of a target word
  - Typically, from 3 to 5 words preceding or subsequent to the target word

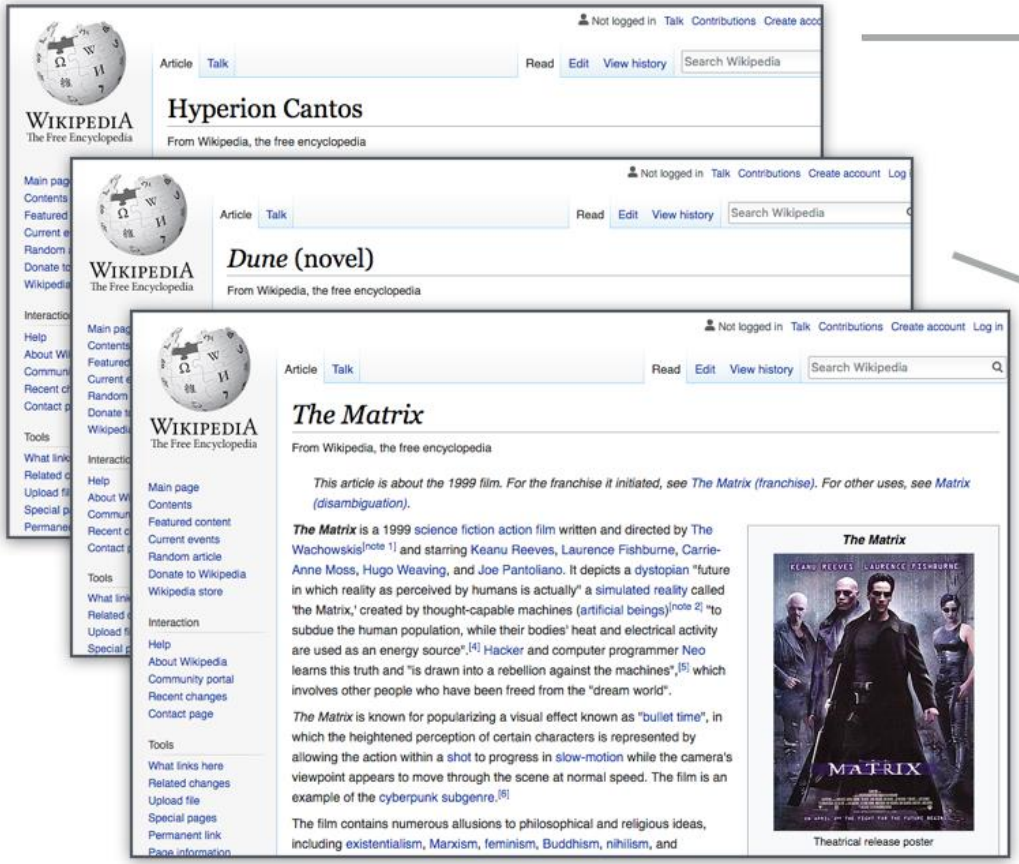
Distributional hypothesis

- The surrounding context of a word is likely to convey its semantic meaning



Richard Socher, Yoshua Bengio, and Christopher D. Manning. 2012. Deep learning for NLP (without magic). In Tutorial Abstracts of ACL 2012. Association for Computational Linguistics, USA, 5..

Distributional hypothesis



The **Hyperion Cantos** is a series of science fiction novels by Dan Simmons. The title ... in the series, *Hyperion* and *The Fall of Hyperion*,[1][2] and later came to refer to the overall storyline, including *Endymion*, *The Rise of Endymion*, and a number of short stories.[3][4] More narrowly, inside the fictional storyline, after the first volume, the Hyperion Cantos is an epic poem written by the character Martin Silenus covering in verse form the events of the first book.[5]

Of the four novels, *Hyperion* received the Hugo and Locus Awards in 1990;[6] *The Fall of Hyperion* won the Locus and British Science Fiction Association Awards in 1991;[7] and *The Rise of Endymion* received the Locus Award in 1998.[8] All four novels were also nominated for various science fiction awards.

An event series is being developed by Bradley Cooper, Graham King, and Todd Phillips for Syfy based on the first novel *Hyperion*.[9]

Dune is a 1965 science fiction novel by American author Frank Herbert, originally published as two separate serials in *Analog* magazine. It tied with Roger Zelazny's *This Immortal* for the Hugo Award in 1966,[3] and it won the inaugural Nebula Award for Best Novel.[4] It is the first installment of the *Dune* saga, and in 2003 was cited as the world's best-selling science fiction novel.[5][6]

Set in the distant future amidst a feudal interstellar society in which noble houses, in control of individual planets, owe allegiance to the Padishah Emperor, *Dune* tells the story of young Paul Atreides, whose noble family accepts the stewardship of the planet Arrakis. It is an inhospitable and sparsely populated desert wasteland, but is also the only source of melange, also known as "spice", a drug that enhances mental abilities. As melange is the most important and valuable substance in the universe, control of Arrakis is a coveted—and dangerous—undertaking. The story explores the multi-layered interactions of politics, religion, ecology, technology, and human emotion, as the factions of the empire confront each other in a struggle for the control of Arrakis

The Matrix is a 1999 science fiction action film written and directed by The Wachowskis[note 1] and starring Keanu Reeves, Laurence Fishburne, Carrie-Anne Moss, Hugo Weaving, and Joe Pantoliano. It depicts a dystopian "future in which reality as perceived by humans is actually" a simulated reality called 'the Matrix,' created by thought-capable machines (artificial beings)[note 2] "to subdue the human population, while their bodies' heat and electrical activity

Word embeddings

- Key steps
  1. Collect a large document corpus
  2. Define a word-level vocabulary
  3. Slide over the text to build training samples
  4. Train an ad hoc network
  5. Learn word vectors from the network

Christopher D. Manning and Pandu Nayak. Introduction to Information Retrieval. CS276

Why do we need neural network learning?

- Drawbacks of window-based co-occurrence models
  - Not effective
    - Curse of dimensionality
  - Not efficient
    - Quadratic complexity with the number of words

Christopher D. Manning and Pandu Nayak. Introduction to Information Retrieval. CS276

Why do we need neural network learning?

- Examples of sentences
  - *I like Deep Learning*
  - *I like NLP*
  - *I enjoy flying*

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Richard Socher, Yoshua Bengio, and Christopher D. Manning. 2012. Deep learning for NLP (without magic). In Tutorial Abstracts of ACL 2012. Association for Computational Linguistics, USA, 5..

Why do we need neural network learning?

- The dimensions of the matrix change very often
  - new words are added very frequently
  - The corpus changes in size
- The matrix is extremely sparse since most words do not co-occur
- The matrix is very high dimensional in general
- Quadratic cost to train
  - E.g., SVD
- Hard to account the imbalances in word frequencies

Word2Vec

Word2Vec has been proposed by *Mikolov et al.* in 2013

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

Word2Vec

- Pioneering approach to efficiently learn word embeddings
  - 20k++ citations according to Google Scholar
- Key idea
  - Train a neural neural network to predict the surrounding words of every word in a dictionary

  or
  - Train a neural neural network to predict the target word given the surrounding words in the dictionary

Efficient estimation of word representations in vector space. T Mikolov, K Chen, G Corrado, J Dean. ICLR (Workshop Poster) 2013

Word2Vec

- Given
  - a large document corpora
  - a dictionary of the words occurring in the documents
- Approach
  - For each word w in the dictionary compute its pairwise vector similarity with every word w' appearing in its contexts C
  - Compute the probabilities $p(w'|w)\ \forall\ w' \in C$
  - Adjust word vectors in order to maximize $p(\cdot)$
- Output
  - A word vector space

Efficient estimation of word representations in vector space. T Mikolov, K Chen, G Corrado, J Dean. ICLR (Workshop Poster) 2013

Word2Vec

- The most relevant concepts proposed in the paper are:
  - **Skip-gram** training algorithm
  - **Negative sampling**
- Both compose the whole training procedure to learn Word2vec embeddings (SGNS)
- Word2Vec models are **static** embeddings
  - Each word (lemma) is represented with a single vector
    - regardless of the context

Word2Vec: the intuition

- The model consists of a classifier trained for a binary classification task

Given a word $\boldsymbol{w_0}$, is it likey to appear near to the word $\boldsymbol{w_1}$?

- The classifier should be trained with some supervision indicating whether $\boldsymbol{w_0}$ and $\boldsymbol{w_1}$ are somewhat related to

Word2Vec: the intuition

- **Context window**
  - it is possible to leverage running text to create positive examples for the classifier
    - A sliding window is used to define contextual positive examples

The window size is an **hyperparameter** of Word2Vec

Word2Vec: the intuition

- **Context window**
  - it is possible to leverage running text to create positive examples for the classifier
    - A sliding window is used to define contextual positive examples

What is the window size?

Word2Vec: the intuition

- **Context window**
  - it is possible to leverage running text to create positive examples for the classifier
    - A sliding window is used to define contextual positive examples

What is the window size?

Word2Vec: the intuition

- **Context window**
  - it is possible to leverage running text to create positive examples for the classifier
    - A sliding window is used to define contextual positive examples

What is the window size?

Word2Vec: the intuition

- **Context window**
  - it is possible to leverage running text to create positive examples for the classifier
    - A sliding window is used to define contextual positive examples

What is the window size?

Word2Vec: the intuition

- **Context window**
  - it is possible to leverage running text to create positive examples for the classifier
    - A sliding window is used to define contextual positive examples

What is the window size?

Word2Vec: the intuition

The size of the context window determines how many words **before and after** a given word are included as context words.

What is the window size? 5

Window size: $2 \times N_{\text{left/right}} + 1$

Word2Vec: the intuition

- This way of defining positive examples is called **self-supervision**
  - It avoids the need for any sort of hand-labeled supervision signal

John	0
likes	**1**
to	0
watch	0
movies	0
...	0

John	**0**
likes	0
to	**1**
watch	0
movies	0
...	0

Word2Vec architectures

- The Skip-gram model (Skip-Gram)
  - Predict the surrounding words given the target word
- The Continuous Bag Of Words model (CBOW)
  - Predict the target word given the surrounding words

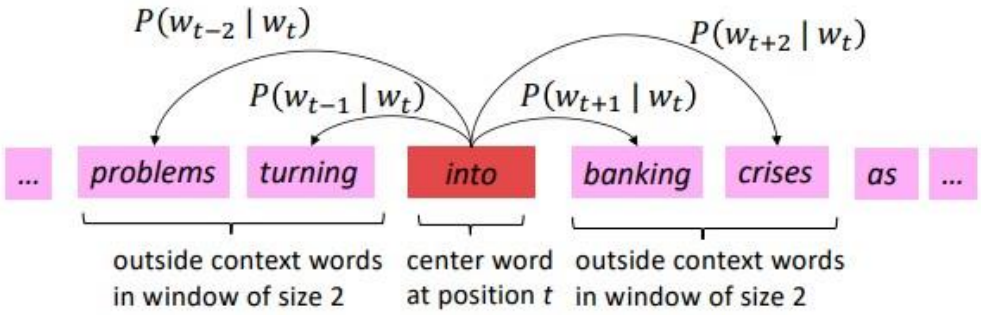
Efficient estimation of word representations in vector space. T Mikolov, K Chen, G Corrado, J Dean. ICLR (Workshop Poster) 2013

Word2Vec architectures

- Skip-Gram ignores the relative word position within the context
- CBOW ignores the eventual presence of word repetition

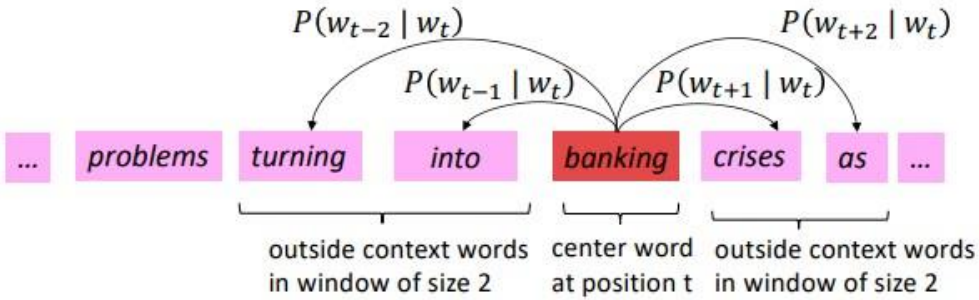
Efficient estimation of word representations in vector space. T Mikolov, K Chen, G Corrado, J Dean. ICLR (Workshop Poster) 2013

Word2Vec: SkipGram model



Efficient estimation of word representations in vector space. T Mikolov, K Chen, G Corrado, J Dean. ICLR (Workshop Poster) 2013

