Generative AI

About me

- Graduated Koźmiński University in Finance&Accounting
- Self-taught Data Scientist with 7 yrs of experience
- Currently working as Senior Data Scientist at Shelf
- Focus on NLP, unstructured data, similarity exploration, data enrichments and pricing
- Enthusiast of building AI Agents and orchestrating complex LLM pipelines combined with classic ML models

Course Goals

What to expect?

- Exploring GenAI use cases and challenges with focus on LLMs
- Introduction to basic LLM and NLP theory
- Introduction to Large Language Models
- Learning how to build first models with popular frameworks such as spacy
- Improving Python skills
- Group assigment to explore one of LLM use cases at home
- Building foundations and a roadmap for further learning

What not to expect?

- Extensive NLP and Machine Learning Theory lectures
- Learning to code in Python from scratch
- Becoming NLP expert in 20 hrs
- Learning from slides alone

Course Agenda

- **W1 – Introduction to GenAI and NLP Theory fundamentals**
- **W2 – LLM toolkit**
- **W3 – AI Agents and LLM flow orchestration**
- **W4 – Course assignment & LLM productization**

Course Agenda

GenAI & NLP

Gen AI Intro
- What is GenAI?
- Why LLMs?

NLP Theory
- Transformers & Attention
- Tokenization and embeddings
- ***Showcasing meaning in vectors***
- Semantic similarity and its pitfalls
- ***BERT finetuning experiment***
- LLM training and fine-tuning

LLM Toolkit

LLM Basics
- Prompt Engineering
- How does next word search work?
- General LLM usage tips

LLM Context enrichment
- Retrieval Augmented Generation („RAG")
- Vector Databases
- ***Hybrid search with ChromaDB***

Classic NLP & LLMs synergy
- Named Entity Recognition
- CrossEncoders as Rerankers and Evaluators
- How to find best models?
- ***Fine tuning Xencoder for evaluation***

AI Agents

Agent Basics
- 6 levels of LLM Autonomy
- Tool usage
- \`Reasoning\` proces

LLM Flow orchestration
- How to improve results with multiple LLM-calls
- ***Building first flow with LangChain and eval with LangSmith***

ChatBots
- Why are chatbots most popular LLM use case?
- How to enable chatbots take actions and access data?
- ***Building level 5 autonomy Chat Bot***

Assignment & Productization

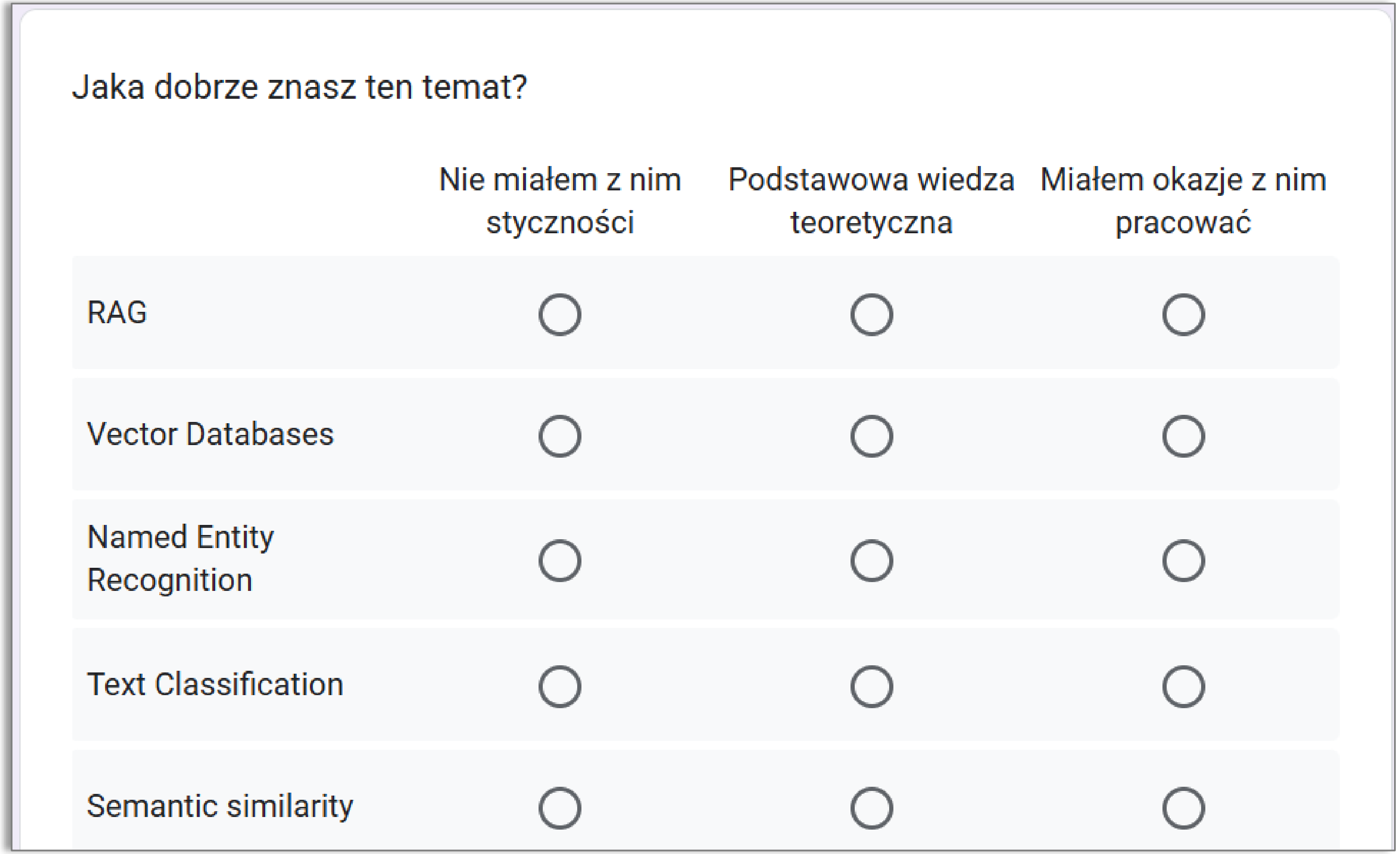
Assignments Presentation
- Discussing Results
- Lessons learned & Challenges

LLMs in production
- Evaluating user feedback
- *Optimizing cost and latency*

On-Premise LLMs
- When is it worth launching LLMs locally?
- ***Launching <10B LLM locally with LM Studio***

Intro questionnaire

[Questionnaire Link]



Jaka dobrze znasz ten temat?

