Title: Word2vec

URL: https://en.wikipedia.org/wiki/Word2vec

PageID: 47527969

Categories: Category:2013 in artificial intelligence, Category:2013 software, Category:Artificial neural networks, Category:Free science software, Category:Natural language processing toolkits, Category:Semantic relations

-----

Supervised learning

Unsupervised learning

Semi-supervised learning

Self-supervised learning

Reinforcement learning

Meta-learning

Online learning

Batch learning

Curriculum learning

Rule-based learning

Neuro-symbolic AI

Neuromorphic engineering

Quantum machine learning

Classification

Generative modeling

Regression

Clustering

Dimensionality reduction

Density estimation

Anomaly detection

Data cleaning

AutoML

Association rules

Semantic analysis

Structured prediction

Feature engineering

Feature learning

Learning to rank

Grammar induction

Ontology learning

Multimodal learning
Apprenticeship learning
Decision trees
Ensembles Bagging Boosting Random forest
Bagging
Boosting
Random forest
k -NN
Linear regression
Naive Bayes
Artificial neural networks
Logistic regression
Perceptron
Relevance vector machine (RVM)
Support vector machine (SVM)
BIRCH
CURE
Hierarchical
k -means
Fuzzy
Expectation–maximization (EM)
DBSCAN
OPTICS
Mean shift
Factor analysis
CCA
ICA
LDA
NMF
PCA
PGD
t-SNE
SDL
Graphical models Bayes net Conditional random field Hidden Markov
Bayes net
Conditional random field
Hidden Markov
RANSAC

k -NN
Local outlier factor
Isolation forest
Autoencoder
Deep learning
Feedforward neural network
Recurrent neural network LSTM GRU ESN reservoir computing
LSTM
GRU
ESN
reservoir computing
Boltzmann machine Restricted
Restricted
GAN
Diffusion model
SOM
Convolutional neural network U-Net LeNet AlexNet DeepDream
U-Net
LeNet
AlexNet
DeepDream
Neural field Neural radiance field Physics-informed neural networks
Neural radiance field
Physics-informed neural networks
Transformer Vision
Vision
Mamba
Spiking neural network
Memtransistor
Electrochemical RAM (ECRAM)
Q-learning
Policy gradient
SARSA
Temporal difference (TD)
Multi-agent Self-play
Self-play
Active learning
Crowdsourcing

Human-in-the-loop

Mechanistic interpretability

RLHF

Coefficient of determination

Confusion matrix

Learning curve

ROC curve

Kernel machines

Bias–variance tradeoff

Computational learning theory

Empirical risk minimization

Occam learning

PAC learning

Statistical learning

VC theory

Topological deep learning

AAAI

ECML PKDD

NeurIPS

ICML

ICLR

IJCAI

ML

JMLR

Glossary of artificial intelligence

List of datasets for machine-learning research List of datasets in computer vision and image processing

List of datasets in computer vision and image processing

Outline of machine learning

v

t

e

Word2vec is a technique in natural language processing for obtaining vector representations of words. These vectors capture information about the meaning of the word based on the surrounding words. The word2vec algorithm estimates these representations by modeling text in a large corpus . Once trained, such a model can detect synonymous words or suggest additional words for a partial sentence. Word2vec was developed by Tomáš Mikolov , Kai Chen, Greg Corrado, Ilya Sutskever and Jeff Dean at Google, and published in 2013. [1] [2]

Word2vec represents a word as a high-dimension vector of numbers which capture relationships between words. In particular, words which appear in similar contexts are mapped to vectors which are nearby as measured by cosine similarity . This indicates the level of semantic similarity between the words, so for example the vectors for walk and ran are nearby, as are those for "but" and "however", and "Berlin" and "Germany".

Approach

Word2vec is a group of related models that are used to produce word embeddings . These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a mapping of the set of words to a vector space , typically of several hundred dimensions , with each unique word in the corpus being assigned a vector in the space.

