

DATA SCIENCE HOLODECK

Glossary of Terms

This document contains a reader-friendly definitions and descriptions of some key terms of the Data Science Holodeck's language, also frequently used in scientific publications and popular media for explanations of the Artificial Intelligence artefacts.

The Glossary complements the Workflow document in a way that turns them both into a usable **AI guide** for readers with limited knowledge and experience in the subject area.

Considering the higher velocity, larger volume and wider variety of information sources and application developments in the field of AI, we keep the Glossary published on the [project's website](#) and opened for periodic updates.

List of Terms

ANN

Attention

BERT

Chunking

Embeddings

GenAI

GPT

Grounding vs Hallucinations

Hugging Face

Inference

Knowledge Graph

LangChain

LlamaIndex

LLM

NLP/NLU

NER

POS

RAG

Software Agent

SpaCy

Streamlit

Supervised, Unsupervised, and Self-Supervised Learning

Transformer

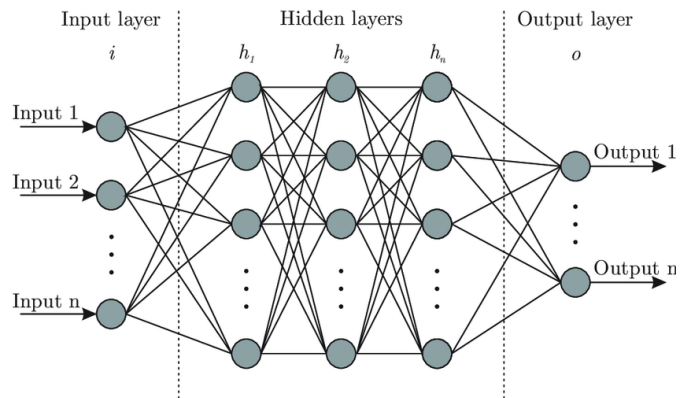
Vector Databases

Vector Similarity

ANN

ANN (Artificial Neural Networks) are software-created virtual structures, designed to model the organisation of the biological neural networks of human brain cells and to simulate the operations of biological neurons, distributing signals to one another.

ANN consists of multiple layers of interconnected nodes (artificial neurons), where the first layer receives and detects the input data (like human receptors accept the stimuli) and passes it further, the middle (hidden) layers process it (like brain neurons process the impulses), and the last layer the outputs the results of the processing and converts them into actionable objects (like human effectors implement the brain commands).



ANN, [image source](#)

Layer k	Neuron j	b_j^k	w_{1j}^k	w_{2j}^k	w_{3j}^k	w_{4j}^k	w_{5j}^k	w_{6j}^k	w_{7j}^k	w_{8j}^k	w_{9j}^k
H1	1	6.89	-5.57	1.43							
	2	-1.13	0.74	-1.35							
	3	2.02	-2.98	-1.36							
	4	1.85	0.33	2.01							
	5	0.98	-1.09	1.57							
	6	0.26	-0.17	-1.56							
	7	-1.89	0.58	2.41							
	8	2.24	2.34	-2.23							
	9	-4.96	-6.16	-0.39							
H2	1	0.50	3.87	-2.42	-0.02	-3.14	-1.76	2.45	0.85	-0.42	0.24
	2	0.93	-1.03	-0.81	0.01	-0.57	-0.62	-0.59	-0.67	0.18	-0.04
	3	-11.47	3.78	1.05	0.78	8.02	-5.28	-6.35	-1.39	0.14	0.05
	4	-4.12	1.35	-14.32	1.30	-2.19	-8.91	-1.29	1.73	0.17	0.01
	5	1.86	-0.66	-0.54	-1.08	0.17	-0.40	-0.70	0.69	0.53	-0.15
	6	-2.53	2.53	4.85	0.12	1.27	1.61	-1.97	0.33	-0.10	0.03
	7	-0.22	2.81	3.42	-0.32	-1.29	2.07	0.77	-0.43	-0.25	0.05
	8	2.01	1.21	1.33	0.07	0.76	1.10	-1.80	1.92	-0.03	0.14
	O	1	1.21	-0.83	0.08	0.18	0.42	0.45	1.45	2.71	-0.07
2		-2.87	0.11	-4.20	0.15	-0.44	0.46	-5.97	-2.58	-1.00	
3		0.12	0.01	-4.43	0.24	-0.57	-1.92	-4.86	-6.93	-3.51	

While flowing from the input to the output, the data gets transformed in different ways that attempt to detect and represent the difference in the impact each input may have on the most relevant output. The relevance of the output is task dependent. It can take multiple repetitions of re-adjusting the different parameters of the network's topology and traversing it forward and backward, before discovering and setting up the optimal estimations (**weights**) of the above-mentioned impact. The process is known as training or **deep learning** (DL), while a reasonably accurate final state of it is called a **model**.