	Nie miałem z nim styczności	Podstawowa wiedza teoretyczna	Miałem okazje z nim pracować
RAG	○	○	○
Vector Databases	○	○	○
Named Entity Recognition	○	○	○
Text Classification	○	○	○
Semantic similarity	○	○	○

W1 Agenda

- Introduction to GenAI
- Why do LLMs and NLP dominate AI?
- NLP Theory basics:
  - Tokenization and Embeddings
  - Semantic similarity
  - Transformers & Attention
  - Training and Fine-tuning LLMs

Introduction
Generative AI

What is Gen AI?

Definition:
AI that generates new content—text, images, music—by learning patterns in large datasets.

Most popular Generative Models:
Includes techniques like GANs (Generative Adversarial Networks), VAEs (Variational Autoencoders), and Transformers.

How It Works: Learns the probability distribution of training data to create novel outputs.

Key Use Cases: From writing assistance (ChatGPT) to image creation (DALL·E) and product design.

Ethical & Practical Considerations: Highlights issues around bias, copyright, and authenticity in AI-generated content.

What differentiates GenAI from classic ML?

	Classic ML	Gen AI
Task Orientation	Primarily focused on classification or regression (predicting labels/values).	Generates new data (text, images, audio) by modeling underlying data distributions
Model Architecture	Often uses discriminative models (e.g., logistic regression, CNNs for classification).	Employs generative models (e.g., GANs, VAEs, Transformers) to create novel outputs
Learning Approach	Learns decision boundaries or function approximations.	Learns the probability distribution of data, enabling synthesis of new, realistic samples.
Data Requirements	Needs labeled data for supervised tasks, or unlabeled data for clustering.	Still benefits from large, diverse datasets—often leveraging self-supervised or unsupervised pretraining.
Applications & Challenges	Price prediction, fraud detection, image classification, recommendation systems.	Creative content generation, language translation, design prototyping—raises ethical questions around bias, copyright, and authenticity.

Generative AI seems to have human like capabilities

Chat GPT

- Generate human-like conversations and answer
- Answer complex queries on any topic
- Great at writing and refactoring code

Stable Diffusion

stability.ai

- Text-to-image generation
- Accelerates creative process
- Still some way to go before replacing actual graphic designers

Prime Voice AI

Eleven Labs

- Mimic any voice and maintain emotions
- Potential to replace lectors and dubbing

GenAI in audio creates or transforms audio content, such as music, speech, and sound effects, rather than simply reproducing or manipulating existing clips.

1. **Text-to-Speech (TTS) and Voice Cloning**
   1. **Personal Assistants** (e.g., Alexa, Siri): Generate natural-sounding responses.
   2. **Voice Over** for Videos: Rapidly produce narration in multiple languages.

2. **Music Generation and Composition**
   1. **Background Music** for Games & Media: Automatically create mood-specific soundtracks.
   2. **Personalized Playlists**: AI-composed music tailored to individual taste.

3. **Voice-Driven Applications**
   1. **Real-time Translation**: Automatically generate speech in another language with a matching vocal profile.
   2. **Conversational AI**: Enhance chatbots with lifelike voice and emotional intonation.

4. **Sound Effect (SFX) Generation**
   1. **Video Games & Films**: Create realistic or stylized soundscapes without the need for large libraries.
   2. **Virtual Reality**: Dynamically generate immersive, context-aware audio.

GenAI in image generation uses machine learning models (e.g., GANs, diffusion models) to produce new images from scratch or modify existing images in novel ways.

1. **Text-to-Image Generation**
    1. **Marketing & Advertising**: Rapid generation of branded visuals without a photo shoot.
    2. **Creative Tools**: Artists use AI to visualize concepts or mood boards instantly.

2. **Image Editing & Inpainting**
    1. **Photo Restoration**: Filling in missing or corrupted parts of images.
    2. **Retouching**: Removing unwanted objects or enhancing image details.

3. **Face Synthesis & Manipulation**
    1. **Virtual Avatars**: Generating lifelike human faces for games, social media, or training data.
    2. **Deepfake Content**: Swapping faces or altering facial expressions in images.

4. **Data Augmentation**
    1. **Training Datasets**: Generating new samples to improve model robustness (e.g., for object detection).

GenAI in text generation trains models to produce written content, such as articles, chat responses, summaries, or code, based on input prompts or context

1. **Content Creation & Copywriting**
   1. **Marketing & Advertising**: Quickly generate product descriptions, headlines, and taglines.
   2. **Journalism & Blogging**: Draft articles or summaries, reducing research and writing time

2. **Conversational Agents & Chatbots**
   1. **Customer Support**: Provide real-time, context-aware responses to frequently asked questions.
   2. **Personal Assistants**: Manage schedules, compose emails, and interact naturally with users

3. **Code Generation & Programming Assistance**
   1. **Automated Code Suggestions**: Speed up coding tasks and reduce errors.
   2. **Documentation**: Generate inline comments and user guides for software projects.

4. **Text Summarization & Translation**
   1. **Document Summaries**: Extract key insights from long reports or articles.
   2. **Multilingual Support**: Provide translations for global audiences or cross-border collaboration.

GenAI also present a wide range of issues

Intellectual property

An image generation model processes thousands of images – is it a digital age equivalent of an artist visiting The Louvre or is it actually profiting from artists' talent for free?

Limitless fake news

Autogenerating content is nearly free, how can we control its quality? If fake news is a problem when someone has to spend time making it up, how can we handle endless, autogenerated fake news?

Cheating and plagiarism

Can students find information and learn faster or will they just auto-generate their homework?

Impersonation

Generative speech and video makes it easier to create fake videos. How will we validate if we are speaking to the actual person if his voice can be copied with a few seconds recording?