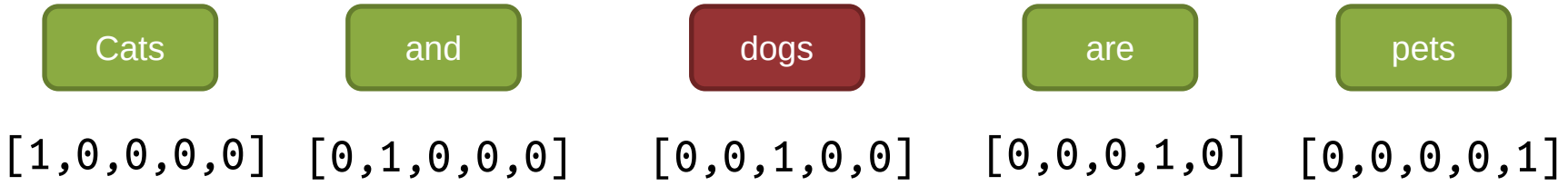
Word2Vec: SkipGram model



Efficient estimation of word representations in vector space. T Mikolov, K Chen, G Corrado, J Dean. ICLR (Workshop Poster) 2013

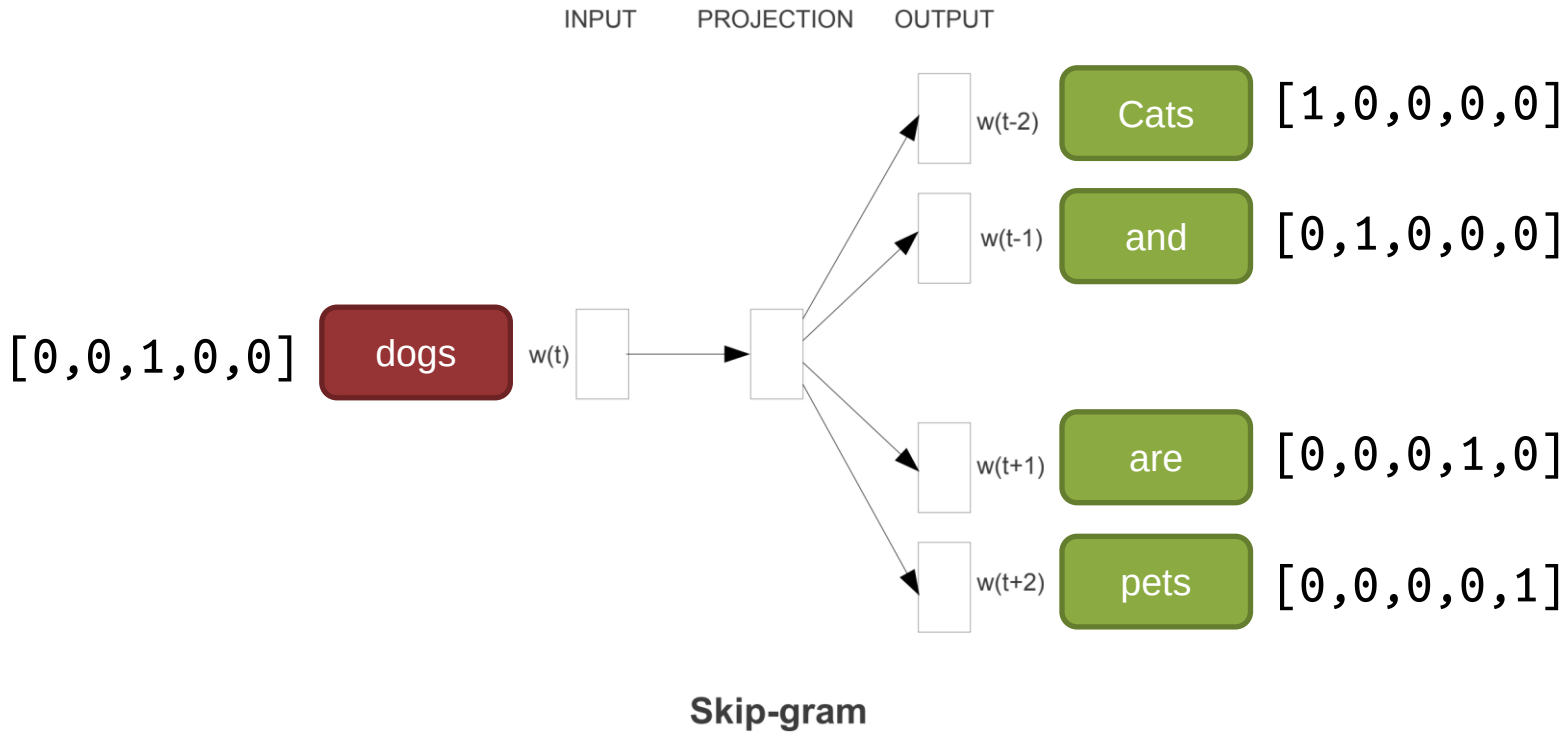
Skip-Gram

- Use the central word to predict the surrounding words



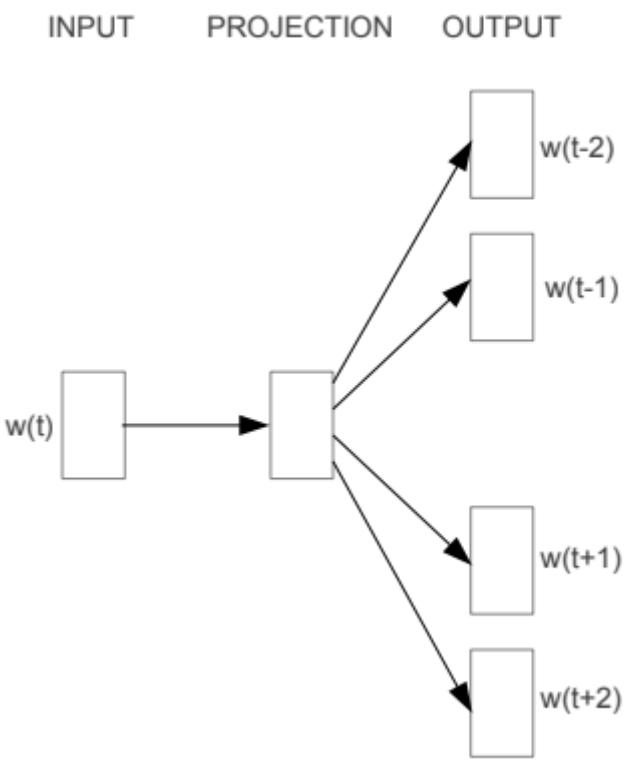
Skip-Gram

- Skip-gram uses the central word to predict the surrounding words



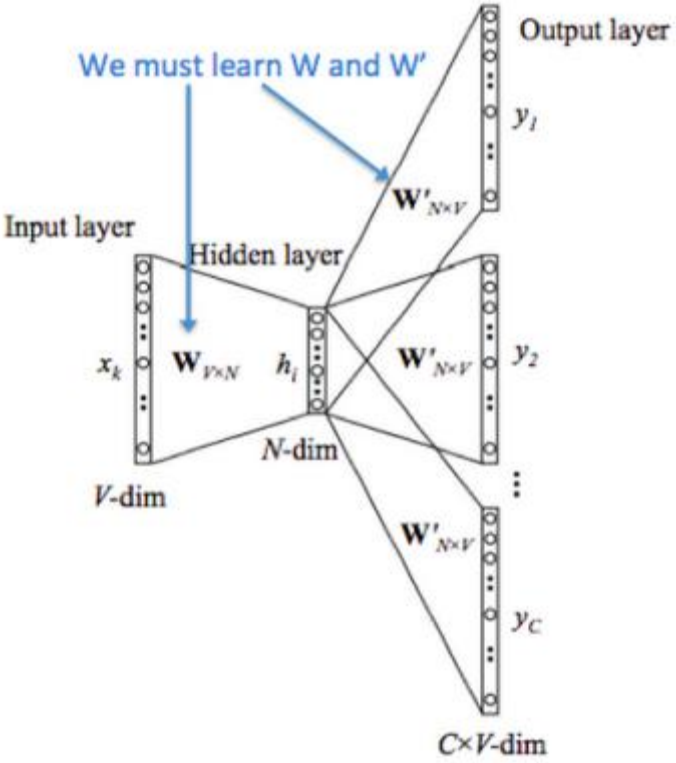
Skip-gram

Skip-Gram



Efficient estimation of word representations in vector space. T Mikolov, K Chen, G Corrado, J Dean. ICLR (Workshop Poster) 2013

Skip-Gram



CS 224D: Deep Learning for NLP. Lecture Notes: Part I. Francois Chaubard, Rohit Mundra, Richard Socher. 2016.

Skip-Gram

- Given a sequence of training words w_1, w_2 ,..., w_T the objective is to maximize the average log probability

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

where c is the size of the training context

Efficient estimation of word representations in vector space. T Mikolov, K Chen, G Corrado, J Dean. ICLR (Workshop Poster) 2013

Skip-Gram

- Softmax function

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^{\top} v_{w_I}\right)}$$

where v_w and v'_w are the input and output vector representations of w and W is the dictionary size

Efficient estimation of word representations in vector space. T Mikolov, K Chen, G Corrado, J Dean. ICLR (Workshop Poster) 2013

Skip-Gram

- Softmax is highly inefficient
  - The cost of computing $\nabla \log p(w_O|w_I)$ is proportional to the dictionary size
    - 10^5-10^7
- Solution: Hierarchical Softmax function
  - Binary tree representation of the output layer
  - Instead of evaluating *W* output nodes compute just log(W) nodes

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'13). Curran Associates Inc., Red Hook, NY, USA, 3111–3119.

Skip-Gram

- Hierarchical Softmax function

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma\left(\llbracket n(w,j+1) = \mathrm{ch}(n(w,j)) \rrbracket \cdot {v'_{n(w,j)}}^{\top} v_{w_I}\right)$$

where

- n(w, j): j-th node on the path from the root to w
  - L(w) path length
  - n(w, 1) = root and n(w, L(w)) = w
- ch(n): an arbitrary fixed child of node n
- [[x]]: 1 if x is true and -1 otherwise
- $\sigma(x) = 1/(1+\exp(-x))$

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'13). Curran Associates Inc., Red Hook, NY, USA, 3111–3119.

Skip-Gram

- Hierarchical Softmax function

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma\left([\![n(w,j+1) = \text{ch}(n(w,j))]\!] \cdot {v'_{n(w,j)}}^\top v_{w_I}\right)$$

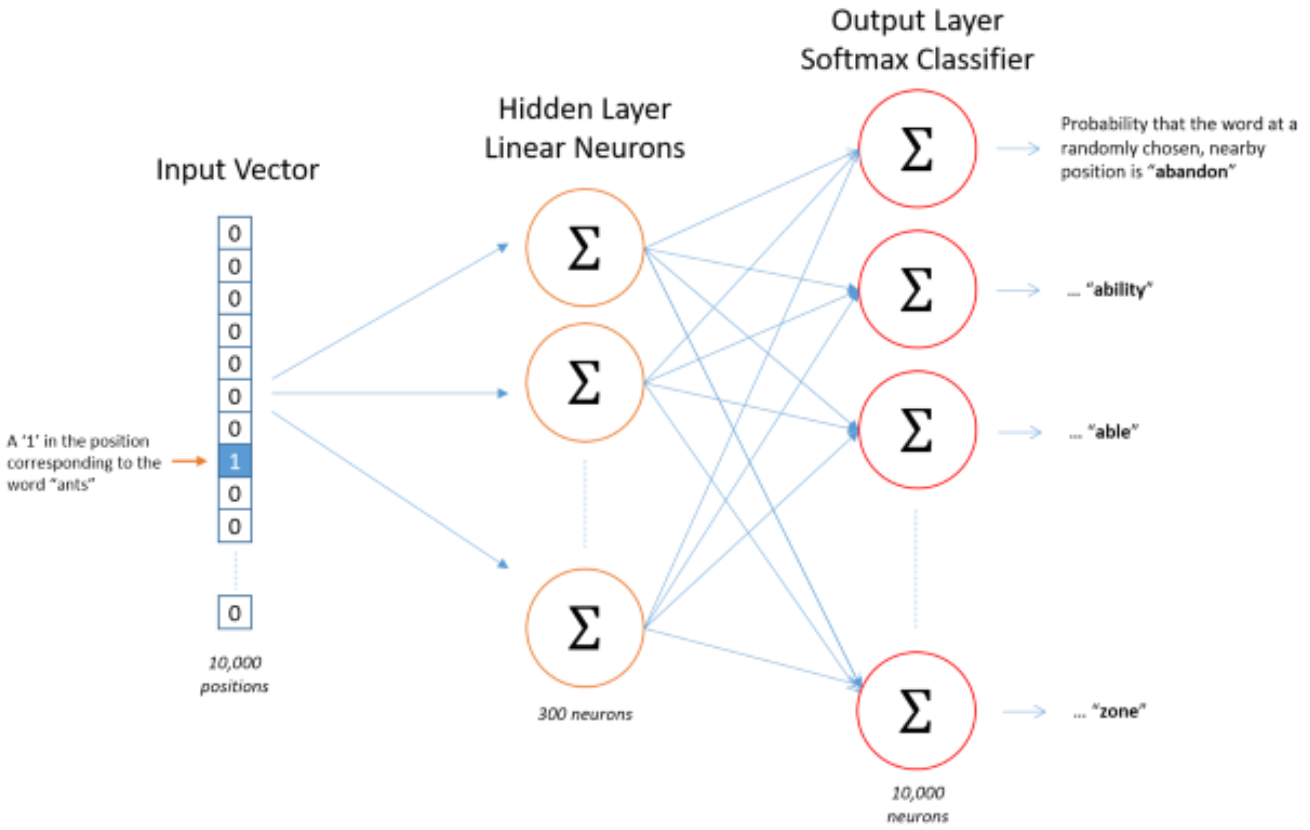
$$\sum_{w=1}^{W} p(w|w_I) = 1$$

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'13). Curran Associates Inc., Red Hook, NY, USA, 3111–3119.