Calculated neuron weights, [image source](#)

Models can be distributed and reused in other Artificial Intelligence (AI) labelled software applications, such as natural language understanding, image and sound recognition and identification, transformation between presentations of information, and decision support agents.

Attention

The attention mechanism in AI is a technique applied in training **ANN**. It enables them to identify the role of the individual elements of the input sequences, as well as their contextual inter-dependencies. The attention mechanism forms the basis of an ANN architecture, known as **transformer**. The approach ensures higher efficiency and better accuracy in solving specific NLU tasks, such as machine translation and text summarisation.

BERT

Google's [BERT](#) (Bidirectional Encoder Representations from Transformers) is a name of a family of machine learning models for natural language understanding, [NLU](#). Similar to the other [LLM](#), it is powered by a deep [ANN](#) architecture, popular with the name [transformer](#). There the significance of each word is weighted to reflect the importance of that word for the sentence by applying an algorithm called [self-attention](#) mechanism.

BERT is one of the first created transformer-based models. *Bidirectional* in its name means that the processing algorithms of the models operate on both left and right sides of any single word in a sentence, while trying to capture that word's semantic meaning, considering the nuances in the words ordering. Unlike the similar [GPT](#), BERT is trained on Wikipedia sources by a method, which focuses on hiding some words in a sentence and then trying to disclose them, by analysing the context of the other words, found on the left and on the right of a hidden one. Therefore, it is known as a [masked language model](#) (MLM).

BERT is a result of [unsupervised learning](#), which means that it doesn't rely on any human interference before or during the training process.

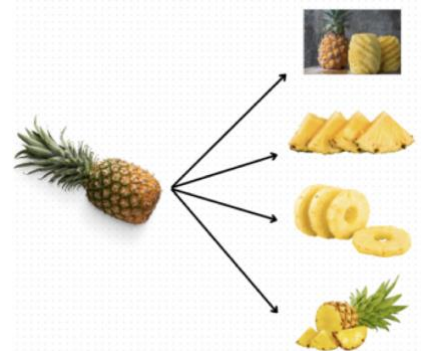
As a result of a combination of their efficient and robust qualities, BERT models prove high performance in extracting context-dependent information and meaning from text.

The Data Science Holodeck project applies [BERT](#), [KeyBERT](#), [Sentence-BERT](#), and [BERTopic](#) for [NLU](#) operations, such as text vectorisation, keywords, key-phrases, and topic extraction, summarisation, and document clustering.

Chunking

Chunking is a process of splitting a large document into smaller parts that can fit to the AI models, which usually can process only input of limited size. There exist multiple strategies of deciding on *the size of chunks*, as well as a variety of software tools for automating the chunking process.

The important criteria to consider, while selecting the size of the chunks is if it enables *preserving the meaning* of the text in the chunks.



Chunking, [image source](#)

Embeddings

Embedding is a process of coding [chunks](#) of documents called [tokens](#), into a sequence of numbers, called [vectors](#). Vector embeddings are numerical representations of natural language, calculated in a way that preserves the semantic meaning of it.

The purpose is to enable the other AI software applications to operate with numbers, instead of operating with text, as the later would have been much more difficult.

The embedded vectors can be of equal size, disregarding the original size of the text they code. Such an approach simplifies operations, like comparison and assessment of similarity between vectors.

There are multiple available embedding tools, each of them applying unique strategy and embedding algorithms. The results of the embedding can be used temporarily or stored permanently for reusing in structures, called [vector databases](#).

In this project, we create both vector and graph embeddings, applying methods provided by [Hugging Face](#) and [Neo4j](#).