Word2vec can use either of two model architectures to produce these distributed representations of words: continuous bag of words (CBOW) or continuously sliding skip-gram. In both architectures, word2vec considers both individual words and a sliding context window as it iterates over the corpus.

The CBOW can be viewed as a ‘fill in the blank’ task, where the word embedding represents the way the word influences the relative probabilities of other words in the context window. Words which are semantically similar should influence these probabilities in similar ways, because semantically similar words should be used in similar contexts. The order of context words does not influence prediction (bag of words assumption).

In the continuous skip-gram architecture, the model uses the current word to predict the surrounding window of context words. [1] [2] The skip-gram architecture weighs nearby context words more heavily than more distant context words. According to the authors' note, [3] CBOW is faster while skip-gram does a better job for infrequent words.

After the model is trained, the learned word embeddings are positioned in the vector space such that words that share common contexts in the corpus — that is, words that are semantically and syntactically similar — are located close to one another in the space. [1] More dissimilar words are located farther from one another in the space. [1]

Mathematical details

This section is based on expositions. [4] [5]

A corpus is a sequence of words. Both CBOW and skip-gram are methods to learn one vector per word appearing in the corpus.

Let V {\displaystyle V} ("vocabulary") be the set of all words appearing in the corpus C {\displaystyle C} . Our goal is to learn one vector v w ∈ R n {\displaystyle v_{w}\in \mathbb {R} ^{n}} for each word w ∈ V {\displaystyle w\in V} .

The idea of skip-gram is that the vector of a word should be close to the vector of each of its neighbors. The idea of CBOW is that the vector-sum of a word's neighbors should be close to the vector of the word.

Continuous bag-of-words (CBOW)

The idea of CBOW is to represent each word with a vector, such that it is possible to predict a word using the sum of the vectors of its neighbors. Specifically, for each word w i {\displaystyle w_{i}} in the corpus, the one-hot encoding of the word is used as the input to the neural network. The output of the neural network is a probability distribution over the dictionary, representing a prediction of individual words in the neighborhood of w i {\displaystyle w_{i}} . The objective of training is to maximize ∑ i ln ■ Pr (w i | w j : j ∈ i + N) {\displaystyle \sum _{i}\ln \Pr(w_{i}|w_{j}:j\in i+N)} .

For example, if we want each word in the corpus to be predicted by every other word in a small span of 4 words. The set of relative indexes of neighbor words will be: N = { − 2 , − 1 , + 1 , + 2 } {\displaystyle N=\{-2,-1,+1,+2\}} , and the objective is ∑ i ln ■ Pr (w i | w i − 2 , w i − 1 , w i + 1 , w i + 2) {\displaystyle \sum _{i}\ln \Pr(w_{i}|w_{i-2},w_{i-1},w_{i+1},w_{i+2})} .

In standard bag-of-words, a word's context is represented by a word-count (aka a word histogram) of its neighboring words. For example, the "sat" in "the cat sat on the mat" is represented as {"the": 2, "cat": 1, "on": 1}. Note that the last word "mat" is not used to represent "sat", because it is outside the neighborhood N = { − 2 , − 1 , + 1 , + 2 } {\displaystyle N=\{-2,-1,+1,+2\}} .

In continuous bag-of-words, the histogram is multiplied by a matrix V {\displaystyle V} to obtain a continuous representation of the word's context. The matrix V {\displaystyle V} is also called a dictionary . Its columns are the word vectors. It has D {\displaystyle D} columns, where D {\displaystyle D} is the size of the dictionary. Let d {\displaystyle d} be the length of each word vector. We have V ∈ R d × D {\displaystyle V\in \mathbb {R} ^{d\times D}} .

For example, multiplying the word histogram {"the": 2, "cat": 1, "on": 1} with V {\displaystyle V} , we obtain 2 v the + v cat + v on {\displaystyle 2v_{\text{the}}+v_{\text{cat}}+v_{\text{on}}} .