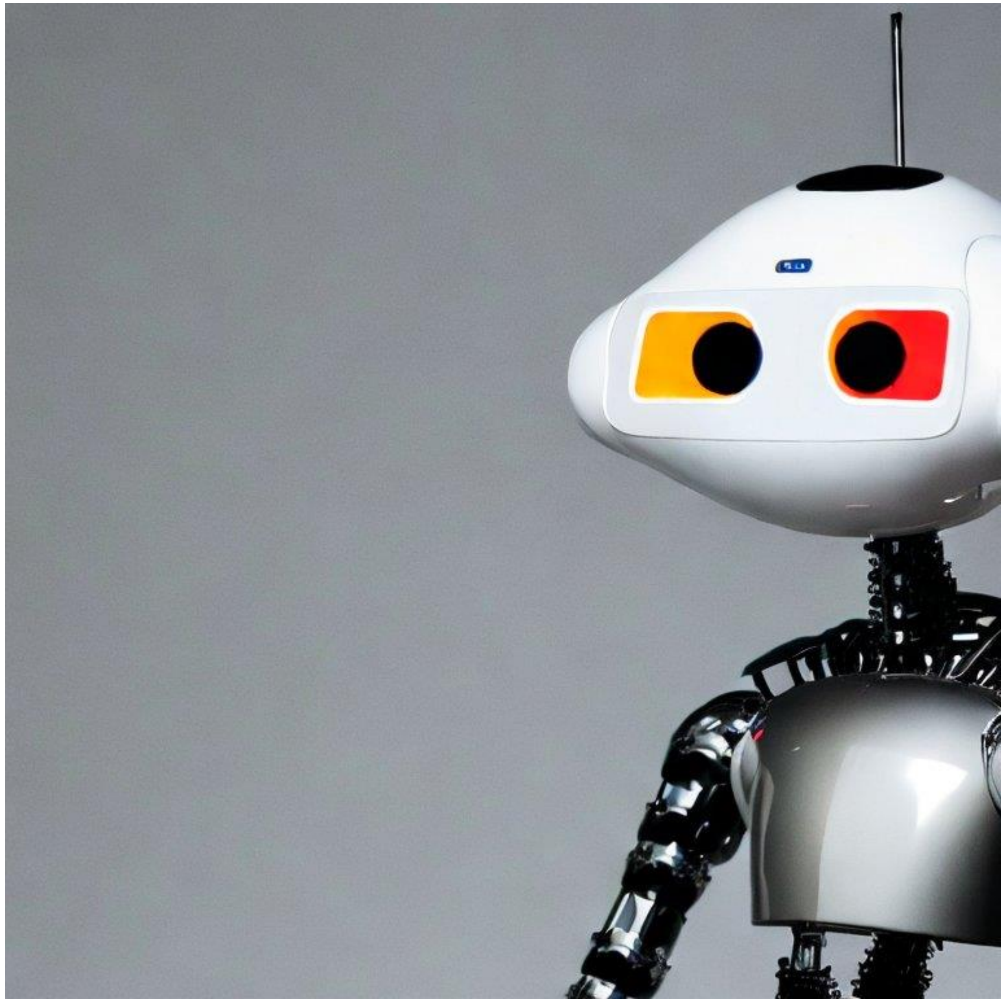
Handling these issue in the future can become a whole new industry

Artificial Intelligence will not replace programmers and engineers…

… but programmers leveraging AI to make their work more efficient will replace ones who don’t

Why do LLMs dominate GenAI revolution?

Is there something odd about this picture?



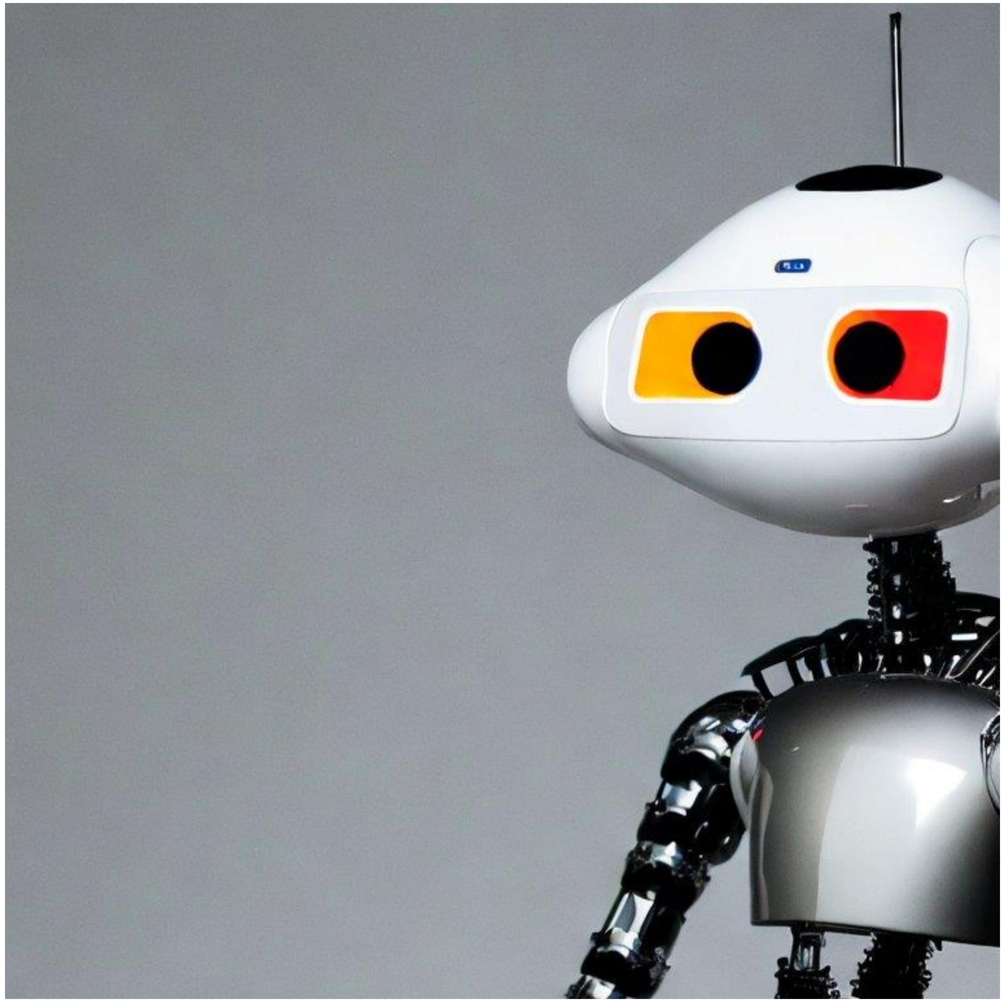
Source: Stable Diffusion

Can you image Artificial Intelligence without a conversation?

- As speech is our primary form of communication we require this ability from an Artificial Intelligence

- For decades you could communicate with a computer only by knowing how to code

- Recent breakthroughs in NLP help computers grasp the human language, which increases their accessibility



Source: Stable Diffusion

Advantages of language as a medium for intelligence

Language is central to human communication and knowledge sharing.