GenAI

GenAI (Generative AI) is a category of AI technologies, able to create new original content by simulating human creativity. The generative technologies implement deep [ANN](#) trained to recognize the complex patterns in large amount of input data, and to apply these patterns further for generating new related output: complementing text, imaginative or realistic images, new video, audio, or other composite media types. Typical application of GenAI is found in language translation, documents summarisation, question answering, conversational interfaces, synthetic data generation, and similar NLU implementation tasks

During the recent years the branch of GenAI has advanced dramatically, gaining power from the deep learning algorithms and the innovative [transformer-based](#) data architectures. [ChatGPT](#), [Midjourney](#), [DALL-E](#) are the names of game-changing creative products with top popularity.

We apply GenAI algorithms for document summarisation, clustering, and Q&A (question answering) components of Data Science Holodeck.

GPT

[GPT](#) (Generative Pre-trained Transformers) are left-to-right oriented uni-directional [generative](#) models based on the [transformer](#) architecture. They are trained to predict a next word in a sentence, based on the context gained from the previous words. It makes them fitting well in solving tasks of generating new texts in a response or as a continuation of previously entered text, which is the typical case of a conversation. The most famous representative of GPT model categories is [ChatGPT](#) of the vendor [OpenAI](#). Since publishing of the earlier versions of their products, the company manages to keep delivering higher power and additional value of the AI to a wider population around the world.

The project Data Science Holodeck uses variety of alternative GPT models and tools.

Grounding vs Hallucinations

The implementation of [LLMs](#) does not always produce accurate results. The language - generated tasks sometimes get answers, which are realistic, but not based on true facts. Such results are called [hallucinations](#). The main reason of their existence is the nature of the models they are generated from - models, trained to *always* produce output, even if such output doesn't exist or is unknown. Producing hallucinations could be dangerous, especially in critical domains and situations, such as driverless driving or in medicine. To reduce the risk and improve the accuracy of the outcomes of the generative models, we apply techniques known as [grounding](#).

Grounding is a relatively new term in the AI slang, which refers to a small variety of methods and tools, such as [RAG](#), engaged in retrieving and supplementing [LLMs](#) with *factual data*, collected from the application domain. Grounding is used in requesting responses, specifically generated from factual data, or as a component in original requests, where it is expected to draw the 'attention' of the responding pre-trained generative models

towards important features of the input. The grounding techniques guide the LLMs to generating responses that are relevant and reliable not just in general, but inside the specific enterprise domain.

Data Science Holodeck implements grounding by integrating of [knowledge graphs](#) and [vector databases](#), which contain domain specific factual data in task-specific context, with pre-trained and publicly available [GenAI](#) models.

Hugging Face



[HuggingFace](#) is a company, family of products, and a platform for publishing software libraries and other usable programming tools for developing and deploying AI applications. It hosts a huge and fast-growing amount of machine learning (ML) models, datasets, APIs, coding examples and others supporting tools, particularly such designed for processing natural language documents and multimodal data. HuggingFace supports various tasks, including text classification, translation, named entity recognition, and question-answering. It also offers integrations with popular machine learning frameworks and hosts public and private ‘spaces’ for applications created by the platform’s users.

At present, the HuggingFace Hub contains over 230K models, 40K datasets, and 30K demos, submitted by developers from all over the world. Practically, it works as a free central repository of programming resources that everyone can share, discover, explore, and experiment with. See more at https://github.com/huggingface/huggingface_hub.

Other friendly and popular AI hosting platforms are [LangChain](#) and [LlamaIndex](#).

The project Data Science Holodeck uses [HuggingFace](#) and [LangChain](#) as providers of LLM, [agents](#), and processing tools.

Inference

Inference is the process of using a trained model to make predictions on new data. The pre-trained models are stored on a server and shared with the client applications through API.

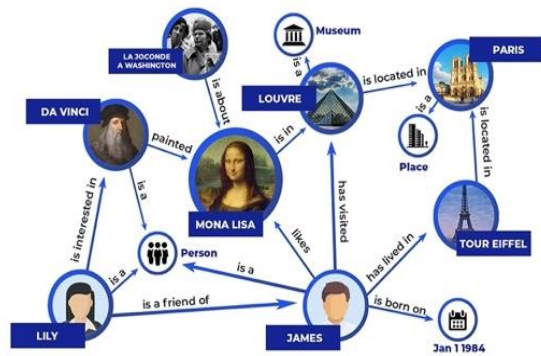
The service is especially valuable for heavy tasks, which require powerful computers, which the clients may not have available. Platforms, like [HuggingFace](#) library provide templates for client applications that would call and run inference processes, associated with the already hosted models.