Skip-Gram

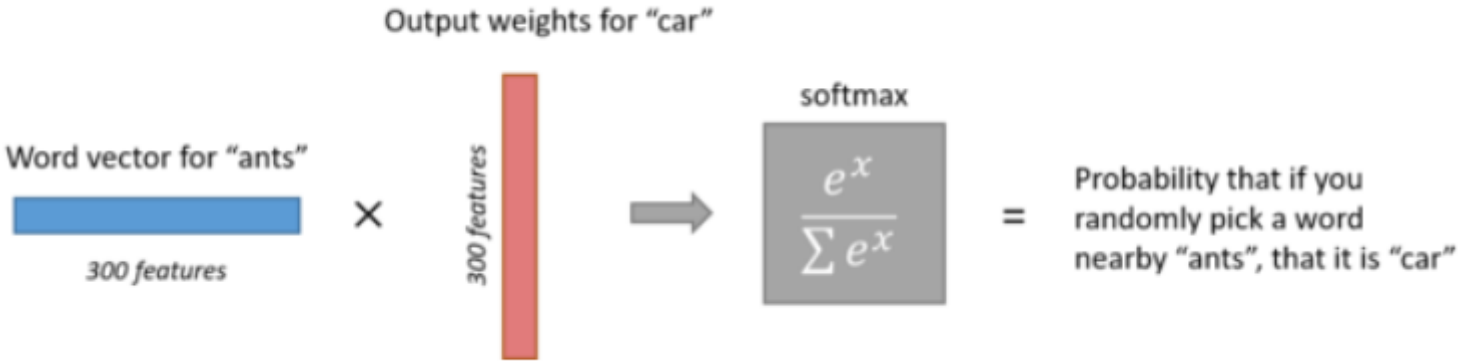
- Running example
  - 10,0000 words in the dictionary
  - 300 features
    - 300 neurons
- Output of the Skip-Gram network
  - A 10,000 component vector
  - For each word in the vocabulary it contains the probability that a randomly selected nearby word is that vocabulary word

Skip-Gram



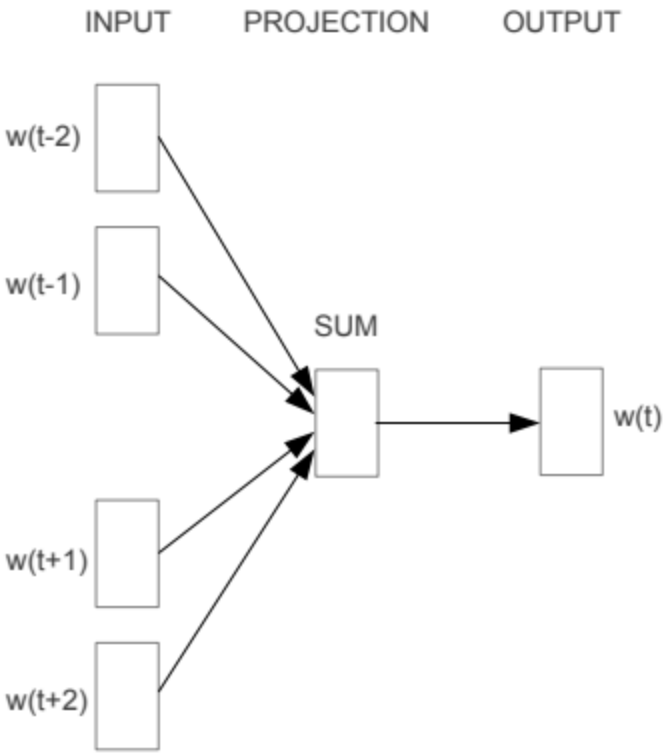
Source: https://towardsdatascience.com (latest access: September 2021)

Skip-Gram



Source: https://towardsdatascience.com (latest access: September 2021)

CBOW



Efficient estimation of word representations in vector space. T Mikolov, K Chen, G Corrado, J Dean. ICLR (Workshop Poster) 2013

CBOW

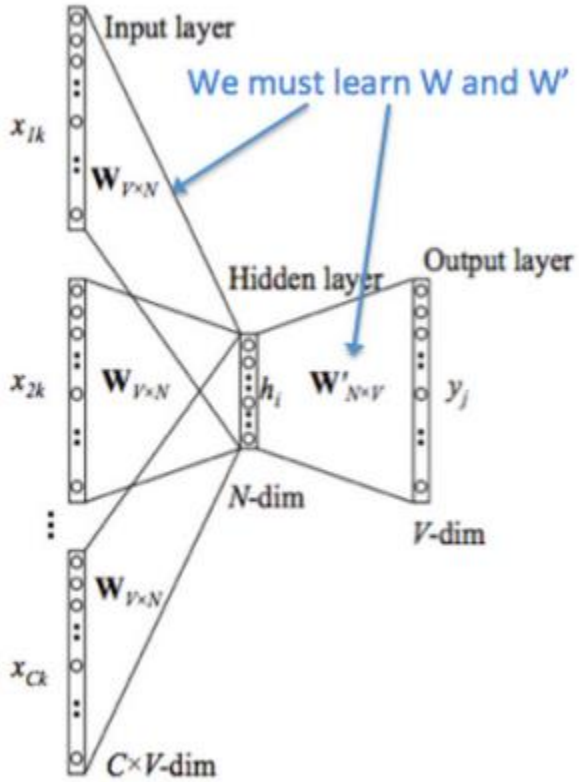
- Given a sequence of training words w_1, w_2 ,..., w_T the objective is to maximize the average log probability

$$\log P(w_c | w_{c-m}, \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+m})$$

where m is the size of the training context

Efficient estimation of word representations in vector space. T Mikolov, K Chen, G Corrado, J Dean. ICLR (Workshop Poster) 2013

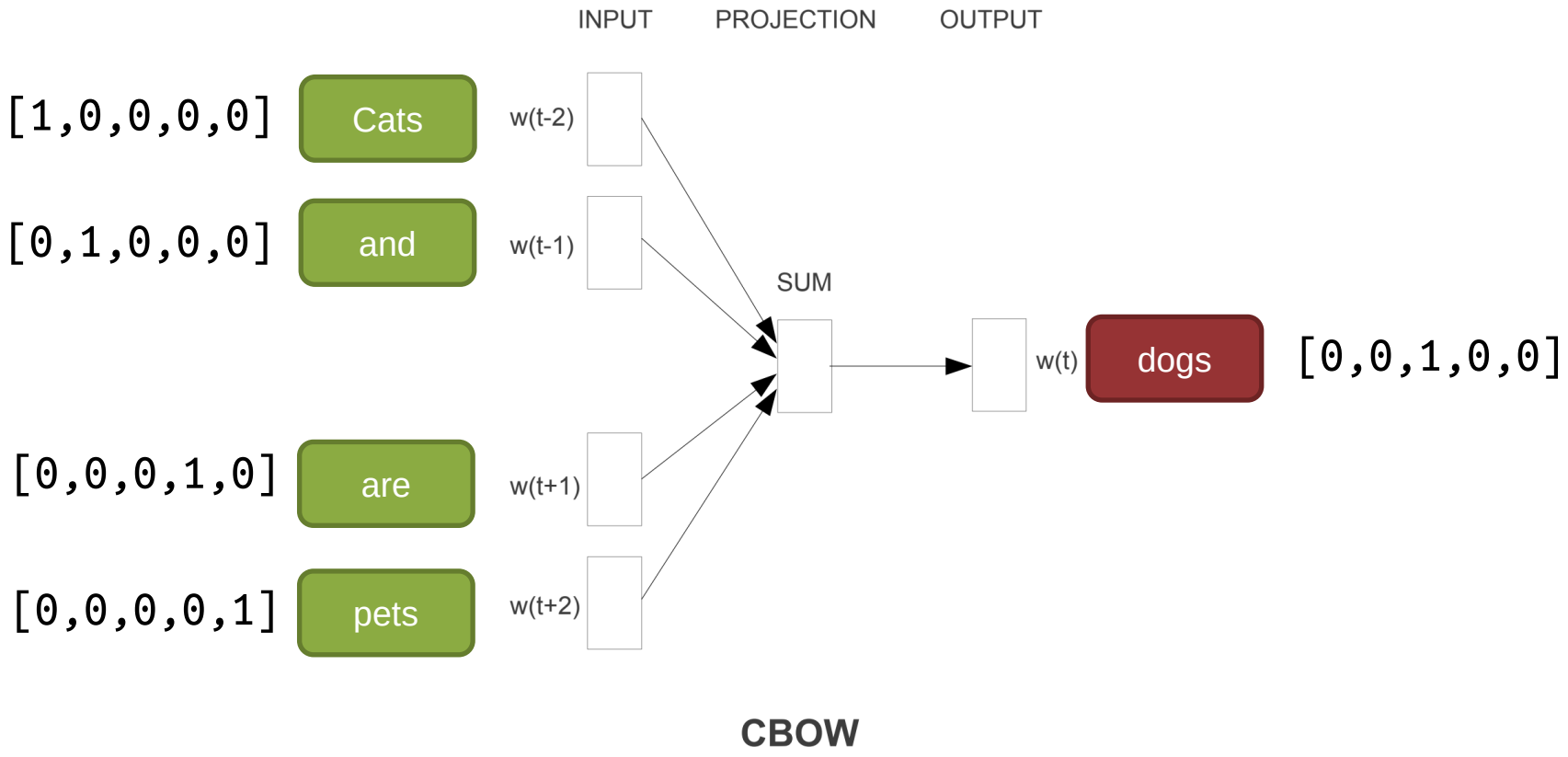
CBOW



CS 224D: Deep Learning for NLP. Lecture Notes: Part I. Francois Chaubard, Rohit Mundra, Richard Socher. 2016.

CBOW

- The distributed representations of the context is used to predict the word in the middle of the window



Word2Vec: the intuition

- The Skip-Gram model learns to predict a target word thanks to a nearby word
- The CBOW model predicts the target word according to its context
- The **context** is represented as a bag of the words contained in a **fixed size window** around the target word

Word2Vec: the intuition

Training steps (Skip-Gram)

1. Target and neighboring words are **positive** examples
2. Randomly sample other words to get **negatives**
3. Use logistic regression to train a classifier able to distinguish those 2 cases
4. Use learned weights as word **embeddings**

Binary classification

The classification task:

Cit. Geoffrey Hinton

Computers will understand sarcasm before Americans do.

c_1 c_2 w c_3 c_4

The goal is to train a classifier such that, given a pair **(w, c)** will return the probability that **c** is a real context word (for **w**).

$$P(+ \mid w, c)$$

Binary classification

The classification task:

Cit. Geoffrey Hinton

Computers will understand sarcasm before Americans do.

c_1 c_2 w c_3 c_4

The probability that **c** is not a context word is:

$$P(\text{ - } \mid w, c) = 1 - P(+ \mid w, c)$$

Binary classification

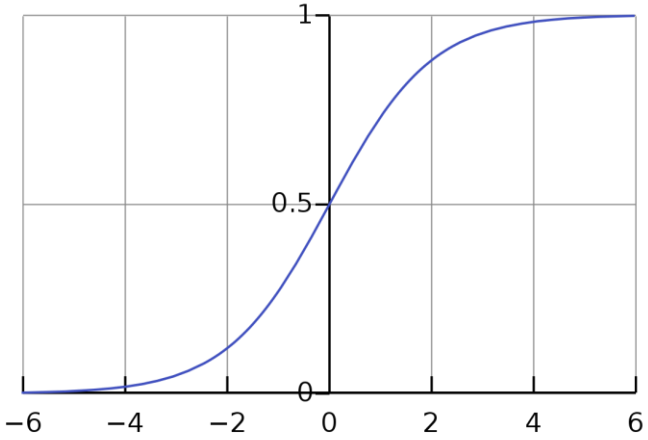
How can we compute P?