This is then multiplied with another matrix V ′ {\displaystyle V'} of shape R D × d {\displaystyle \mathbb {R} ^{D\times d}} . Each row of it is a word vector v ′ {\displaystyle v'} . This results in a vector of length D {\displaystyle D} , one entry per dictionary entry. Then, apply the softmax to obtain a probability distribution over the dictionary.

This system can be visualized as a neural network, similar in spirit to an autoencoder , of architecture linear-linear- softmax , as depicted in the diagram. The system is trained by gradient descent to minimize the cross-entropy loss .

In full formula, the cross-entropy loss is: − ∑ i ln ■ e v w i ′ ⋅ (∑ j ∈ i + N v w j) ∑ w ′ e v w ′ ′ ⋅ (∑ j ∈ i + N v w j) {\displaystyle -\sum _{i}\ln {\frac {e^{v_{w_{i}}'\cdot (\sum _{j\in i+N}v_{w_{j}})}}{\sum _{w'}e^{v_{w'}'\cdot (\sum _{j\in i+N}v_{w_{j}})}}}} where the outer summation ∑ i {\displaystyle \sum _{i}} is over the words in a corpus, the quantity ∑ j ∈ i + N v w j {\displaystyle \sum _{j\in i+N}v_{w_{j}}} is the sum of a word's neighbors' vectors, etc.

Once such a system is trained, we have two trained matrices V , V ′ {\displaystyle V,V'} . Either the column vectors of V {\displaystyle V} or the row vectors of V ′ {\displaystyle V'} can serve as the dictionary. For example, the word "sat" can be represented as either the "sat"-th column of V {\displaystyle V} or the "sat"-th row of V ′ {\displaystyle V'} . It is also possible to simply define V ′ = V ■ {\displaystyle V'=V^{\top }} , in which case there would no longer be a choice.

Skip-gram

The idea of skip-gram is to represent each word with a vector, such that it is possible to predict the vectors of its neighbors using the vector of a word.

The architecture is still linear-linear-softmax, the same as CBOW, but the input and the output are switched. Specifically, for each word w i {\displaystyle w_{i}} in the corpus, the one-hot encoding of the word is used as the input to the neural network. The output of the neural network is a probability distribution over the dictionary, representing a prediction of individual words in the neighborhood of w i {\displaystyle w_{i}} . The objective of training is to maximize ∑ i ∑ j ∈ i + N ln ■ Pr (w j | w i) {\displaystyle \sum _{i}\sum _{j\in i+N}\ln \Pr(w_{j}|w_{i})} .

In full formula, the loss function is − ∑ i ∑ j ∈ i + N ln ■ e v w j ′ ⋅ v w i ∑ w ′ e v w ′ ′ ⋅ v w i {\displaystyle -\sum _{i}\sum _{j\in i+N}\ln {\frac {e^{v_{w_{j}}'\cdot v_{w_{i}}}}{\sum _{w'}e^{v_{w'}'\cdot v_{w_{i}}}}}} Same as CBOW, once such a system is trained, we have two trained matrices V , V ′ {\displaystyle V,V'} . Either the column vectors of V {\displaystyle V} or the row vectors of V ′ {\displaystyle V'} can serve as the dictionary. It is also possible to simply define V ′ = V ■ {\displaystyle V'=V^{\top }} , in which case there would no longer be a choice.

Essentially, skip-gram and CBOW are exactly the same in architecture. They only differ in the objective function during training.

History

During the 1980s, there were some early attempts at using neural networks to represent words and concepts as vectors. [6] [7] [8]

In 2010, Tomáš Mikolov (then at Brno University of Technology) with co-authors applied a simple recurrent neural network with a single hidden layer to language modelling. [9]

Word2vec was created, patented, [10] and published in 2013 by a team of researchers led by Mikolov at Google over two papers. [1] [2] The original paper was rejected by reviewers for ICLR conference 2013. It also took months for the code to be approved for open-sourcing. [11] Other researchers helped analyse and explain the algorithm. [4]

Embedding vectors created using the Word2vec algorithm have some advantages compared to earlier algorithms [1] such as those using n-grams and latent semantic analysis . GloVe was developed by a team at Stanford specifically as a competitor, and the original paper noted multiple improvements of GloVe over word2vec. [12] Mikolov argued that the comparison was unfair as GloVe was trained on more data, and that the fastText project showed that word2vec is superior when trained on the same data. [13] [11]

As of 2022, the straight Word2vec approach was described as "dated". Transformer -based models, such as ELMo and BERT , which add multiple neural-network attention layers on top of a word embedding model similar to Word2vec, have come to be regarded as the state of the art in natural language processing. [14]

Parameterization

Results of word2vec training can be sensitive to parametrization . The following are some important parameters in word2vec training.