Knowledge Graph

A graph is probably the simplest mathematical term and a data structure that can be used to explain real world objects and their relations.

Knowledge graph (KG) is such a computer readable structure that organizes related data into a graphical format. Related entities like domain objects and events are stored as **nodes** of a graph, while their logical relations are presented by graph **edges**. Both the nodes and the edges may have **properties**, whose values are stored in a graph database.

Except of being highly informative, intuitive, fast and flexible for extending and maintaining, the knowledge graph benefits from multiple known from mathematics **graph algorithms**, such as shortest path, centrality, betweenness, etc. Such an algorithm can, for example, perform analysis of the graph shown here, and answer questions like: *Which is the most connected node? Which is the shortest connection of Lily with Louvre? How many and which cities are included in the graph?*



Knowledge graph, [image source](#)

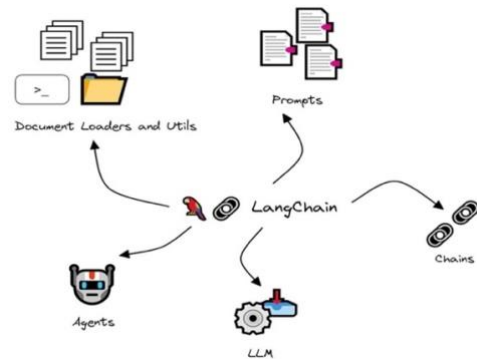
The project Data Science Holodeck uses KG as a global domain data structure, storing documents from variety of sources and formats. We aggregate this data into a pre-trained LLM to enable semantic search and Q & A operations within that domain.

LangChain



[LangChain](#) is a library of software components supporting the developers in creating NLU applications for solving various language related tasks, such as document processing, transformation, and generation with LLM, through composition of modules.

In general, it works as a hub, accommodating a broad availability of modules, which have been originally generated and hosted on other platforms. It also contains predefined operation templates and patterns broadly implemented in typical NLU solutions.



LangChain components [image source](#)

The core components of LangChain are organised in the following categories:

- **connectors** to documents found in various media in different formats
- instruments for **embedding, integration and storage** of the collected data in both traditional databases and innovative vector data stores
- **interfaces** to various pre-trained LLMs from different vendors and tasks relevance, including multiple open-source models and products of major players like OpenAI, Google, and Meta, as well as **support** for their implementation in user applications
- pre-designed pipelines of NLU operations - **chains of calls** to relevant components, used for faster building of a complex solutions
- **software agents** – autonomous components, which can plan operations responding to the user input, select relevant task-solving tools, and act in a step-by-step way
- **callback functions** that can run at specified points at LLM running – can be used, for example, for logging, monitoring, streaming and other support of debugging and testing of applications

For more details on the rich LangChain content and how to use it in applications see their documents, on https://python.langchain.com/docs/get_started/introduction.html

The project Data Science Holodeck uses several models, pipelines, and tools, provided by the LangChain library.

LlamaIndex



[LlamaIndex](#) is another valuable source of available instruments for building AI applications, specialised in connecting of a broad variety of custom data sources to multiple [LLMs](#) (where the name comes from). It also provides algorithmic support for augmenting and integrating private and public-domain data, structuring, indexing, searching, and storing data and applications. Their resources can also be used in building knowledge bases and agent-controlled automated systems.

LLM

LLMs (Large Language Models) are machine-trained models of AI solutions of tasks, related to natural language processing and understanding (NLP/NLU). They are built by deep iterative training, testing, and validating of [ANN](#) processing sources of natural language. As trained generic products, possessing sufficient accuracy, the models and their parameters get stored and made available for common use in AI applications. Such models can be fine-tuned additionally for addressing a specific domain of data or for achieving higher precision.