- By using skipgram this probability is based on the embedding similarity
- A word is likely to occur near the target if its embedding vector is similar to the target embedding
- Two words are **similar** if their vectors have a **high dot product**

$$\text{Similarity}\ (w, c) \approx \mathbf{c} \cdot \mathbf{w}$$

Dot product -> probability function

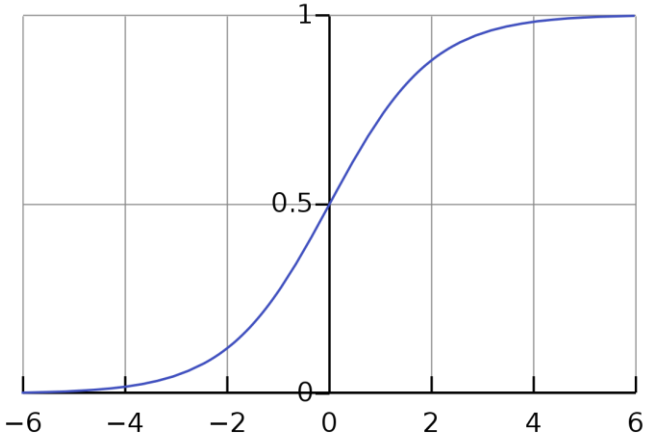
- The dot product **is not** a probability function
- It ranges from -∞ to +∞
- Logistic (sigmoid) function is commonly used to compute the probability



$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Probability function

- The logistic (sigmoid) function is commonly used to compute the probability



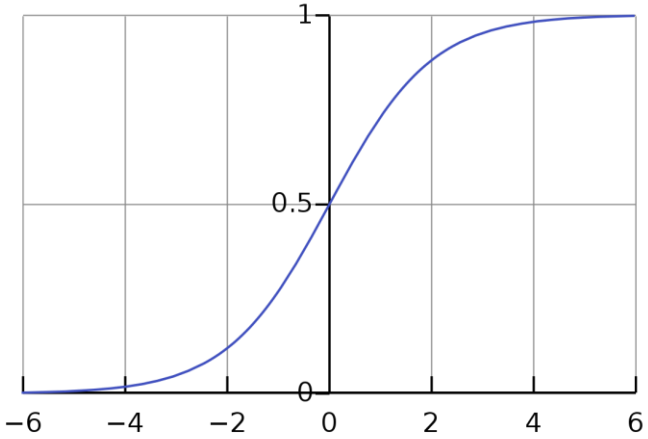
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$P(+|w, c) = \sigma(c \cdot w)$$

$$P(+|w, c) = \sigma(c \cdot w) = \frac{1}{1 + e^{-c \cdot w}}$$

Probability function

- We apply the same function to compute the probability of the negative class



$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$P(-|w, c) = \sigma(-c \cdot w)$$

$$P(-|w, c) = \sigma(-c \cdot w) = \frac{1}{1 + e^{c \cdot w}}$$

The Skip-Gram assumption

$$P(+|w,c) = \sigma(c \cdot w) = \frac{1}{1 + e^{-c \cdot w}}$$

- This formulation gives the probability of occurrence of **one** word, but ...
  - The context contains 2 x $N_{left/right}$ words!
- **The Skip-Gram assumption**
  - All context words are independent of each other

Skip-Gram assumption

- All context words are independent of each other

$$P(+|w, c_{1:L}) = \prod_{i=1}^{L} \sigma(c_i \cdot w)$$

$$log\ P(+|w, c_{1:L}) = \sum_{i=1}^{L} log\ \sigma(c_i \cdot w)$$

Computing the probabilities

- Given a word and one of its context words, the probabilistic classifier assigns a probability based on how similar those words are
- Skip-Gram exclusively relies on the embeddings of the target and context words
  - For all the words in the vocabulary

Learning embeddings

Input: a corpus of text and a chosen vocabulary size $|V|$

Output: embedding matrix containing all words in V

1. Random embedding vector $\forall$ w $\in$ V
2. Iteratively shift embeddings of words to be similar to their context words (and less similar to non-context words)
3. Return embedding matrix

Learning embeddings

Cit. Geoffrey Hinton

Computers will understand sarcasm before Americans do.

c_1 c_2 w c_3 c_4

w	c_{pos}
...	...

Window size: _

Positive examples: _

Learning embeddings

Cit. Geoffrey Hinton

Computers will understand sarcasm before Americans do.

c_1 c_2 w c_3 c_4

Window size: 5

Positive examples: 4

w	c_{pos}
sarcasm	will
sarcasm	understand
sarcasm	before
sarcasm	Americans

Learning embeddings

Cit. Geoffrey Hinton

Computers will understand sarcasm before Americans do.

c_1 c_2 w c_3 c_4

Window size: 5

Positive examples: 4

w	c_{pos}
sarcasm	will
sarcasm	understand
sarcasm	before
sarcasm	Americans

What about negative examples?

Skip-Gram vs. CBOW

- CBOW is faster to train than Skip-Gram
  - CBOW complexity: $N \times D + D \times \log_2(V)$
  - Skip-Gram performance and complexity also depends on the context size
- Skip-Gram better captures semantic word relationships whereas CBOW better captures syntactic ones
- CBOW is more sensitive than Skip-Gram to word frequency imbalance

Further Word2Vec optimizations

- Negative sampling
  - Modified optimization function
  - Each training sample triggers the update of only a small percentage of the model's weights
- Frequent word subsampling
  - Decrease the number of training examples

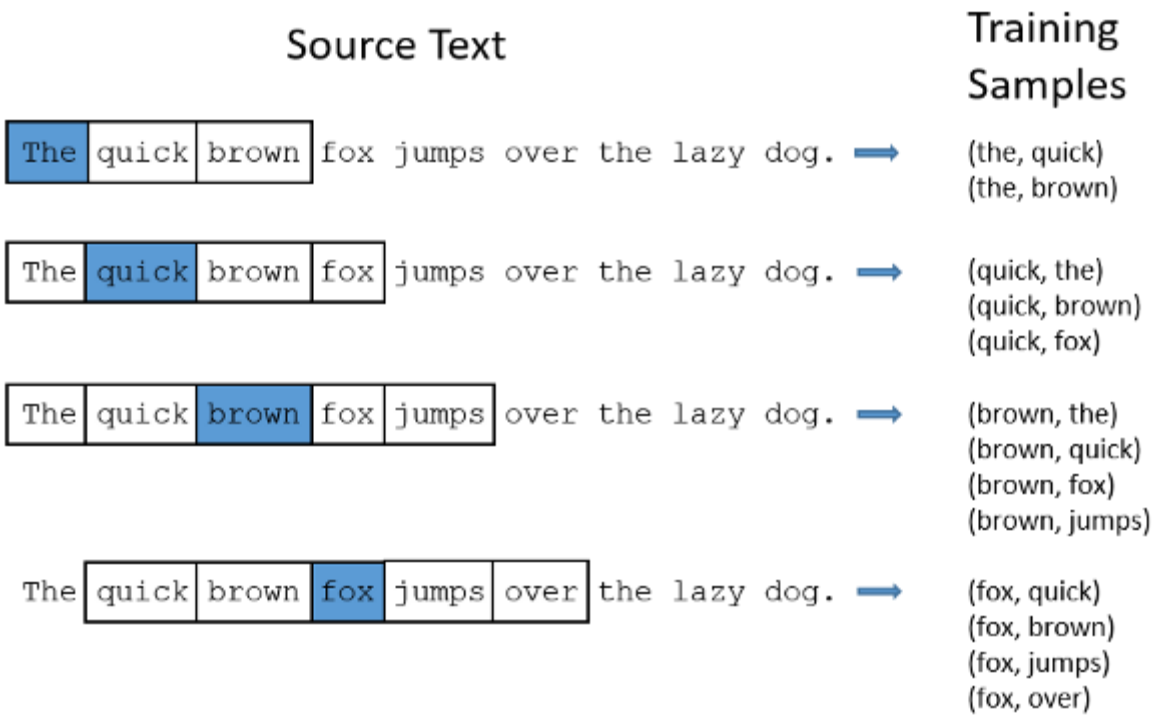
Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'13). Curran Associates Inc., Red Hook, NY, USA, 3111–3119.

Negative sampling

- The Skip-Gram requires the updating of a very large number of network weights
- We randomly select just a small number of "negative" words to update the weights for
- We still update the weights for the "positive" word

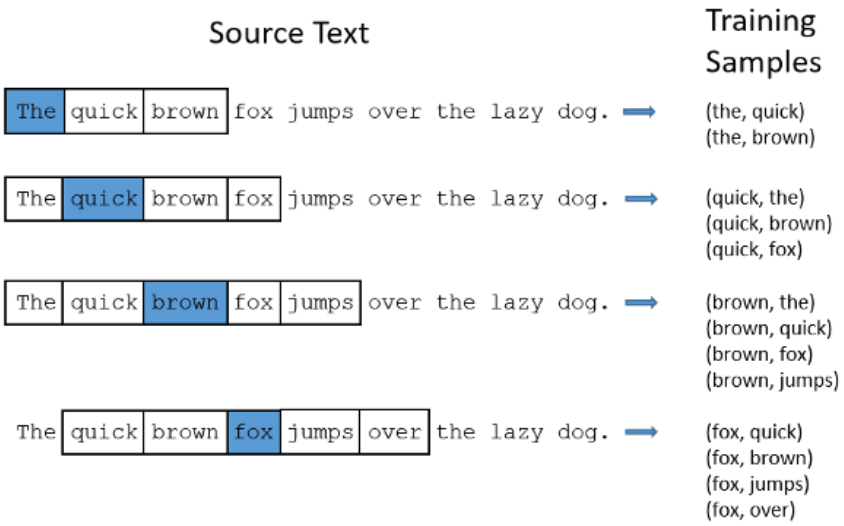
Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'13). Curran Associates Inc., Red Hook, NY, USA, 3111–3119.

Frequent word subsampling



Source: https://towardsdatascience.com (latest access: September 2021)

Frequent word subsampling



- Common words like "the" cause problems
  - Looking at word pairs such as ("fox", "the") doesn't tell us much about the meaning of "fox".
  - "the" appears in the context of pretty much every word.
  - We will have many more samples of ("the", ...) than we need to learn a good vector for "the".

Source: https://towardsdatascience.com (latest access: September 2021)

Negative sampling

Cit. Geoffrey Hinton

Computers will understand sarcasm before Americans do.

c_1 c_2 w c_3 c_4

For each positive example **K** random words are selected from the vocabulary V

K is an hyperparameter of the Word2Vec model

Vocabulary
a
ability
able
about
...
zone
zoo

Negative sampling

Cit. Geoffrey Hinton

Computers will understand sarcasm before Americans do.

c_1 c_2 w c_3 c_4

Skip-gram selects more negative than positive examples

Training pairs **(w, c_{neg})** are passed to the classifier associating the negative class

Vocabulary
a
ability
able
about
...
zone
zoo

Negative sampling

Negative examples are chosen according to their *weighted unigram frequency*.

Un-weighted unigram frequency: all words have the same probability to be chosen

$$p(w) = \frac{1}{|V|}$$

Negative sampling

Negative examples are chosen according to their *weighted unigram frequency*.

Weighted unigram frequency: the probability of selecting a word is function of its occurrences in the corpus

$$p_\alpha(w) = \frac{count(w)^\alpha}{\sum_{w'} count(w')^\alpha}$$

Standard value for $\alpha = 0.75$

Negative sampling

- Negative examples are chosen according to their *weighted unigram frequency*
- Let us set α to 0.75 and consider the following count:

w	occurrences
the	100
sarcasm	10
deep	15
of	75
Total	**200**

Negative sampling

- Negative examples are chosen according to their *weighted unigram frequency*
- Let us set α to 0.75 and consider the following count:

w	occurrences
the	100
sarcasm	10
deep	15
of	75
Total	**200**

Un-weighted unigram frequency

$$p(the) = 0.25$$

$$p(sarcasm) = 0.25$$

Negative sampling

- Negative examples are chosen according to their *weighted unigram frequency*
- Let us set α to 0.75 and consider the following count:

w	occurrences
the	100
sarcasm	10
deep	15
of	75
Total	**200**

Weighted unigram frequency

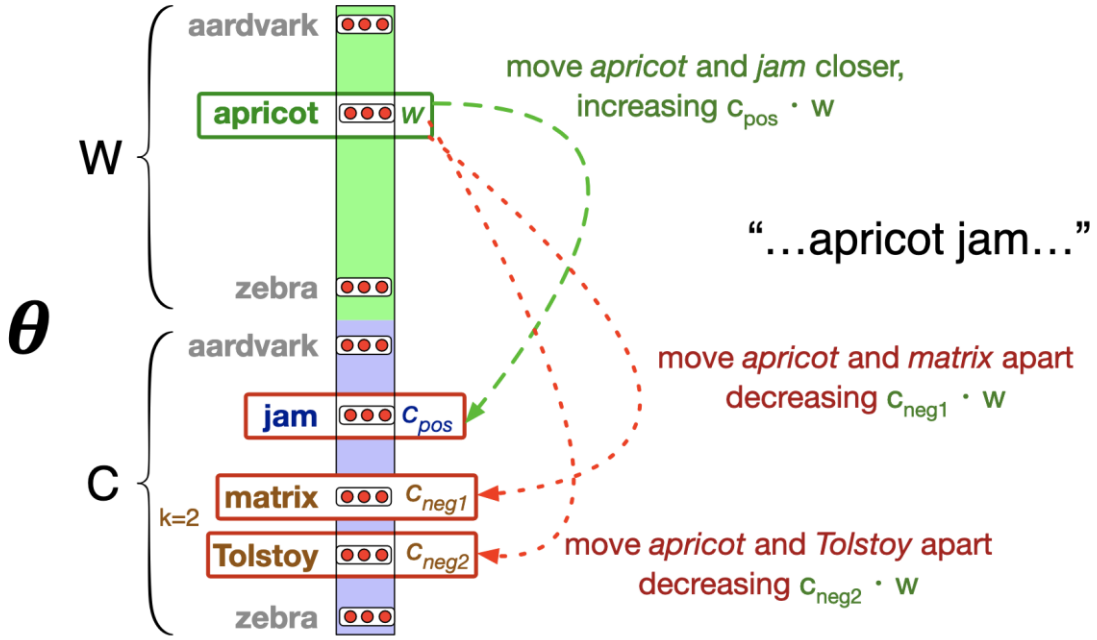
$$p_\alpha(the) = \frac{(0.5)^{0.75}}{(0.5)^{0.75} + (0.05)^{0.75} + (0.075)^{0.75} + (0.375)^{0.75}} = 0.45$$

$$p_\alpha(sarcasm) = \frac{(0.05)^{0.75}}{(0.5)^{0.75} + (0.05)^{0.75} + (0.075)^{0.75} + (0.375)^{0.75}} = 0.08$$

Classification objective

The goal of the classification model is to

- **Maximize** the similarity of the target word and positive examples
- **Minimize** the similarity of the target word and negative examples



Jurafsky, D., & Martin, J. H. (2014). Speech and language processing. Vol. 3. *US: Prentice Hall*.