Training algorithm

A Word2vec model can be trained with hierarchical softmax and/or negative sampling. To approximate the conditional log-likelihood a model seeks to maximize, the hierarchical softmax method uses a Huffman tree to reduce calculation. The negative sampling method, on the other hand, approaches the maximization problem by minimizing the log-likelihood of sampled negative instances. According to the authors, hierarchical softmax works better for infrequent words while negative sampling works better for frequent words and better with low dimensional vectors. [3] As training epochs increase, hierarchical softmax stops being useful. [15]

Sub-sampling

High-frequency and low-frequency words often provide little information. Words with a frequency above a certain threshold, or below a certain threshold, may be subsampled or removed to speed up training. [16]

Dimensionality

Quality of word embedding increases with higher dimensionality. But after reaching some point, marginal gain diminishes. [1] Typically, the dimensionality of the vectors is set to be between 100 and 1,000.

Context window

The size of the context window determines how many words before and after a given word are included as context words of the given word. According to the authors' note, the recommended value is 10 for skip-gram and 5 for CBOW. [3]

Extensions

There are a variety of extensions to word2vec.

doc2vec

doc2vec, generates distributed representations of variable-length pieces of texts, such as sentences, paragraphs, or entire documents. [17] [18] doc2vec has been implemented in the C , Python and Java / Scala tools (see below), with the Java and Python versions also supporting inference of document embeddings on new, unseen documents.

doc2vec estimates the distributed representations of documents much like how word2vec estimates representations of words: doc2vec utilizes either of two model architectures, both of which are allegories to the architectures used in word2vec. The first, Distributed Memory Model of Paragraph Vectors (PV-DM), is identical to CBOW other than it also provides a unique document identifier as a piece of additional context. The second architecture, Distributed Bag of Words version of Paragraph Vector (PV-DBOW), is identical to the skip-gram model except that it attempts to predict the window of surrounding context words from the paragraph identifier instead of the current word. [17]

doc2vec also has the ability to capture the semantic ‘meanings’ for additional pieces of ‘context’ around words; doc2vec can estimate the semantic embeddings for speakers or speaker attributes, groups, and periods of time. For example, doc2vec has been used to estimate the political positions of political parties in various Congresses and Parliaments in the U.S. and U.K., [19] respectively, and various governmental institutions. [20]

top2vec

Another extension of word2vec is top2vec, which leverages both document and word embeddings to estimate distributed representations of topics. [21] [22] top2vec takes document embeddings learned from a doc2vec model and reduces them into a lower dimension (typically using UMAP). The space of documents is then scanned using HDBSCAN , [23] and clusters of similar documents are found. Next, the centroid of documents identified in a cluster is considered to be that cluster's topic vector. Finally, top2vec searches the semantic space for word embeddings located near to the topic vector to ascertain the 'meaning' of the topic. [21] The word with embeddings most similar to the topic vector might be assigned as the topic's title, whereas far away word embeddings may be considered unrelated.

As opposed to other topic models such as LDA , top2vec provides canonical ‘distance’ metrics between two topics, or between a topic and another embeddings (word, document, or otherwise). Together with results from HDBSCAN, users can generate topic hierarchies, or groups of related topics and subtopics.

Furthermore, a user can use the results of top2vec to infer the topics of out-of-sample documents. After inferring the embedding for a new document, must only search the space of topics for the closest topic vector.