Different LLMs are specialised for working on different AI language tasks, such as [embedding](#), summarisation, transformation, generation, and others. The difference comes from the different algorithms and datasets the models have been trained with. By definition, each model is best used in natural language tasks, like those it has been trained to solve. The popular AI application [ChatGPT](#) implements such models for generating conversations.

This project Data Science Holodeck uses multiple LLMs from [Spacy](#), [Hugging Face](#) and [Lang Chain](#) families, such as Google's [BERT](#), Facebook's [BART](#), MistralAI's [Mistral](#), and, TheBloke's [Zephyr](#), to mention some.

NLP/NLU

NLP and NLU (Natural Language Processing and Understanding) form a large branch of AI, which consists of multiple machine learning methods, models, and tools for intelligent analysis, operations, and transformation of written or spoken in natural human languages text. These are instruments of higher complexity, which can be used for automation of complex language specific processes, such as extracting and reproducing language semantics in variety of grammatical and lexical implementations.

The modern NLP and NLU instruments vary significantly, in dependence with the tasks they are designed to solve. Some of the most popular AI language specific [tasks](#) and [sub-tasks](#) are:

Information Extraction - processing the text for identification and classification of key items, which can be used for generating a meaningful [summarisation](#) of it

The process considers parameters, such as frequency of appearance of an entity (a word or expression) in the one or more text documents, as well as its syntactical use and semantical meaning (see also [POS](#) and [NER](#)).

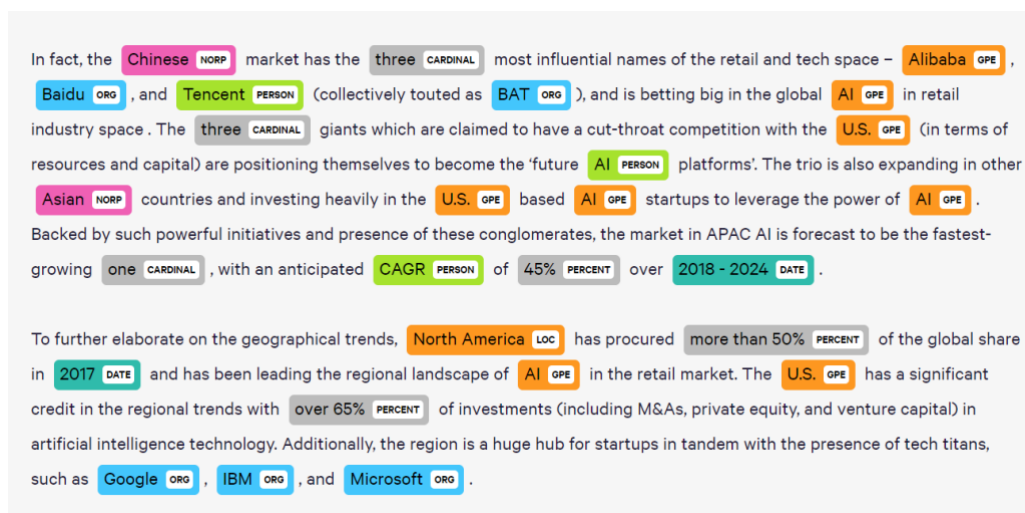
The representation summary of the text can be generated as either as an [extraction](#) of set of the most meaningful and identifying entities found in the original document, or as an [abstraction](#) - a generation of new text, similar to the original but shorter and narrower in context.

Machine Translation – one of the oldest NLP solutions, related to processing the text for understanding the semantic content and reproducing the same content by *generation* of new text in another natural human language. Applied methods include solving tasks from the information extraction class, as well.

Sentiment Analysis – processing the text for identification, *classification*, and labelling the meaning of it according to polarity (positive, negative), emotional (angry, happy), or aspect based (fast, slow) subjective criteria. The sentiment analysis techniques are applied in businesses monitoring of customers feedback and satisfaction, for evaluation of brand or product acceptance and better understanding customer needs. It plays essential role in automated recommender systems and team building, as well.

NER

NER (Named Entity Recognition) is a core component of the pre-processing stage in many [NLP/NLU](#) applications, aiming at searching, summarization, language translation, or question answering. Its particular task is to seek, allocate, classify, and annotate the **named objects**, mentioned in the text, such as names of persons, cities, universities, companies, currencies, months.