Once Computers learned to minic conversations they are perceived as inteligent

Language is a bridge allowing to join all possible abstract concepts.

This allows LLMs to leverage language as a medium to memorize and estimate most probable outcomes

Since the invention of the internet text data has been growing expotentially.

This allowed to build vast datasets… but may be more tricky in the future

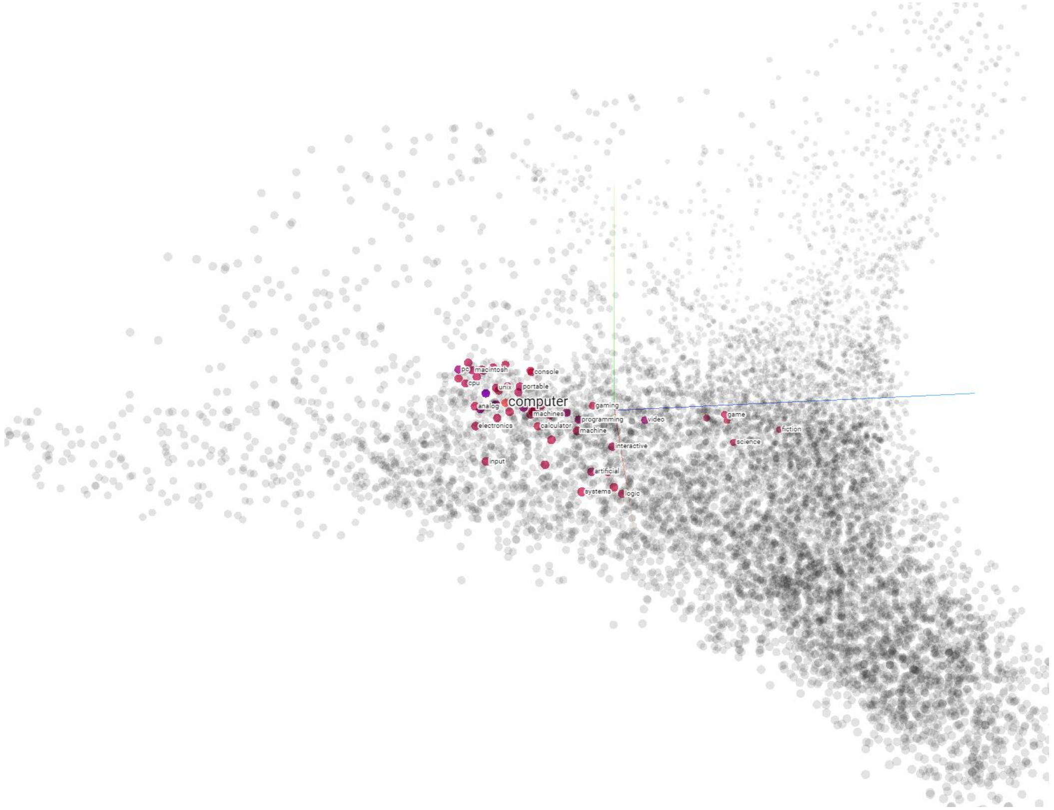
The flexibility of language itself, allows to train versatile models, which can perform complex task without specific training.
This lifted most of barriers for classical ML models.

Natural Language Processing theory

At the end of the day a computer can only understand vectors

Word2Vec vectors

- Machines rely on numbers and are not able to understand characters and words
- Converting words and sentences to vectors is the foundation of NLP
- Even current State-of-the-art solutions, which often reach near human performance are still based on vectors
- Enormous increase in computing power over the last few years fuelled rapid development of NLP in last decade



How do machines learn to convert language to vectors?

- Computers learn to „understand" language by imitating humans – similar to a child learning new words
- Predicting the most probable word in a sequence is how transformer-based NLP models are trained to convert language to vectors
- This ability to learn requires complex architecture – GPT3 has about 175B parameters, and GTP4 will be 500X larger

GPT3 model architecture

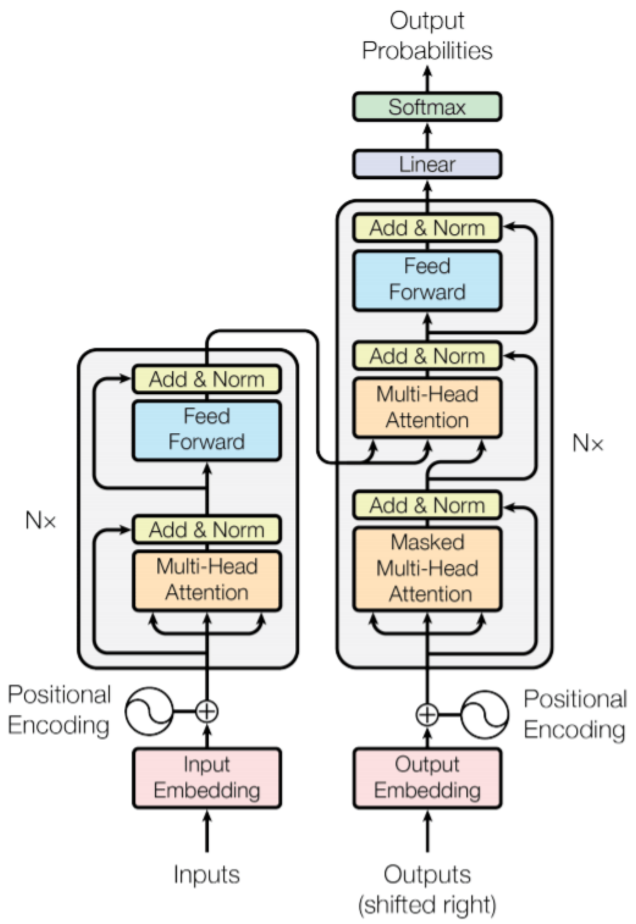


Figure 1: The Transformer - model architecture.