BioVectors

An extension of word vectors for n-grams in biological sequences (e.g. DNA , RNA , and proteins) for bioinformatics applications has been proposed by Asgari and Mofrad. [24] Named bio-vectors (BioVec) to refer to biological sequences in general with protein-vectors (ProtVec) for proteins (amino-acid sequences) and gene-vectors (GeneVec) for gene sequences, this representation can be widely used in applications of machine learning in proteomics and genomics. The results suggest that BioVectors can characterize biological sequences in terms of biochemical and biophysical interpretations of the underlying patterns. [24] A similar variant, dna2vec, has shown that there is correlation between Needleman–Wunsch similarity score and cosine similarity of dna2vec word vectors. [25]

Radiology and intelligent word embeddings (IWE)

An extension of word vectors for creating a dense vector representation of unstructured radiology reports has been proposed by Banerjee et al. [26] One of the biggest challenges with Word2vec is how to handle unknown or out-of-vocabulary (OOV) words and morphologically similar words. If the Word2vec model has not encountered a particular word before, it will be forced to use a random vector, which is generally far from its ideal representation. This can particularly be an issue in domains like medicine where synonyms and related words can be used depending on the preferred style of radiologist, and words may have been used infrequently in a large corpus.

IWE combines Word2vec with a semantic dictionary mapping technique to tackle the major challenges of information extraction from clinical texts, which include ambiguity of free text narrative style, lexical variations, use of ungrammatical and telegraphic phases, arbitrary ordering of words, and frequent appearance of abbreviations and acronyms. Of particular interest, the IWE model (trained on the one institutional dataset) successfully translated to a different institutional dataset which demonstrates good generalizability of the approach across institutions.

Analysis

The reasons for successful word embedding learning in the word2vec framework are poorly understood. Goldberg and Levy point out that the word2vec objective function causes words that occur in similar contexts to have similar embeddings (as measured by cosine similarity) and note that this is in line with J. R. Firth's distributional hypothesis . However, they note that this explanation is "very hand-wavy" and argue that a more formal explanation would be preferable. [4]

Levy et al. (2015) [27] show that much of the superior performance of word2vec or similar embeddings in downstream tasks is not a result of the models per se, but of the choice of specific hyperparameters. Transferring these hyperparameters to more 'traditional' approaches yields similar performances in downstream tasks. Arora et al. (2016) [28] explain word2vec and related algorithms as performing inference for a simple generative model for text, which involves a random walk generation process based upon loglinear topic model. They use this to explain some properties of word embeddings, including their use to solve analogies.

Preservation of semantic and syntactic relationships

The word embedding approach is able to capture multiple different degrees of similarity between words. Mikolov et al. (2013) [29] found that semantic and syntactic patterns can be reproduced using vector arithmetic. Patterns such as "Man is to Woman as Brother is to Sister" can be generated through algebraic operations on the vector representations of these words such that the vector representation of "Brother" - "Man" + "Woman" produces a result which is closest to the vector representation of "Sister" in the model. Such relationships can be generated for a range of semantic relations (such as Country–Capital) as well as syntactic relations (e.g. present tense–past tense).

This facet of word2vec has been exploited in a variety of other contexts. For example, word2vec has been used to map a vector space of words in one language to a vector space constructed from another language. Relationships between translated words in both spaces can be used to assist with machine translation of new words. [30]

Assessing the quality of a model

Mikolov et al. (2013) [1] developed an approach to assessing the quality of a word2vec model which draws on the semantic and syntactic patterns discussed above. They developed a set of 8,869 semantic relations and 10,675 syntactic relations which they use as a benchmark to test the accuracy of a model. When assessing the quality of a vector model, a user may draw on this accuracy test which is implemented in word2vec, [31] or develop their own test set which is meaningful to the corpora which make up the model. This approach offers a more challenging test than simply arguing that the words most similar to a given test word are intuitively plausible. [1]