NER tagging in [spaCy](#), [image source](#)

For higher precision of the results, it is possible to train NER for recognizing not only generic named entities, but also user-defined set of categories. The results of NER are passed to other modules for higher level of natural language analysis.

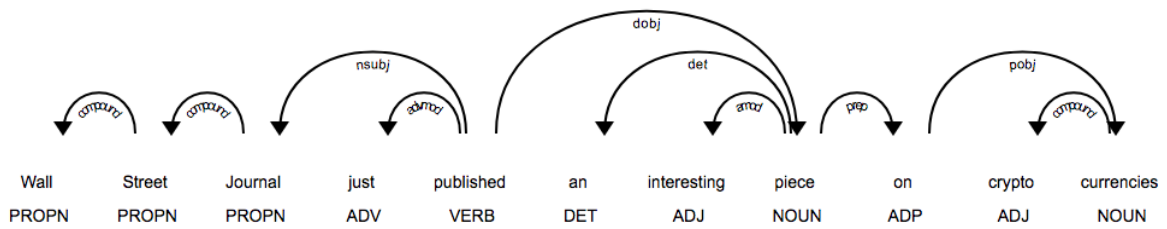
NER implementations can be found in multiple AI development platforms, most of which are language dependent. All available resources offer support for English language, but few produce good results for Danish and the other low-resource languages.

Data Science Holodeck uses the NER tools offered by Spacy for training domain specific entity models.

POS

POS or PoS (Part Of Speech) is a grammatical category, describing the syntactical role of the individual words within the structure of a sentence. The definition of such roles can differ from language to language. In English, parts of speech are noun, verb, adjective, adverb, pronoun, preposition, conjunction, interjection, numeral, article, and determiner (Wikipedia).

In NLP, each word's role is identified and labelled with a relevant tag, used in rebuilding the structure of the sentence and extracting semantic information from it.



POS tagging in [spaCy](#), [image source](#)

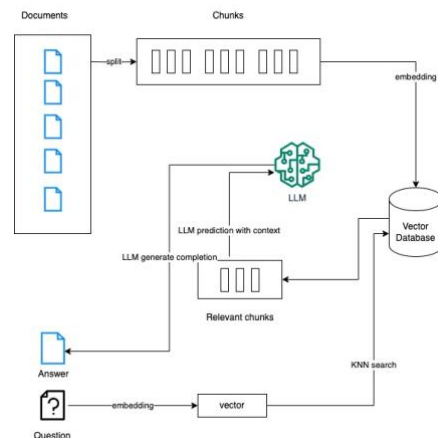
By means of POS we can create a sentence dependency tree, where the verb is a root and a link between the subject and the object.

RAG

RAG (Retrieval-Augmented Generation) is an approach of combining the classic *information retrieval* techniques with modern *generative technologies*. It is introduced by Meta as a solution for improving the accuracy of **LLM** applications by enabling them with access to new and dynamically extendable information, that exists in external, for them, sources.

In RAG, the developer can collect a set of documents, relevant to the domain, to pre-process them independently from the LLM processing, and at the next step, to integrate the RAG pre-processing outcome with the input to the LLM application.

The process requires applying same models of [chunking](#), [vectorisation](#), and [embedding](#) of the content of the external documents, as the models used for vectorisation of the prompts - the human questions in question-answering, chat, and other text generation systems.



RAG Architecture, [image source](#)

The advantages of involving RAG in the process of NLU are the enabling of personalisation and better adaptation to the tasks, keeping consistency of the LLM by providing it with the new available facts, and therefore ensuring higher reliability of the operations results.

Data Science Holodeck applies RAG by aggregating data in **KG** and pre-processing it with graph-based algorithms, before integrating it with LLM for generative language-specific processing.

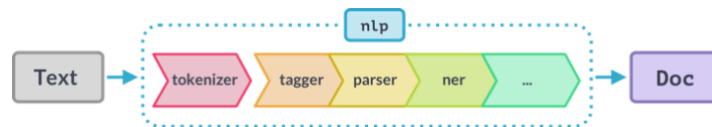
Software Agent

In AI, an agent is an autonomous software component, which can perceive the environment, make choices and take actions to orchestrate the tasks of the program. Nowadays, the agents get popularity in connection with the development and implementation of various AI tools and hosting platforms, such as [LangChain](#) and [LlamaIndex](#). In the context of LangChain, the software agents play role of dynamic chains of prescribed

alternatives of behavior, on which they can decide in reaction of the input. Once an agent chooses a model to follow, it also chooses the appropriate tools to use.