Tokenization

Tokenization – BERT example

- SoTA Tokenizers are more complex in handling words
- They will have some most common words in their vocabulary
- But for less frequent ones are represented by multiple tokens
- Some words can not be present in tokenizer vocab, they would et an OOV token
- in BERT tokenizer case they would probably still be tokenized as chars

BERT Tokenizer example

- Spell has its own token – 6297
- Mispelled will be tokenized as [3335, 11880, 3709], which corresponds to ['miss', '##pel', '##led']

Word vectors and similarity

How to vectorize words

- Translating words into vectors might sound abstrack at first but with a text corpus large enought we can vectorize words based on the context they appear in
- We can approach word vectors by word-by-word approach if we want to classify their meaning
- Or any other context e.g. word-by-doc if we want to Focus on task-specific vectors, which will allow us to use text vectorization to classify text by its domain

Data Science is **popular** these days

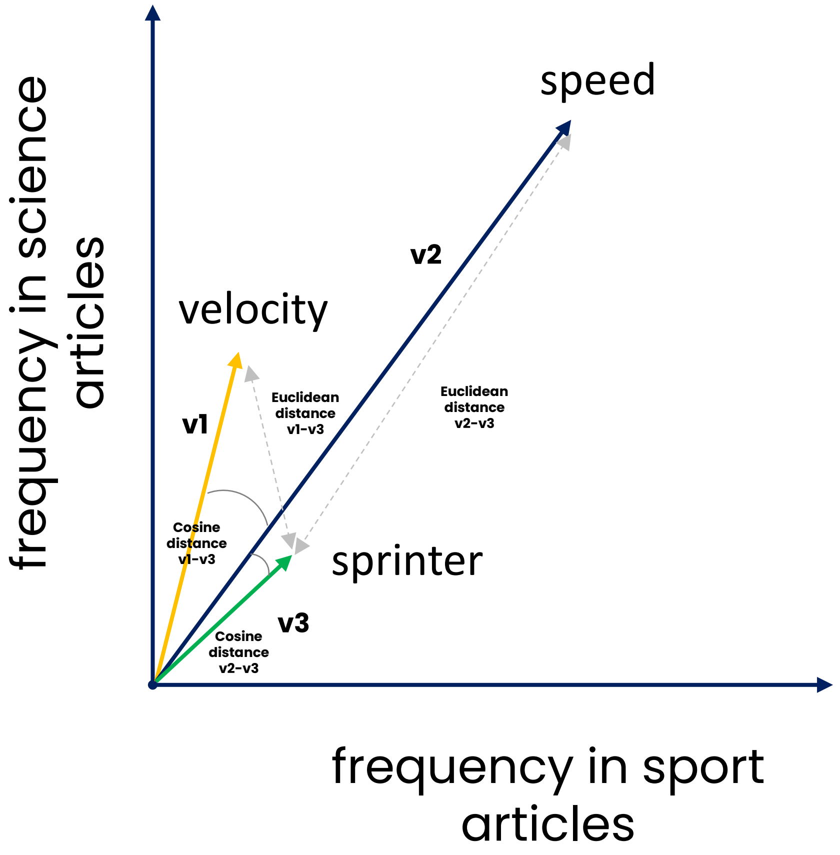
Quality of **data** is **popular** issue among companies

Co-occurrence matrix allows gives us a meaningful word vector based on words

	popular	science	quality
data	2	1	1

How to measure similarity

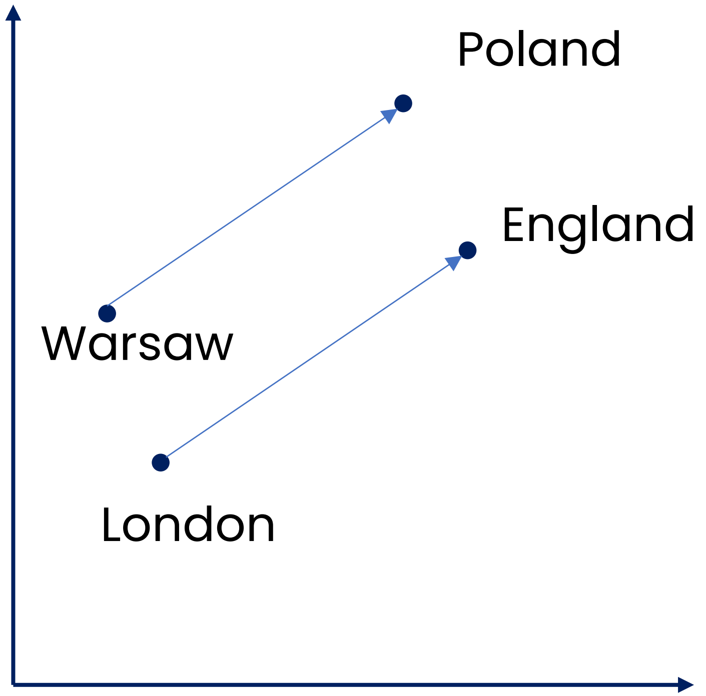
- Measuring vector similarity is challenging, especially in high dimensionality
- While Euclidean distance is easily applicable for 2D data it performs badly with higher dimensions
- Cosine distance is most popular in measuring similarity
- It is based on vectors direction and ranges from -1 (opposing vectors) to 1 (proportional vectors), while orthogonal vectors have a similarity of 0.



Word vectors allow us to actually extract relations between words

- If we take two word pairs – (Poland, Warsaw) and (England, London) both these pairs will create a similar vector
- This is the product of similar context they are present in
- If we have a large corpus of news, and articles there will be multiple coocurences of phrases like „{Warsaw/London} is the capital city of {Poland, England}"

Word vector space



Word vectors are not just some abstract conversion of words, to numbers. As they are based on words context and coocurence vectors keep some of this information