Classification objective

W and **C** are two randomly initialized matrices.

The classification algorithm update both W and C weights according to the objective function.

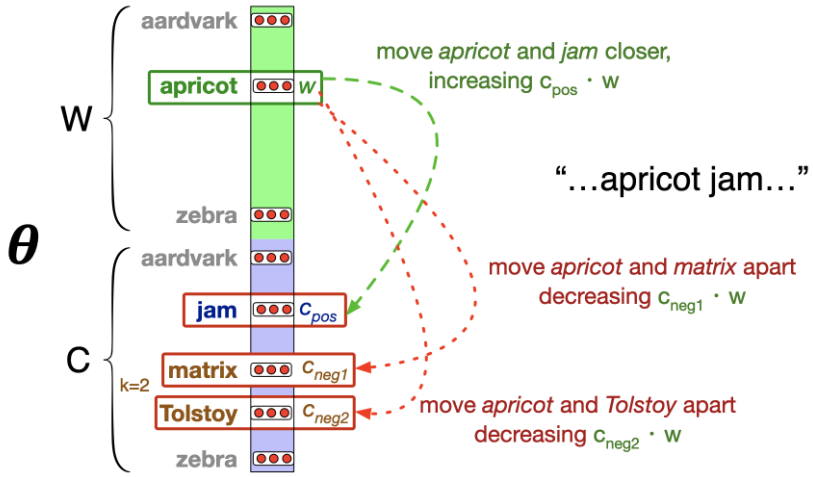
Skip-Gram learns two separate embeddings for a given word

Target embeddings: w_i

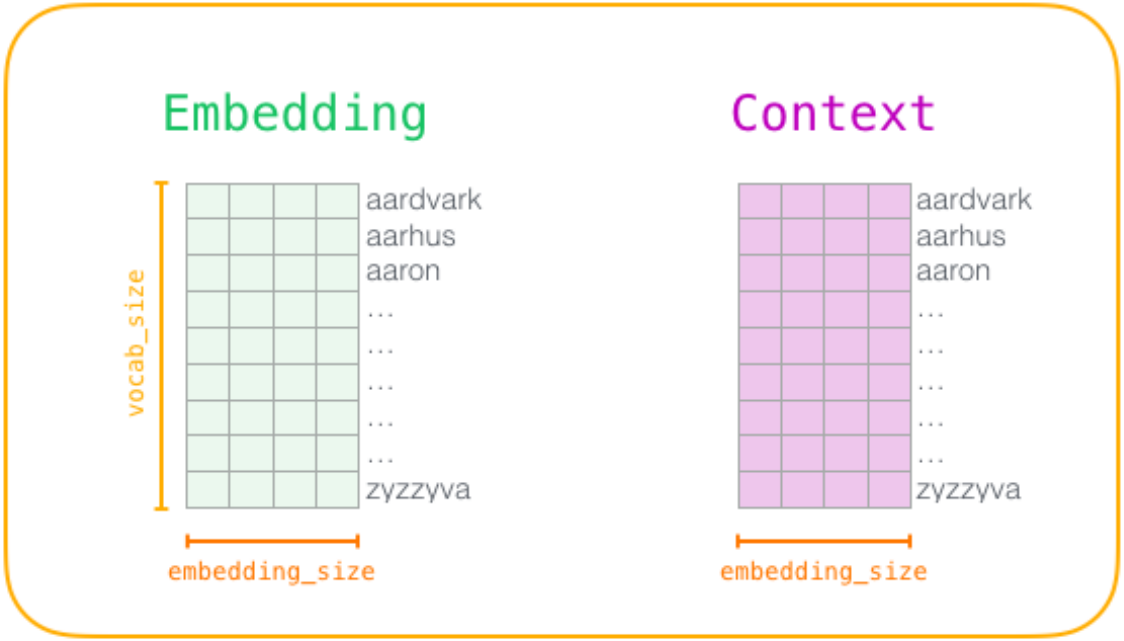
Context embeddings: c_i

Final embeddings:

- $w_i + c_i$
- w_i (**C** matrix is discarded)



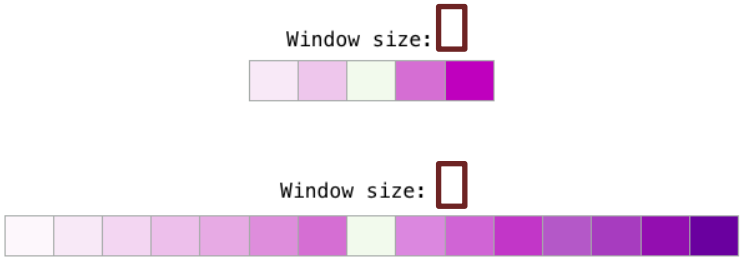
Final Word2Vec model

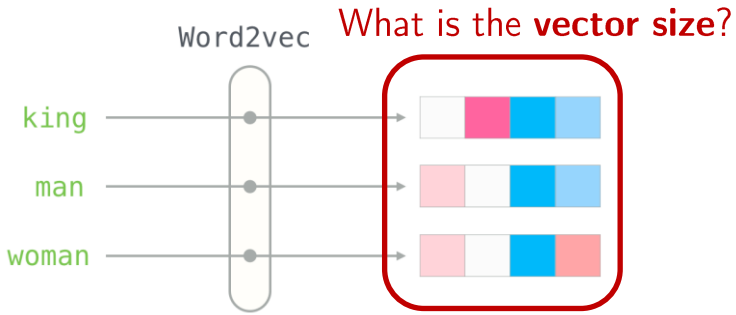


Word2Vec hyperparameters

Additional hyper-parameters of the network need to be set:

- Window size
- Embedding vector size



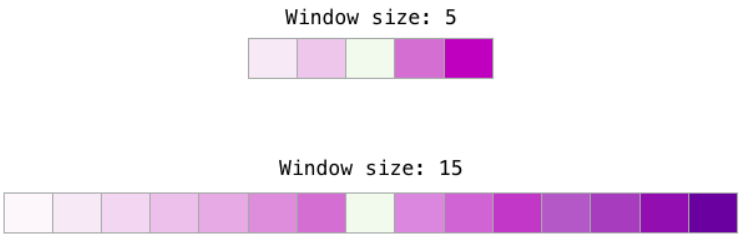


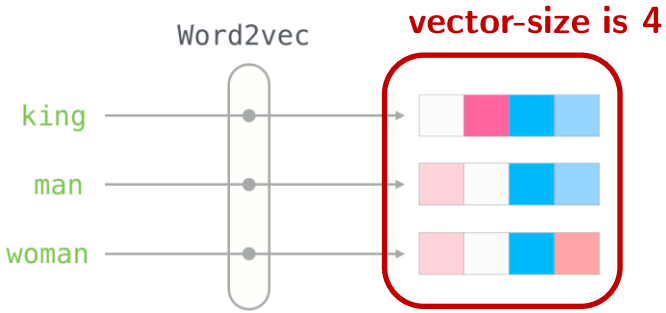
https://radimrehurek.com/gensim/models/word2vec.html - https://jalammar.github.io/illustrated-word2vec/ (latest access: October 2021)

Word2Vec hyperparameters

Additional hyper-parameters of the network need to be set:

- Window size
- Embedding vector size

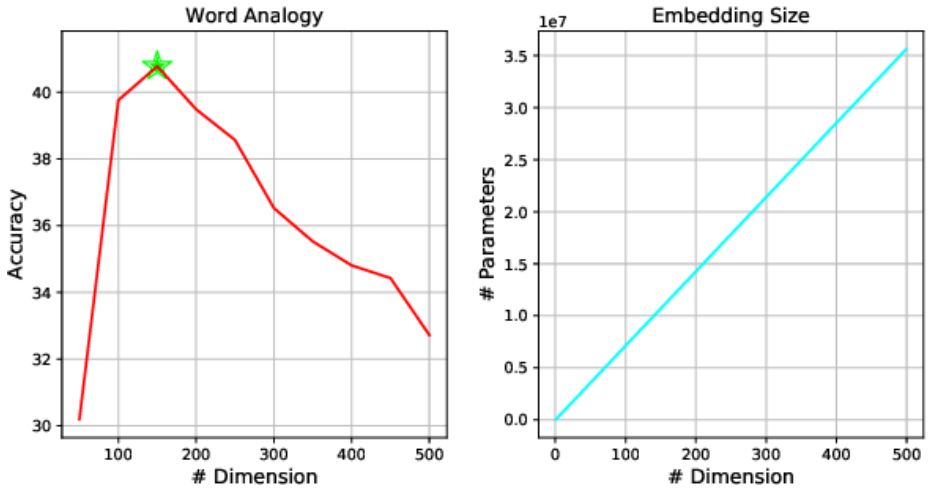




https://radimrehurek.com/gensim/models/word2vec.html - https://jalammar.github.io/illustrated-word2vec/ (latest access: October 2021)

What is the correct embedding size?

- There is no unique answer to the above question
- It depends on
  - The methodology used to perform the projection
  - The amount of data available to train the projection architecture



Wang, Y. (2019, November). Single Training Dimension Selection for Word Embedding with PCA. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3597-3602).

What is the correct window size?

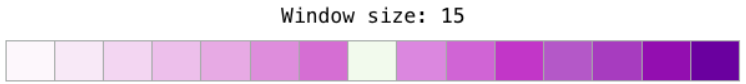
Even in this case there is no unique answer

Bigger window size (15-49):

- Longer training
- Similar words *refer to the same context*
  - E.g. hospital, doctor, nurse

Smaller window size (3-13):

- Shorter training
- Similar words are *interchangeable*
  - E.g. pet, animal



Pre-processing

Stopwords removal: remove words that are not relevant to the meaning of the sentence, such as articles, prepositions, and conjunctions. They are common words that do not add much information to the sentence.

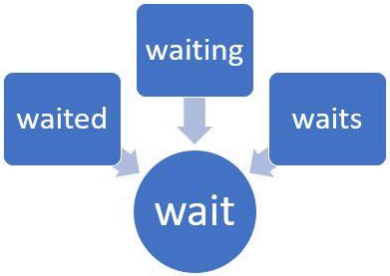
It can both have a positive or negative impact depending on the domain and the set of stopwords that are removed.

A simple sentence containing some stopwords that can be removed

Pre-processing

Stemming: reduce words to their stem, i.e., the root of the word. For example, the stem of the word *running* is *run*.

It reduces vocabulary size. The same preprocessing step should be done at training and inference phases.