SpaCy

spaCy is one of the first popular powerful open-source software libraries for advanced natural language processing. It is designed for support of development of large-scale enterprise applications and use in production. It provides tools and pipelines for solving wide range of major NLP tasks, as well as multiple extensions for data and models visualisation.




Operations pipeline in spaCy, [image source](#)

The library is dynamically maintained and permanently upgraded to enable convenient access and tutoring help for implementation of the recently developed [LLMs](#) and other AI tools. It also has integral connection with other advanced platforms, such as [HuggingFace](#) and [LangChain](#). For more details and downloads visit [spaCy's universe of resources](#).

The project Data Science Holodeck uses spaCy for pre-processing of natural language data.

Streamlit

 **Streamlit** is a framework of tools for building interactive web applications. It is designed to facilitate the development of data science projects in Python programming language.

A Streamlit application can be deployed on a cloud server or reside on a local machine. Besides the typical instruments for serving a web application, the framework offers a library of predefined graphical widgets for visualisation, including variety of dialogue frames data diagrams. The applications, running Streamlit display interactive web pages, which are compatible with the popular web browsers.

The project Data Science Holodeck uses Streamlit as a front-end development platform and Streamlit community cloud as a deployment platform.

Supervised, Unsupervised, and Self-Supervised Learning

In **supervised learning**, the AI software is provided with historical data or existing documents, that have already been labelled – meaning, given additional context, which can help to the training process. During the training process, where the model learns patterns in the input data, the labels play a role of a supervisor telling what is right or wrong.

On the contrary, in unsupervised learning, the AI software has similarly huge amount of input data, available for the training process, but it is not labelled. The applied algorithms use feedback loops to find out if and how they have been wrong or right about the patterns they discover. The loops are repeated until the feedback is acceptably positive.

Self-supervised learning is a subset of unsupervised learning, in which the output results are part of the input data – meaning the models are trained by use of additional support mechanisms, just like people learn to cook by using recipes or to program by using tutorials.

Self-supervised methods of learning have become extremely popular nowadays, as they enable pre-training a model on unlabelled data, like the unsupervised methods, and consequent fine-tuning on smaller labelled data sets, tailored to the domain and the task, as supervised methods do. The result is higher performance and better accuracy.

Transformer

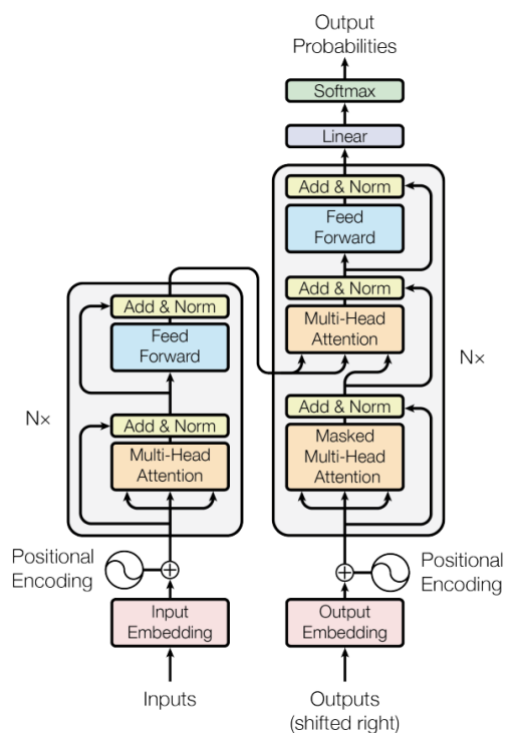
Transformer is a name for type of architecture of [neural networks](#), used for solving tasks, related to machine understanding of the meaning of human text. A typical example of such task is the machine translation from one natural language into another.

Transformer consists of two separate software components — **encoder**, which reads the input text and codes it into a sequence of digital vectors ([embeddings](#)) and **decoder**, which produces a new sequence of text, *word by word*, while trying to predict the expected output.