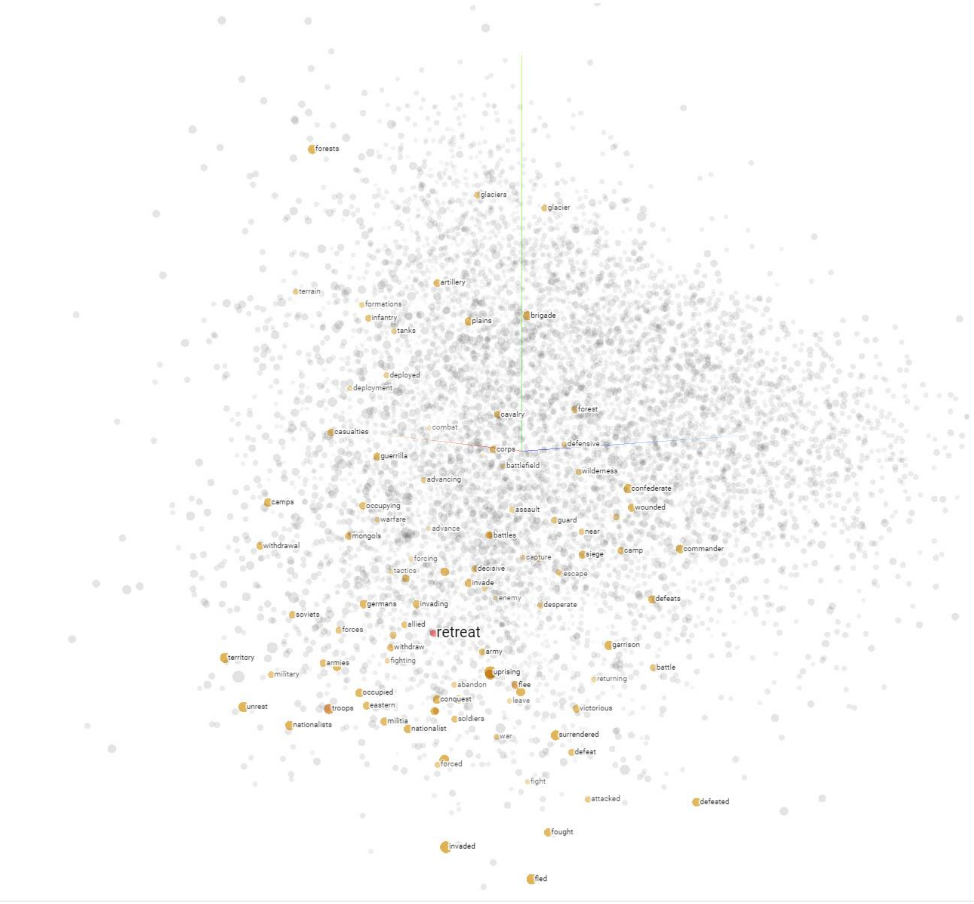
How did embedding methods start?

- **word2vec** (Google, 2013)
- ***Continuous bag-of-words (CBOW):*** the model learns to predict the center word given some context words.
- ***Continuous skip-gram / Skip-gram with negative sampling (SGNS)***: the model learns to predict the words surrounding a given input word.
- ***Global Vectors (GloVe) (Stanford, 2014)***: factorizes the logarithm of the corpus's word co-occurrence matrix, similar to the count matrix you've used before.
- ***fastText (Facebook, 2016):*** based on the skip-gram model and takes into account the structure of words by representing words as an n-gram of characters. It supports out-of-vocabulary (OOV) words.

Classical (non Deep learning) word embeddings were an important step in NLP development and are great for understanding the basics, but they are not used that much in real life use cases anymore

Visualizing vectors – dimensionality reduction

- Multi-dimensional vectors are hard to visualize as we can see only 3D
- With word vectors they tend to have hundreds of dimensions
- Depending on the task, majority of the information might be hidden within ~5% of dimensions only
- We use Dimensionality Reduction algorithms to visualize similarities in high-dimensional space without loosing too much information
- Most commonly used algorithms include T-SNE or PCA
- Explained variance represents loss of information due to dimensionality reduction
- Experiment yourself on https://projector.tensorflow.org/



Leveraging Neural Neutworks in embeddings

- There are multiple ways of word embeddings, but leveraging Deep Learning for this task is becoming more and more common
- This is an example of a self-supervised task. It is unsupervised itself, as we have no labeled data for the correct embeddings. However the corpus we use for training provides necessary contex, which has some similarities to a supervised learning problem
- Embeddings are often created as by-product of a supervised NLP task, this allows us to guide our embeddings to match our specific objective

Specialized & Sentence-Level Embedding Models

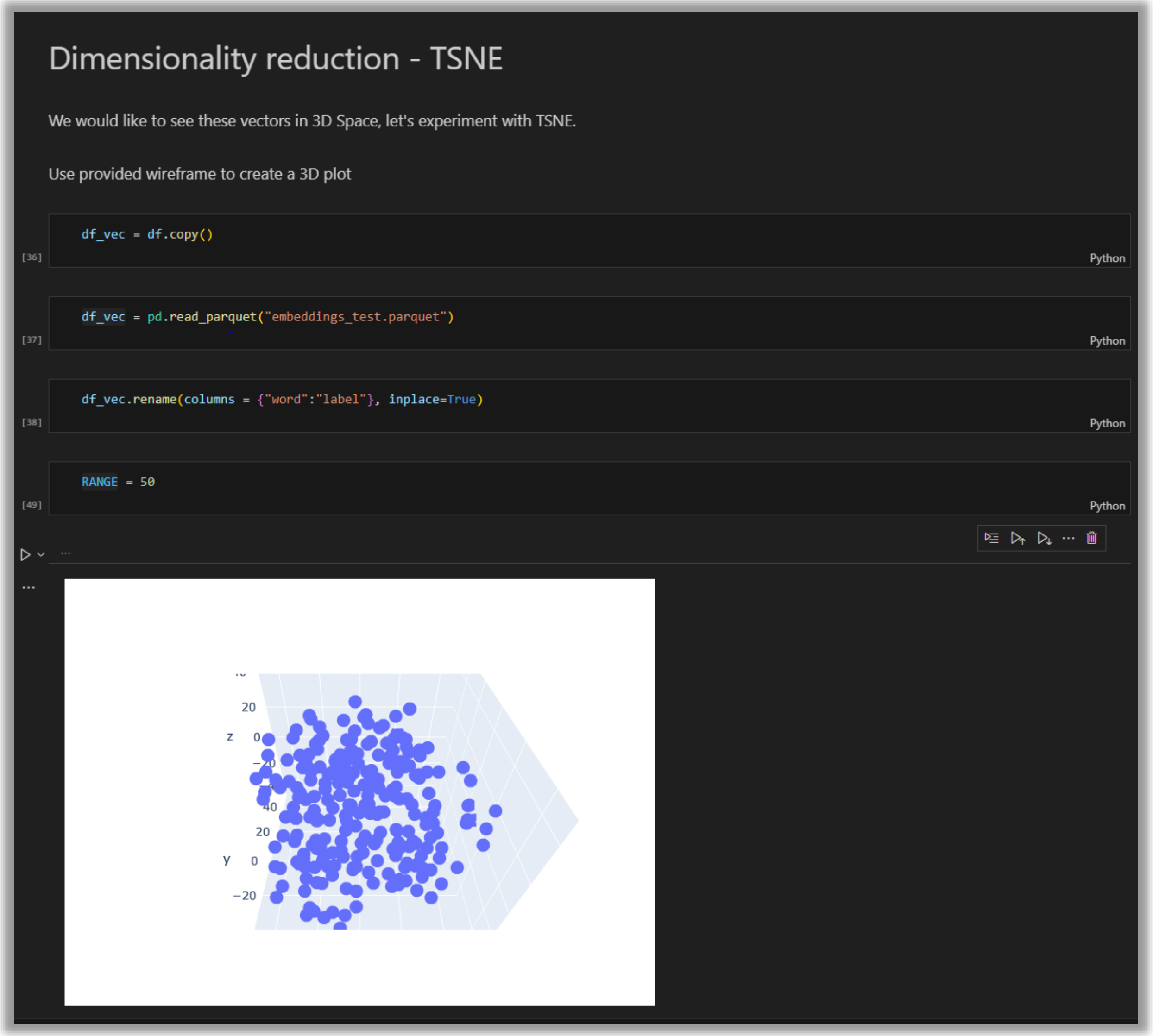
Instructor (2022–2023)
A family of embedding models that incorporate "instruction" prompts to produce task-relevant embeddings (e.g., in the "sentence-transformers" library on Hugging Face).