Parameters and model quality

The use of different model parameters and different corpus sizes can greatly affect the quality of a word2vec model. Accuracy can be improved in a number of ways, including the choice of model architecture (CBOW or Skip-Gram), increasing the training data set, increasing the number of vector dimensions, and increasing the window size of words considered by the algorithm. Each of these improvements comes with the cost of increased computational complexity and therefore increased model generation time. [1]

In models using large corpora and a high number of dimensions, the skip-gram model yields the highest overall accuracy, and consistently produces the highest accuracy on semantic relationships, as well as yielding the highest syntactic accuracy in most cases. However, the CBOW is less computationally expensive and yields similar accuracy results. [1]

Overall, accuracy increases with the number of words used and the number of dimensions. Mikolov et al. [1] report that doubling the amount of training data results in an increase in computational complexity equivalent to doubling the number of vector dimensions.

Altszyler and coauthors (2017) studied Word2vec performance in two semantic tests for different corpus size. [32] They found that Word2vec has a steep learning curve , outperforming another word-embedding technique, latent semantic analysis (LSA), when it is trained with medium to large corpus size (more than 10 million words). However, with a small training corpus, LSA showed better performance. Additionally they show that the best parameter setting depends on the task and the training corpus. Nevertheless, for skip-gram models trained in medium size corpora, with 50 dimensions, a window size of 15 and 10 negative samples seems to be a good parameter setting.

See also

Autoencoder

Document-term matrix

Feature extraction

Feature learning

Language model § Neural models

Vector space model

Thought vector

fastText

GloVe

ELMo

BERT (language model)

Normalized compression distance

References

External links

Wikipedia2Vec [1] (introduction)

Implementations

C

C#

Python (Spark)

Python (TensorFlow)

Python (Gensim)

Java/Scala

R

v

t

e

AI-complete

Bag-of-words

n -gram Bigram Trigram

Bigram

Trigram

Computational linguistics

Natural language understanding

Stop words

Text processing

Argument mining

Collocation extraction

Concept mining

Coreference resolution

Deep linguistic processing

Distant reading

Information extraction

Named-entity recognition

Ontology learning

Parsing Semantic parsing Syntactic parsing

Semantic parsing

Syntactic parsing

Part-of-speech tagging

Semantic analysis

Semantic role labeling

Semantic decomposition

Semantic similarity

Sentiment analysis

Terminology extraction

Text mining

Textual entailment

Truecasing

Word-sense disambiguation

Word-sense induction

Compound-term processing

Lemmatisation

Lexical analysis

Text chunking

Stemming

Sentence segmentation

Word segmentation

Multi-document summarization

Sentence extraction

Text simplification

Computer-assisted

Example-based

Rule-based

Statistical

Transfer-based

Neural

BERT
Document-term matrix
Explicit semantic analysis
fastText
GloVe
Language model (large)
Latent semantic analysis
Seq2seq
Word embedding
Word2vec
Corpus linguistics
Lexical resource
Linguistic Linked Open Data
Machine-readable dictionary
Parallel text
PropBank
Semantic network
Simple Knowledge Organization System
Speech corpus
Text corpus
Thesaurus (information retrieval)
Treebank
Universal Dependencies
BabelNet
Bank of English
DBpedia
FrameNet
Google Ngram Viewer
UBY
WordNet
Wikidata
Speech recognition
Speech segmentation
Speech synthesis
Natural language generation
Optical character recognition
Document classification
Latent Dirichlet allocation

Pachinko allocation
Automated essay scoring
Concordancer
Grammar checker
Predictive text
Pronunciation assessment
Spell checker
Chatbot
Interactive fiction
Question answering
Virtual assistant
Voice user interface
Formal semantics
Hallucination
Natural Language Toolkit
spaCy
v
t
e
History timeline
timeline
Companies
Projects
Parameter Hyperparameter
Hyperparameter
Loss functions
Regression Bias–variance tradeoff Double descent Overfitting
Bias–variance tradeoff
Double descent
Overfitting
Clustering
Gradient descent SGD Quasi-Newton method Conjugate gradient method
SGD
Quasi-Newton method
Conjugate gradient method
Backpropagation
Attention
Convolution