The encoder has the task to generate the most informative embeddings of the words that fit best to the semantic meaning of the whole sentence. It does this in several iterations.

First, it splits the text into small chunks – tokens and encodes each of them by a set of numbers (embeds them). Then it tries to improve the embeddings, transforming them in a way that gives higher weight to key tokens, while diminishing the less important tokens.

After generating the initial embedding of one word, the encoder pays attention to every other word in the same sentence and gives each of them a score (attention score).



Transformer's encoder and decoder, [Source](#)

The words that fit better into a connection with the base word and contribute more to disambiguate its semantic meaning get higher attention score than the unrelated words.

The algorithm then re-calculates the initial embedding of the word, aggregating the attention scores of the other words, interpreted as weights. This way, the word gets 'informed' about the context, created by its neighbours.

The operation is repeated multiple times in parallel for all words. As a result, the whole sentence receives better digital representation of its semantic meaning and the inter-related role of the words in it.

The applied approach is called [self-attention mechanism](#).

The decoder applies similar mechanism, iteratively generating word by word, from left to right, while paying attention to both the encoder's outcome and the already predicted words on the left.



The encoder self-attention distribution for the word "it" from the 5th to the 6th layer of a Transformer trained on English to French translation (one of eight attention heads).

[Source](#)

Transformers have been introduced by Google researchers in 2017, creating revolution in [NLP/NLU](#) branch of AI. Similar architectures and implementations have been reported as early as in [1992 and 2014](#).

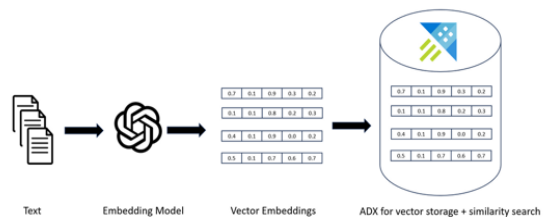
See the original article here: <https://arxiv.org/pdf/1706.03762.pdf>

Vector Databases

The vector databases are stores, designed for optimal storage and efficient search of [embeddings](#) of the text documents. The documents are encoded into set of numbers (in computer programming called **vectors**) by implementation of intelligent algorithms.

The access to the vectors is facilitated by advanced indexing system, which enables quick search and finding of specific content or similar elements.

Vector stores volumes can be scaled up or down, as necessary. Applications applying vector stores are particularly more efficient in working with large datasets than the other types of databases.



Vector Database, [image source](#)

The project Data Science Holodeck uses [ChromaDB](#), [Faiss](#), and [Weaviate](#), as well as [Neo4j's Vector Index](#) for persisting and searching vector embeddings.

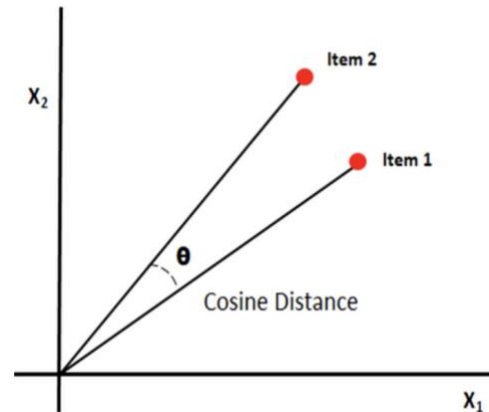
Vector Similarity

This is a mathematical measure of similarity of the content of two documents, based on the estimation of the difference between the vectors, representing these documents in the multidimensional vector store. This way, vector similarity provides a simple and effective method for searching in a large-scale dataset and identifying the related content, stored in it.

One of the popular measurement scales is based on the mathematical knowledge of measuring the similarity of two geometrical vectors by estimating the size of the angle that separates them.

The primary assumption is that just like a geometrical vector, one text document or one other type of data set can be represented by a sequence of numbers that determine the position of that object in the space.

As the space is shared by multiple objects of the same kind, it is not difficult to estimate how close or distant the space objects are.



Cosine similarity: two objects are similar if the angle between them is small, [image source](#)

In geometry, two vectors are count similar, if the angle between them is insignificantly small. Therefore, in AI programming, the measurement of documents' and data sets' similarity applies same methods and terminology of **cosine similarity**.