OpenAI Embeddings: text-embedding-ada-002 (2022+)
A specialized model available through OpenAI's API to create embeddings for semantic search, clustering, classification, etc.
Currently one of the most commonly used API-based embedding solutions.

Cohere Embeddings (2021–present)
Cohere offers text-embedding endpoints via API with a focus on enterprise use.

Jupyter notebook part 2:

Proceed to notebook `W1-tokenization-and-vectorization`



Semantic similarity – advantages and pitfalls

•**Deeper Textual Understanding**: Semantic embeddings capture the contextual and relational meaning of words or phrases, providing a more nuanced understanding than simple keyword matching.

•**Robust to Synonyms and Polysemy**: By mapping semantically similar terms closer together, embedding-based similarity can handle synonyms and words with multiple meanings more effectively than strict keyword-based approaches.

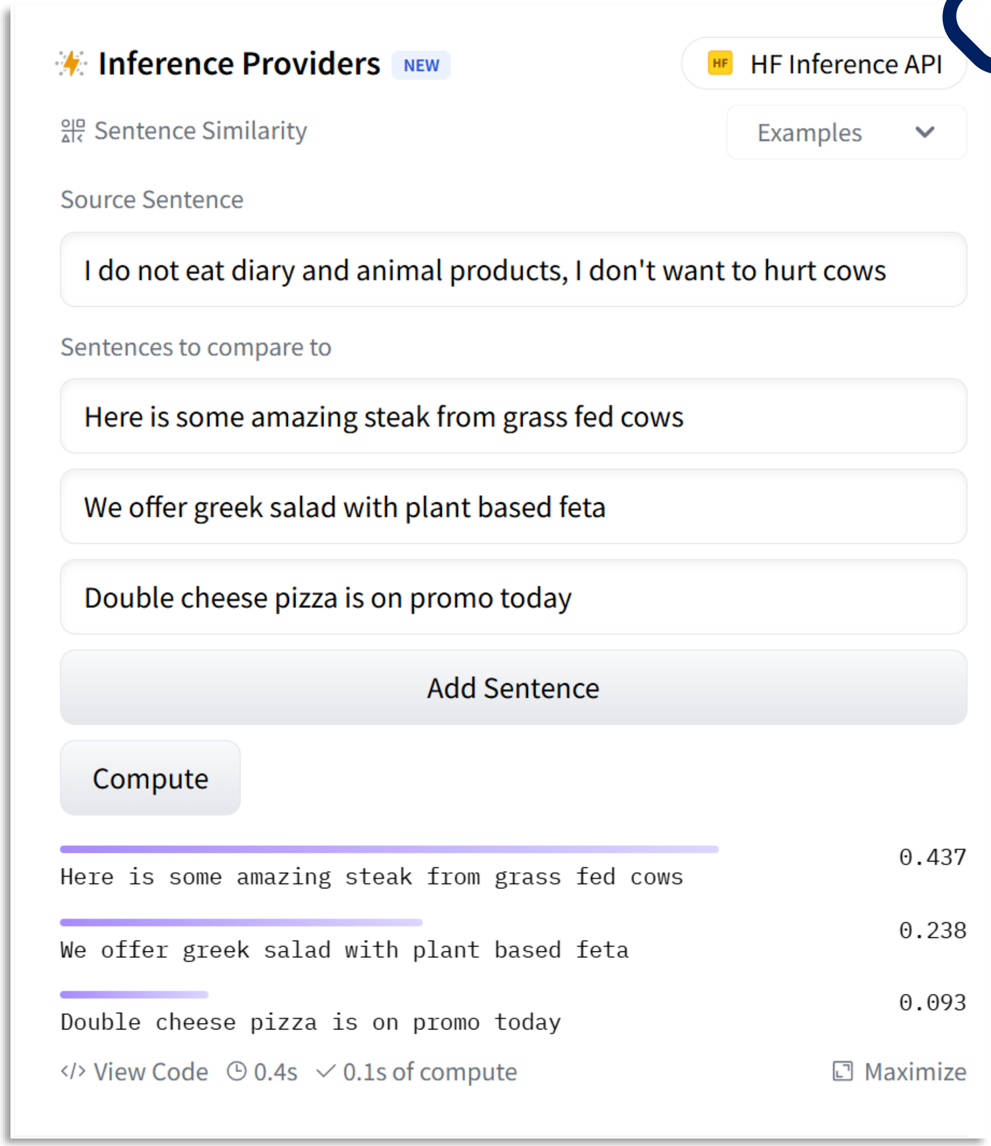
Continuous Representation: Representing text in a continuous vector space allows for smooth comparisons and enables mathematical operations (e.g., vector arithmetic) that can reveal relationships between concepts.

•**Semantic similarity != meaning:** Semantic embeddings usually focus on general context of the sentence, and tend to loose the most important details in more nuanced examples

•**Domain Adaptation Challenges**: Off-the-shelf embeddings may not capture domain-specific vocabulary or nuances well, necessitating domain-specific fine-tuning or retraining.

•**Lack of Explainability**: Similarity scores derived from embeddings can be difficult to interpret, making it challenging to understand why certain texts are considered similar.

Semantic similarity – vectors DO NOT reason



Which sentences are closest?

I am looking for a red Ferrari

This Blue Lamborghini sounds great

Here are red sport shoes

Passat 1.9 TDI has the best engine

Which sentences are closest?