GENSIM
topic modelling for humans

```python
# after pip install gensim

from gensim.test.utils import common_texts
from gensim.models import Word2Vec

model = Word2Vec(sentences=common_texts, vector_size=100, window=5, min_count=1, workers=4)
model.save("word2vec.model")
```

https://radimrehurek.com/gensim/models/word2vec.html - (latest access: October 2021)

Additional reading on Word2Vec

- Efficient estimation of word representations in vector space. T. Mikolov, K. Chen, G. Corrado, J. Dean. ICLR (Workshop Poster) 2013
- Download and read the paper: https://arxiv.org/pdf/1301.3781.pdf

Additional reading on Word2Vec optimizations

- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'13). Curran Associates Inc., Red Hook, NY, USA, 3111–3119.
- Download and read the paper: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

Evaluation process

- The assessment of the quality of NLP system can be either intrinsic or extrinsic
- Intrinsic evaluation
  - Outputs are evaluated against pre-determined ground truth
- Extrinsic evaluation
  - Outputs evaluation is based on their impact on the performance of other NLP systems or in specific NLP tasks

Cosine distance

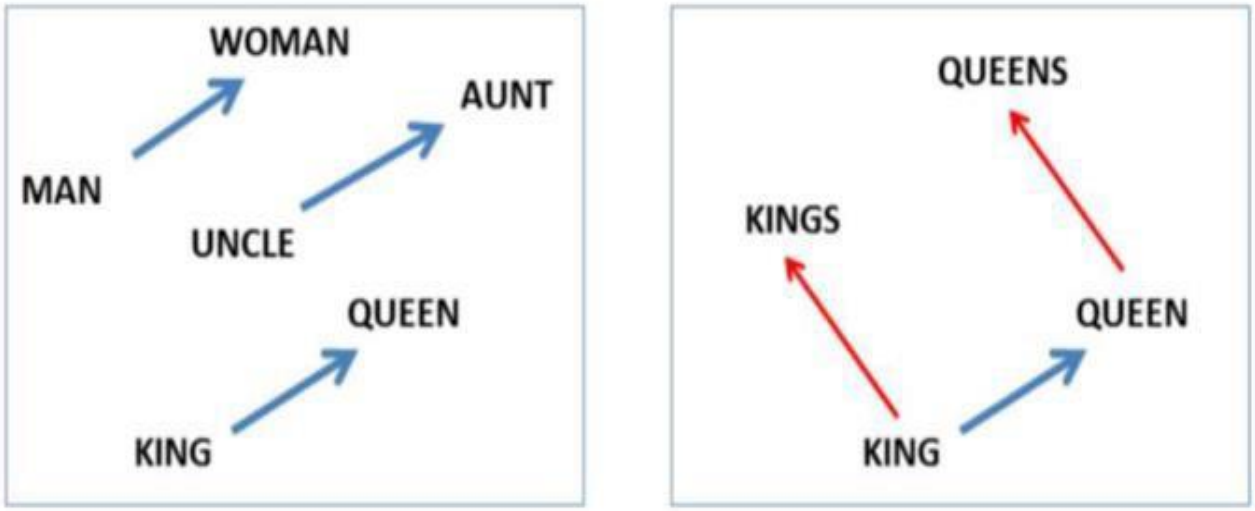
$$Cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \ \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2} \ \sqrt{\sum_1^n b_i^2}}$$

The word analogy task

- Example of extrinsic evaluation
  - Find word analogies
  - Evaluate word vectors by how well their cosine distance after addition captures intuitive semantic and syntactic analogy questions
- Question
  - Man : woman :: king : ?
- Answer
  - queen

Efficient estimation of word representations in vector space. T Mikolov, K Chen, G Corrado, J Dean. ICLR (Workshop Poster) 2013

The word analogy task



Efficient estimation of word representations in vector space. T Mikolov, K Chen, G Corrado, J Dean. ICLR (Workshop Poster) 2013

The word analogy task

Expression	*Nearest token*
Paris - France + Italy	Rome
bigger - big + cold	colder
sushi - Japan + Germany	bratwurst
Cu - copper + gold	Au
Windows - Microsoft + Google	Android
Montreal Canadiens - Montreal + Toronto	Toronto Maple Leafs

Efficient estimation of word representations in vector space. T Mikolov, K Chen, G Corrado, J Dean. ICLR (Workshop Poster) 2013

Pre-trained embedding models

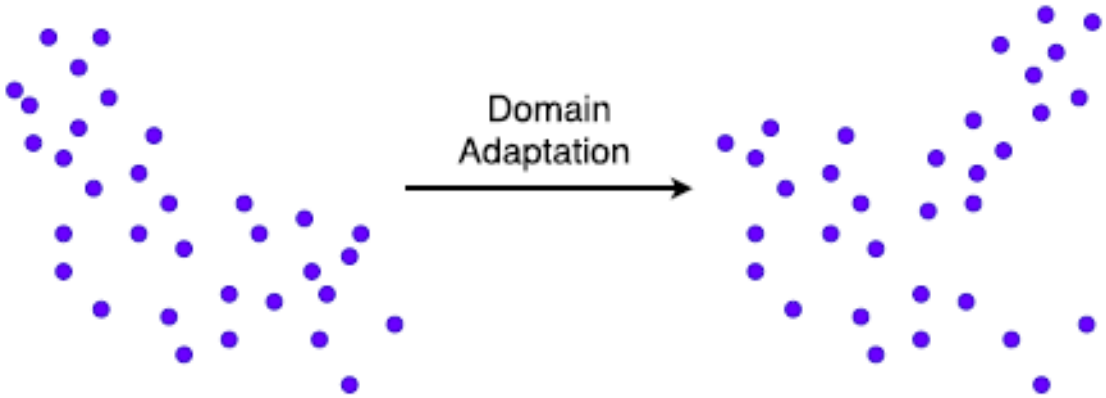
- Pretrained word embeddings are embeddings learned in one task that are used for solving another similar task
  - Transfer learning paradigm
- Pretrained embeddings are
  - trained on large datasets (e.g., Wikipedia)
  - stored and shared
  - used for solving other related tasks

W2V: code and pre-trained model examples

- Source code
  - https://code.google.com/archive/p/word2vec/
- Pre-trained on Wikipedia corpus
  - General-purpose
  - https://wikipedia2vec.github.io/wikipedia2vec/pretrained/
- Pre-trained on Google News
  - Domain-specific
  - https://mccormickml.com/2016/04/12/googles-pretrained-word2vec-model-in-python/

Efficient estimation of word representations in vector space. T Mikolov, K Chen, G Corrado, J Dean. ICLR (Workshop Poster) 2013

Domain adaptation



L. Cagliero, M. La Quatra: Inferring Multilingual Domain-Specific Word Embeddings From Large Document Corpora. IEEE Access. 2021.

Domain adaptation

- Model specialization to a particular domain
- Useful when
  - There is a lack of training data
    - Noisy outputs
  - General-purpose embedding models perform poorly
    - they fail to capture specific local meanings within a domain

Domain Adapted Word Embeddings for Improved Sentiment Classification Prathusha K Sarma, Yingyu Liang and William A Sethares. ACL 2018.

Domain adaptation

- Example: sentiment analysis
  - word “alcohol”
    - has a somewhat neutral to a mildly positive tone in the common crawl corpus (https://commoncrawl.org/)
    - has a strongly negative sentiment in a substance use disorder (SUD) dataset (https://www.ncsl.org/research/health/substance-use-disorder-sud-treatment-database.aspx)

Domain Adapted Word Embeddings for Improved Sentiment Classification. Prathusha K Sarma, Yingyu Liang and William A Sethares. ACL 2018.

Domain adaptation

- Refinement of a pretrained model
  - Keep training the general-purpose model on a smaller set of domain-specific data
  - Instrumental for a specific task or domain
- It yields rotations or translations of the original word vectors
  - Domain-specific words are mainly involved

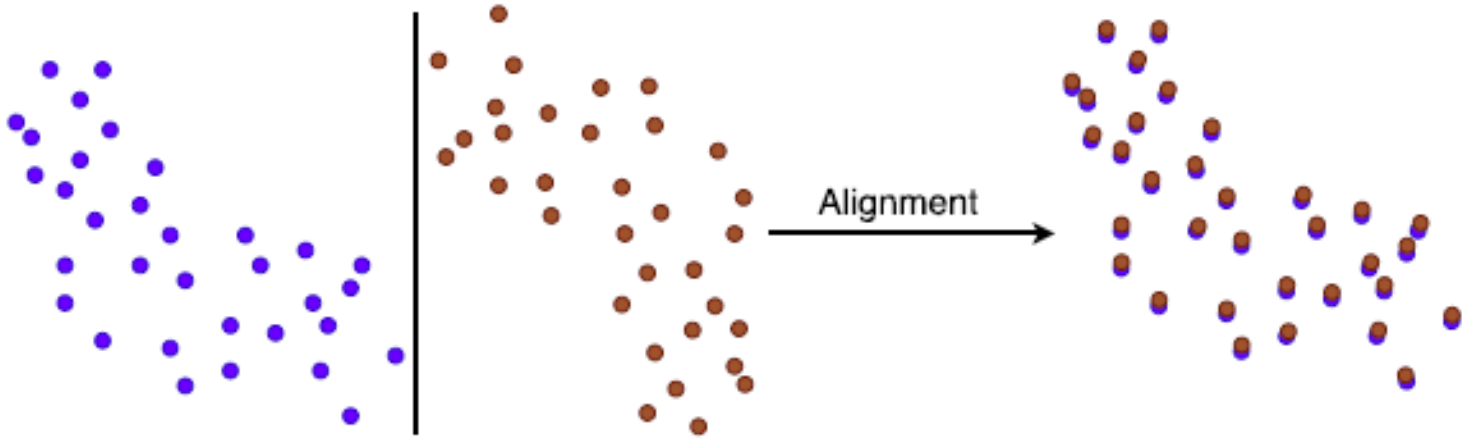
Domain Adapted Word Embeddings for Improved Sentiment Classification. Prathusha K Sarma, Yingyu Liang and William A Sethares. ACL 2018.

Domain adaptation

- Commonly addressed by using heuristic approaches to identify pivot words
  - Words that are frequently used in a specific domain
- Already applied to several domains
  - Finance, healthcare, industry 4.0

Domain Adapted Word Embeddings for Improved Sentiment Classification. Prathusha K Sarma, Yingyu Liang and William A Sethares. ACL 2018.

Bilingual vector alignment



L. Cagliero, M. La Quatra: Inferring Multilingual Domain-Specific Word Embeddings From Large Document Corpora. IEEE Access. 2021.

Bilingual embedding alignment

- Word embedding are language-specific
- Translations of the same words in different languages are not aligned
- Bilingual alignment: map words of a source language to the corresponding ones of the target language
  - Useful for machine translation

Domain Adapted Word Embeddings for Improved Sentiment Classification Prathusha K Sarma, Yingyu Liang and William A Sethares. ACL 2018.

Bilingual embedding alignment

- Unsupervised approaches
  - Learn a (non-linear) transformation from the source to the target by assuming an empirical distribution in the embedding models
- Supervised approaches
  - Given a bilingual lexicon infer a mapping function and apply to the remaining words

Domain Adapted Word Embeddings for Improved Sentiment Classification Prathusha K Sarma, Yingyu Liang and William A Sethares. ACL 2018.

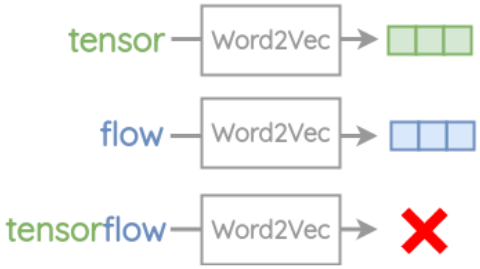
Word2Vec: main drawbacks

- Unable to model sequences of words
  - Solution: Use RNN or LSTM architectures
- Unable to consider out-of-vocabulary words (frequent in morphologically rich languages)
  - Solution: FastText
- Ignores global word co-occurrence
  - Solution: GloVe
- Not designed for encoding contextual representations
  - Solution: BERT

Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The author-topic model for authors and documents. In Proceedings of the 20th conference on Uncertainty in artificial intelligence (UAI '04). AUAI Press, Arlington, Virginia, USA, 487–494.