Normalization Batchnorm

Batchnorm

Activation Softmax Sigmoid Rectifier

Softmax

Sigmoid

Rectifier

Gating

Weight initialization

Regularization

Datasets Augmentation

Augmentation

Prompt engineering

Reinforcement learning Q-learning SARSA Imitation Policy gradient

Q-learning

SARSA

Imitation

Policy gradient

Diffusion

Latent diffusion model

Autoregression

Adversary

RAG

Uncanny valley

RLHF

Self-supervised learning

Reflection

Recursive self-improvement

Hallucination

Word embedding

Vibe coding

Machine learning In-context learning

In-context learning

Artificial neural network Deep learning

Deep learning

Language model Large language model NMT

Large language model

NMT

Reasoning language model

Model Context Protocol

Intelligent agent

Artificial human companion

Humanity's Last Exam

Artificial general intelligence (AGI)

AlexNet

WaveNet

Human image synthesis

HWR

OCR

Computer vision

Speech synthesis 15.ai ElevenLabs

15.ai

ElevenLabs

Speech recognition Whisper

Whisper

Facial recognition

AlphaFold

Text-to-image models Aurora DALL-E Firefly Flux Ideogram Imagen Midjourney Recraft Stable Diffusion

Aurora

DALL-E

Firefly

Flux

Ideogram

Imagen

Midjourney

Recraft

Stable Diffusion

Text-to-video models Dream Machine Runway Gen Hailuo AI Kling Sora Veo

Dream Machine

Runway Gen

Hailuo AI

Kling

Sora

Veo

Music generation Riffusion Suno AI Udio

Riffusion

Suno AI

Udio

Word2vec

Seq2seq

GloVe

BERT

T5

Llama

Chinchilla AI

PaLM

GPT 1 2 3 J ChatGPT 4 4o o1 o3 4.5 4.1 o4-mini 5

1

2

3

J

ChatGPT

4

4o

o1

o3

4.5

4.1

o4-mini

5

Claude

Gemini Gemini (language model) Gemma

Gemini (language model)

Gemma

Grok

LaMDA

BLOOM

DBRX

Project Debater

IBM Watson

IBM Watsonx

Granite

PanGu-Σ

DeepSeek

Qwen

AlphaGo

AlphaZero

OpenAI Five

Self-driving car

MuZero

Action selection AutoGPT

AutoGPT

Robot control

Alan Turing

Warren Sturgis McCulloch

Walter Pitts

John von Neumann

Claude Shannon

Shun'ichi Amari

Kunihiko Fukushima

Takeo Kanade

Marvin Minsky

John McCarthy

Nathaniel Rochester

Allen Newell

Cliff Shaw

Herbert A. Simon

Oliver Selfridge

Frank Rosenblatt

Bernard Widrow

Joseph Weizenbaum

Seymour Papert

Seppo Linnainmaa

Paul Werbos

Geoffrey Hinton

John Hopfield

Jürgen Schmidhuber

Yann LeCun

Yoshua Bengio

Lotfi A. Zadeh

Stephen Grossberg

Alex Graves

James Goodnight

Andrew Ng

Fei-Fei Li

Alex Krizhevsky

Ilya Sutskever

Oriol Vinyals

Quoc V. Le

Ian Goodfellow

Demis Hassabis

David Silver

Andrej Karpathy

Ashish Vaswani

Noam Shazeer

Aidan Gomez

John Schulman

Mustafa Suleyman

Jan Leike

Daniel Kokotajlo

François Chollet

Neural Turing machine

Differentiable neural computer

Transformer Vision transformer (ViT)

Vision transformer (ViT)

Recurrent neural network (RNN)

Long short-term memory (LSTM)

Gated recurrent unit (GRU)

Echo state network

Multilayer perceptron (MLP)

Convolutional neural network (CNN)

Residual neural network (RNN)

Highway network

Mamba

Autoencoder

Variational autoencoder (VAE)

Generative adversarial network (GAN)

Graph neural network (GNN)

Category