FastText

- Limitations of Word2Vec models
  - **Out of Vocabulary**: One word is not part of the vocabulary?
  - **Morphology**: A word is presented with a different suffix/prefix
    - E.g., eat**en**
    - Some word forms rarely occur in the corpus



https://amitness.com/2020/06/fasttext-embeddings/

FastText uses subword models representing each word as:

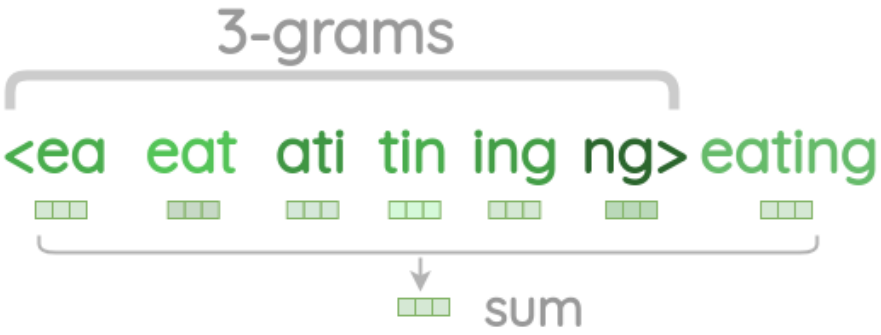
- Itself (e.g., going)
- A bag of constituent n-grams (e.g., <go, goi, oin, ing, ng>)

FastText

- Word embedding model proposed by Facebook researchers
- Key ideas
  - overcome the out-of-vocabulary issue by considering sub-words (beyond words) as textual units
  - Apply a character n-gram based model
  - Incorporate sub-word information in the embedding space

Enriching Word Vectors with Subword Information Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov. Transactions of the ACL. 2017
DOI: 10.1162/tacl_a_00051

FastText

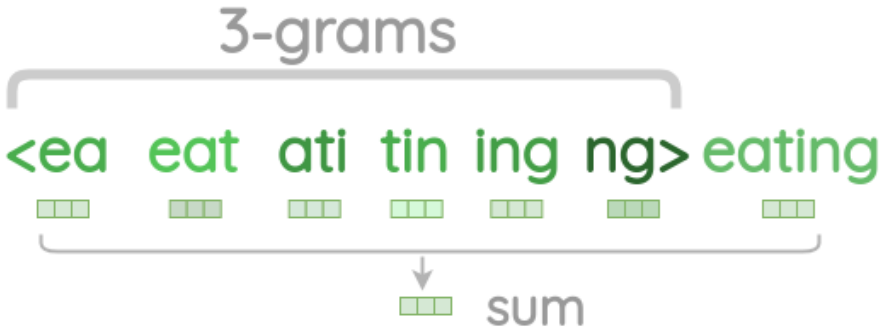


https://amitness.com/2020/06/fasttext-embeddings/

< and > are special symbols to indicate the beginning or the end of a word

The number of letters (**n**-grams) to create subword is a an hyperparameter of FastText

FastText



https://amitness.com/2020/06/fasttext-embeddings/

The embedding of a given word is obtained by summing up all the constituent n-grams

FastText: sub-word generation

Given a word generate character n-grams of length 3 to 6 present in it:

1. **Add angular bracket at the beginning and at the end**

FastText: sub-word generation

Given a word generate character n-grams of length 3 to 6 present in it:

1. Add angular bracket at the beginning and at the end
2. **Generate character n-grams of length n (sliding window)**

FastText: sub-word generation

Given a word generate character n-grams of length 3 to 6 present in it:

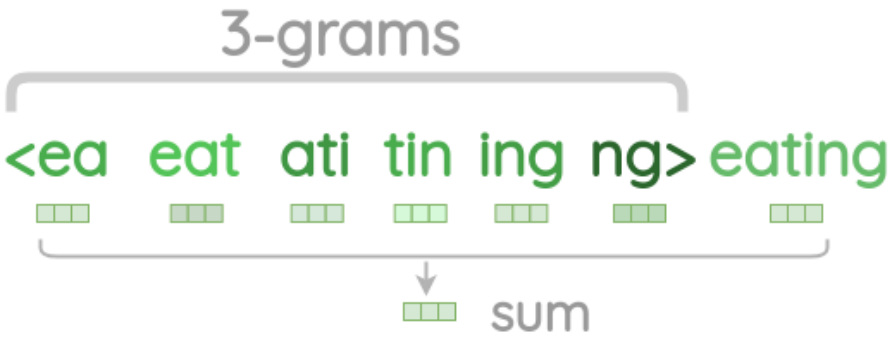
1. Add angular bracket at the beginning and at the end
2. **Generate character n-grams of length n (sliding window)**

Word	Length(n)	Character n-grams
eating	3	<ea, eat, ati, tin, ing, ng>
eating	4	<eat, eati, atin, ting, ing>
eating	5	<eati, eatin, ating, ting>
eating	6	<eatin, eating, ating>

FastText: sub-word generation

Given a word generate character n-grams of length 3 to 6 present in it:

1. Add angular bracket at the beginning and at the end
2. Generate character n-grams of length n (sliding window)
3. **Compute word embedding by summing character n-grams**



FastText: the subword model

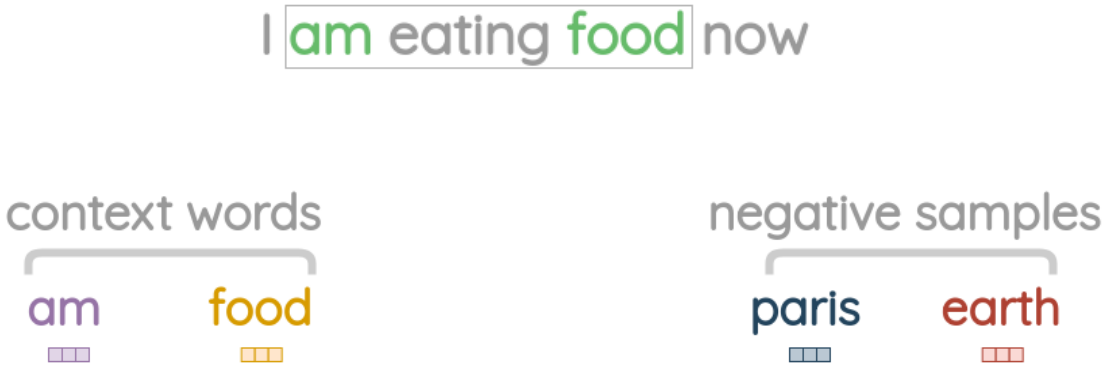
- Word Where
  - “Wh” (bigram)
  - “Whe” (trigram)
  - “Her” (trigram)
  - “Ere” (trigram)
  - “Re” (bigram)
- Notice that trigram “Her” differs from word <Her>

Enriching Word Vectors with Subword Information Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov. Transactions of the ACL. 2017
DOI: 10.1162/tacl_a_00051

FastText

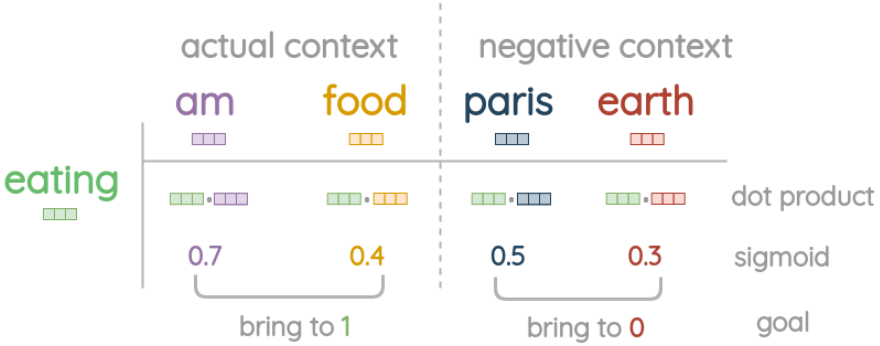
The remaining algorithm steps are similar to Word2Vec:

- Context words (sliding window)
- Negative sampling
- Skip-Gram training



FastText

- The remaining part of the algorithm are similar to Word2Vec:
  - Context words (sliding window)
  - Negative sampling
  - Skip-Gram training



FastText

Much'ananayakapushasqakupuniñataqsunamá

Much'a -na -naya -ka -pu -sha -sqa -ku -puni -ña -taq -suna -má

"So they really always have been kissing each other then"

```
Much'a   to kiss
-na      expresses obligation, lost in translation
-naya    expresses desire
-ka      diminutive
-pu      reflexive (kiss 'eachother')
-sha     progressive (kiss'ing')
-sqa     declaring something the speaker has not personally witnessed
-ku      3rd person plural (they kiss)
-puni    definitive (really')
-ña      always
-taq     statement of contrast (...then)
-suna    expressing uncertainty (So...)
-má      expressing that the speaker is surprised
```

	Singular+neut	Plural+neut	
Nominative	предложение	предложения	sentence (s)
Genitive	предложения	предложений	(of) sentence (s)
Dative	предложению	предложениям	(to) sentence (s)
Accusative	предложение	предложения	sentence (s)
Instrumental	предложением	предложениями	(by) sentence (s)
Prepositional	предложении	предложениях	(in/at) sentence (s)

Enriching Word Vectors with Subword Information Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov. Transactions of the ACL. 2017
DOI: 10.1162/tacl_a_00051

FastText: the subword model

- A word is represented by the sum of the character n-gram based vectors that compose it

$$s(w, c) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g^\top \mathbf{v}_c.$$

where w is a word, c is the word context, z_g is the vector representation of n-gram g, and G_w is the set of n-grams appearing in w

- N varies from 3 to 6
  - Shorter n-grams better capture syntactic information
  - Longer n-grams better capture semantic information

Enriching Word Vectors with Subword Information Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov. Transactions of the ACL. 2017
DOI: 10.1162/tacl_a_00051

FastText pretrained models

- Trained on Common Crawl and Wikipedia
- Pretrained word vectors for 157 languages
  https://fasttext.cc/docs/en/crawl-vectors.html
- Pretrained aligned multilingual vectors for 44 languages
  https://fasttext.cc/docs/en/aligned-vectors.html

Enriching Word Vectors with Subword Information Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov. Transactions of the ACL. 2017
DOI: 10.1162/tacl_a_00051

Additional reading on FastText

- Enriching Word Vectors with Subword Information. Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov. Transactions of the ACL. 2017. DOI: 10.1162/tacl_a_00051
- Download and read the paper: https://arxiv.org/pdf/1607.04606.pdf

GloVe

- GloVe: **Glo**bal **Ve**ctors
- It focuses on modelling global corpus statistics
- Word2Vec ignores whether some context words apper more often than other
- A large matrix of co-occurrence information is constructed
  - It counts the frequency of each "word" (i.e., the rows)
  - It counts how frequently a word occurs in some "context" (i.e., the columns) within a large corpus

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014.

GloVe

- Embedding model proposed by Stanford researchers
- Motivation
  - The word-word co-occurrence matrix is rather sparse while coping with relatively short pieces of text
  - Not suitable for direct Neural Network learning
- Key idea
  - Combine prediction-based neural methods with occurrence-based ones
    - Take the best of the two worlds
    - More effective on shorter text snippets

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

GloVe algorithm

- GloVe uses a weighted least squares objective **J**
  - It minimizes the difference between the dot product of the vectors of two words and the logarithm of their number of co-occurrences:

$$J = \sum_{i,j=1}^{V} f\left(_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log {}_{ij}\right)^2$$

- $\boldsymbol{w_i}$ and $\boldsymbol{b_i}$ are the word vector and bias respectively of word ***i***
  - the same applies to j
- $\boldsymbol{X_{ij}}$ is the number of times word ***i*** occurs in the context of word ***j***
- ***f*** is a weighting function that assigns lower weights to rare and frequent co-occurrences

GloVe algorithm

- Similarly to Word2Vec:
  - It learns word-level representations
  - Word representations are independent on the context
  - It exploits statistics over raw data

- Differently from Word2Vec:
  - It is a matrix factorization technique
  - It uses global statistics over the entire corpus

The cat sat on the mat

The mat

The cat sat

	the	cat	sat	on	mat
the	0	0	0	0	0
cat	0	0	0	0	0
sat	0	0	0	0	0
on	0	0	0	0	0
mat	0	0	0	0	0

GloVe algorithm

- Similarly to Word2Vec:
  - It learns word-level representations
  - Word representations are independent on the context
  - It exploits statistics over raw data

- Differently from Word2Vec:
  - It is a matrix factorization technique
  - It uses global statistics over the entire corpus

The cat sat on the mat

The mat

The cat sat

	the	cat	sat	on	mat
the	0	**1**	**1**	**1**	**1**
cat	**1**	0	**1**	**1**	**1**
sat	**1**	**1**	0	**1**	**1**
on	**1**	**1**	**1**	0	**1**
mat	**1**	**1**	**1**	**1**	0

GloVe algorithm

- Similarly to Word2Vec:
  - It learns word-level representations
  - Word representations are independent on the context
  - It exploits statistics over raw data

- Differently from Word2Vec:
  - It is a matrix factorization technique
  - It uses global statistics over the entire corpus

The cat sat on the mat

The mat

The cat sat

	the	cat	sat	on	mat
the	0	1	1	1	**2**
cat	1	0	1	1	1
sat	1	1	0	1	1
on	1	1	1	0	1
mat	**2**	1	1	1	0

GloVe algorithm

- Similarly to Word2Vec:
  - It learns word-level representations
  - Word representations are independent on the context
  - It exploits statistics over raw data

- Differently from Word2Vec:
  - It is a matrix factorization technique
  - It uses global statistics over the entire corpus

The cat sat on the mat

The mat

The cat sat

	the	cat	sat	on	mat
the	0	**2**	**2**	1	2
cat	**2**	0	1	1	1
sat	**2**	1	0	1	1
on	1	1	1	0	1
mat	2	1	1	1	0

GloVe algorithm

- Similar to Word2Vec
  - It learns word-level representations
  - Word representations are independent on the context
  - It exploits statistics over raw data

- Unlike Word2Vec
  - It is a matrix factorization technique
  - It uses global statistics over the entire corpus

	the	cat	sat	on	mat
the	0	2	2	1	2
cat	2	0	1	1	1
sat	2	1	0	1	1
on	1	1	1	0	1
mat	2	1	1	1	0

Matrix Factorization

features

words

	f1	f2
the	0.0	**1.2**
cat	**1.2**	0.0
sat	**1.2**	0.6
on	0.6	0.6
mat	**1.2**	0.6

GloVe

	x = solid	x = gas	x = water	x = fashion
$P(x\|\text{ice})$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(x\|\text{steam})$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$\frac{P(x\|\text{ice})}{P(x\|\text{steam})}$	8.9	8.5×10^{-2}	1.36	0.96

- Co-occurrence probabilities of a probe word w_k given the selected context words w_i=*ice* and w_j=*steam* from a 6 billion token corpus

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

GloVe

	x = solid	x = gas	x = water	x = fashion
$P(x\|\text{ice})$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(x\|\text{steam})$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$\frac{P(x\|\text{ice})}{P(x\|\text{steam})}$	8.9	8.5×10^{-2}	1.36	0.96

- Noise from non-discriminative words like *water* and *fashion* are canceled out only in the probability ratios
- Values larger than 1 correlate well with properties specific to *ice*
- Values smaller than 1 correlate well with properties specific of *steam*

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

GloVe

- Idea
  - Use the ratio of co-occurrence probabilities rather than the probabilities themselves
  - They encode meaning components
- The starting point for word vector learning are the ratios of co-occurrence probabilities of words w_i, w_j, and $\tilde{w}_k$ (from a different vector space)

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

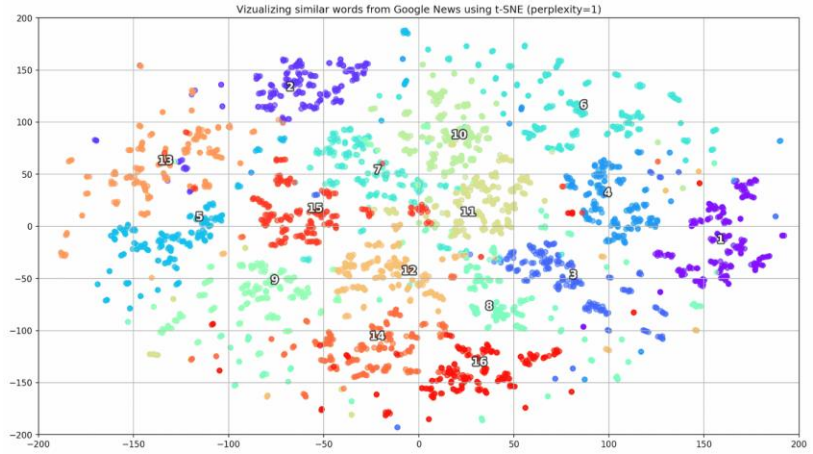
$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

Additional reading on GloVe

- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation
- Visit the GloVe project (source code and pretrained models): https://nlp.stanford.edu/projects/glove/
- Download and read the paper: https://nlp.stanford.edu/pubs/glove.pdf

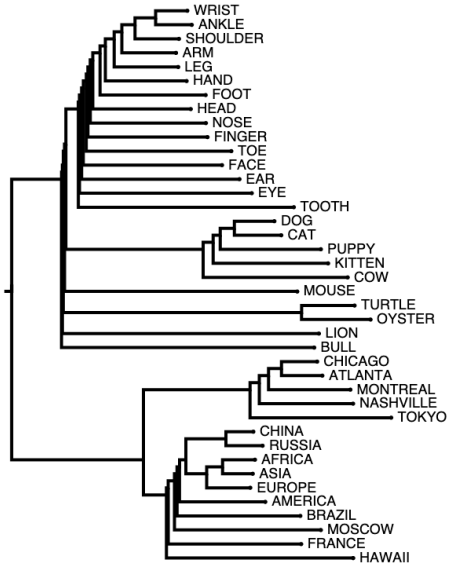
Visualization

- Visualizing embeddings is **important** to understand, apply, and improve these models of word meaning
- Standard embedding dimension are in the range of 100-300



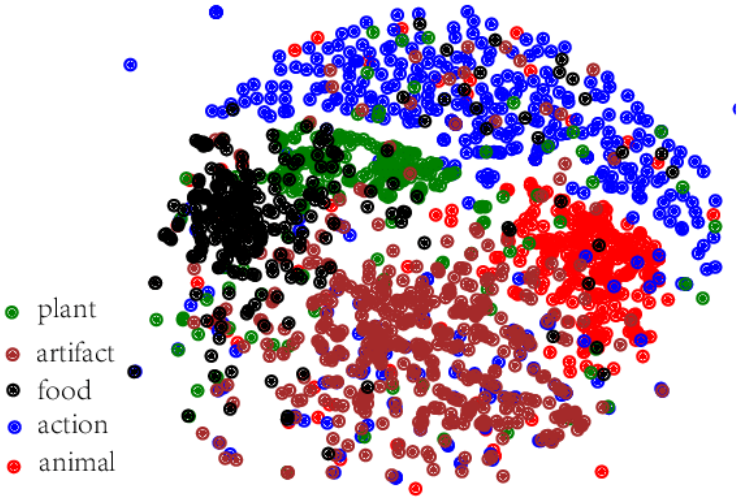
Hierarchical clustering

- One approach to visualize word embeddings is by means of hierarchical visualization
  - Given a word w find the most similar word by using the cosine similarity
  - Iteratively add words to create a clustering tree



Dimensionality reduction

- The most common visualization method is to project the embedding dimensions of a word down into 2 dimensions
- A commonly used dimensionality reduction approach is **TSNE**



Zhu, L., Xu, Z., Yang, Y., & Hauptmann, A. G. (2017). Uncovering the temporal context for video question answering. *International Journal of Computer Vision*, *124*(3), 409-421

Dimensionality reduction

1. Train word embedding model

```
# after pip install gensim

from gensim.test.utils import common_texts
from gensim.models import Word2Vec

model = Word2Vec(sentences=common_texts, vector_size=100, window=5, min_count=1, workers=4)
model.save("word2vec.model")
```


Dimensionality reduction

1. Train word embedding model
2. **Select some seed words and get nearby similar tokens**

```
# model = is the trained word2vec model
# Defining seeds
keys = ['Paris', 'Python', 'Sunday', 'Tolstoy', 'Twitter', 'bachelor', 'delivery', 'election',
'expensive','experience', 'financial', 'food', 'iOS', 'peace', 'release', 'war']

embedding_clusters = []
word_clusters = []
for word in keys:
    embeddings = []
    words = []
    for similar_word, _ in model.most_similar(word, topn=30): # set topn to lower value to improve visibility
        words.append(similar_word)
        embeddings.append(model[similar_word])
    embedding_clusters.append(embeddings)
    word_clusters.append(words)

# TSNE
from sklearn.manifold import TSNE
import numpy as np

embedding_clusters = np.array(embedding_clusters)
n, m, k = embedding_clusters.shape
tsne_model_in_2d = TSNE(perplexity=15, n_components=2, init='pca', n_iter=3500)
embeddings_in_2d = np.array(tsne_model_in_2d.fit_transform(embedding_clusters.reshape(n * m, k))).reshape(n, m,
2)
```

Adapted from: https://github.com/sismetanin/word2vec-tsne/blob/master/Visualizing%20Word2Vec%20Word%20Embeddings%20using%20t-SNE.ipynb

Dimensionality reduction

1. Train word embedding model
2. Select some seed words and get nearby similar tokens
3. **Apply TSNE to reduce dimensions to 2**

```python
# model = is the trained word2vec model
# Defining seeds
keys = ['Paris', 'Python', 'Sunday', 'Tolstoy', 'Twitter', 'bachelor', 'delivery', 'election',
'expensive','experience', 'financial', 'food', 'iOS', 'peace', 'release', 'war']

embedding_clusters = []
word_clusters = []
for word in keys:
    embeddings = []
    words = []
    for similar_word, _ in model.most_similar(word, topn=30): # set topn to lower value to improve visibility
        words.append(similar_word)
        embeddings.append(model[similar_word])
    embedding_clusters.append(embeddings)
    word_clusters.append(words)

# TSNE
from sklearn.manifold import TSNE
import numpy as np

embedding_clusters = np.array(embedding_clusters)
n, m, k = embedding_clusters.shape
tsne_model_in_2d = TSNE(perplexity=15, n_components=2, init='pca', n_iter=3500)
embeddings_in_2d = np.array(tsne_model_in_2d.fit_transform(embedding_clusters.reshape(n * m, k))).reshape(n, m,
2)
```

Adapted from: https://github.com/sismetanin/word2vec-tsne/blob/master/Visualizing%20Word2Vec%20Word%20Embeddings%20using%20t-SNE.ipynb

Dimensionality reduction

1. Train word embedding model
2. Select some seed words and get nearby similar tokens
3. Apply TSNE to reduce dimensions to 2
4. **Visualize the result by using a visualization library**

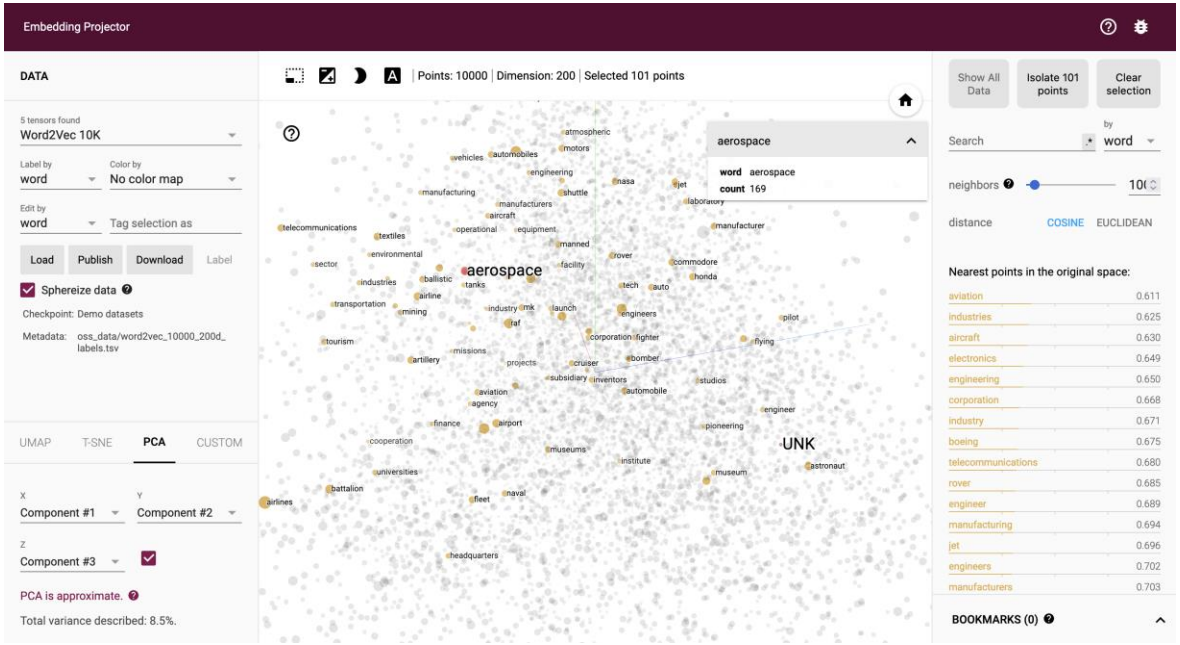
```python
import matplotlib.pyplot as plt
import matplotlib.cm as cm


def tsne_plot_similar_words(title, labels, embedding_clusters, word_clusters, a, filename=None):
    plt.figure(figsize=(16, 9))
    colors = cm.rainbow(np.linspace(0, 1, len(labels)))
    for label, embeddings, words, color in zip(labels, embedding_clusters, word_clusters, colors):
        x = embeddings[:, 0]
        y = embeddings[:, 1]
        plt.scatter(x, y, c=color, alpha=a, label=label)
        for i, word in enumerate(words):
            plt.annotate(word, alpha=0.5, xy=(x[i], y[i]), xytext=(5, 2),
                         textcoords='offset points', ha='right', va='bottom', size=8)
    plt.legend(loc=4)
    plt.title(title)
    plt.grid(True)
    if filename:
        plt.savefig(filename, format='png', dpi=150, bbox_inches='tight')
    plt.show()


tsne_plot_similar_words('W2V visualization', keys, embeddings_in_2d, word_clusters, 0.7, 'similar_words.png')
```

Adapted from: https://github.com/sismetanin/word2vec-tsne/blob/master/Visualizing%20Word2Vec%20Word%20Embeddings%20using%20t-SNE.ipynb

Visualization demo

https://projector.tensorflow.org/

- Interactive demo provided by TensorFlow
- It exploits PCA as dimensionality reduction method to show high dimensional embeddings in a 3D space



Acknowlegdements and copyright license

- Copyright licence
  - Attribution + Noncommercial + NoDerivatives

- Acknowledgements
  - I would like to thank Dr. Moreno La Quatra, who collaborated to the writing and revision of the teaching content
- Affiliation
  - The author and his staff are currently members of the Database and Data Mining Group at Dipartimento di Automatica e Informatica (Politecnico di Torino) and of the SmartData interdepartmental centre
    - https://dbdmg.polito.it
    - https://smartdata.polito.it

Thank you!