Aby odpowiedzieć na to pytanie musimy przeanalizować art. 10 pkt. 8 kodeksu podstępowania cywilnego

Art. 100 kodeksu podstępowania cywilnego mówi o problemie zadłużenia

W kodeksie cywilnym, artykule 8 jest mowa o karze 10 zł

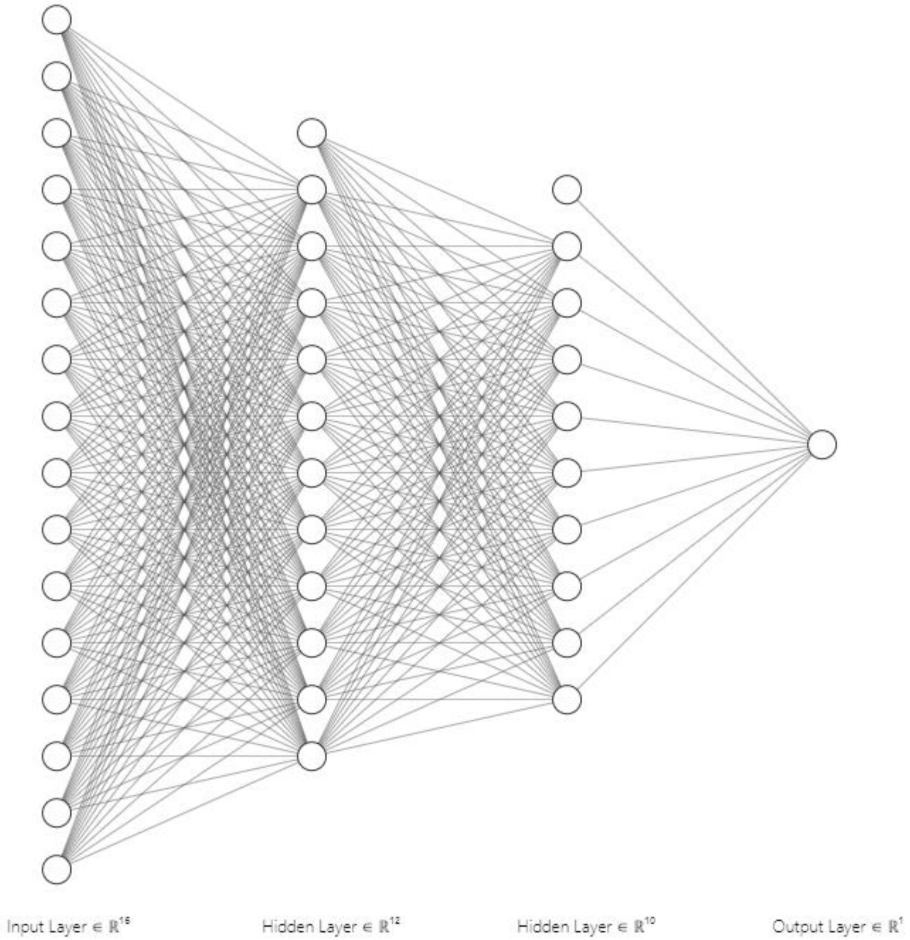
Chodzi tutaj o artykuł 10, pkt 8 kodeksu postępowania cywilnego, który wznawia postępowanie po 10 dniach

Sequence models

Intro to Deep Neural Networks

- Deep Neural Networks can learn complex patterns and handle language data well
- Another advantage of DNNs is the fact, that their training process creates langiage embeddings as a side-product
- We can extract embeddings (with different dimensionality) by disecting one of hidden layers

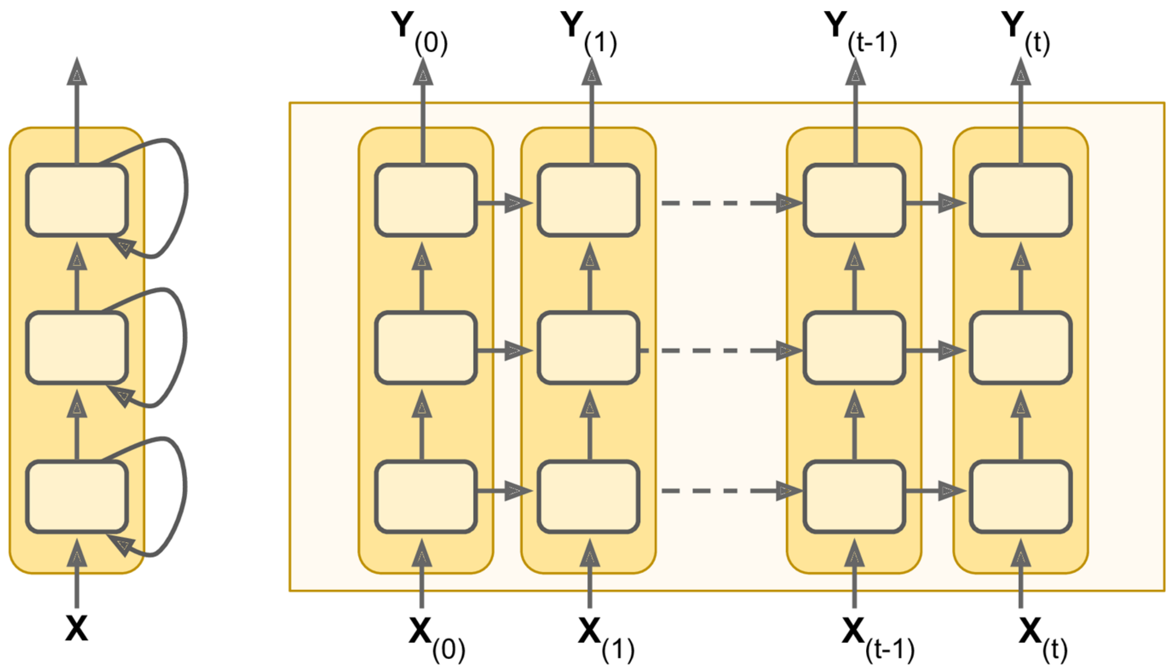
Deep Neural Network diagram



Reccurent Neural Networks

- RNNs is a DNN architecture created specially for sequential data such as timeseries or language
- It has the ability to „remember" outputs from previous steps
- More advanced cells like LSTMs and GRUs make the „memory" aspect of Neural Networks even more robust

Recurrent Neural Networks



Source: https://www.oreilly.com/

Attention & Transformers

Sequence models shortcomings

- Reccurent models rely strongly on sequentiality, where output of previous state is essential for calculating next state
- This makes parallelization impossible and creates issues when working with larger sequence lenghts
- This sequential approach also looses information with the distance. Majority of the information at the begging of the text will be lost towards the end

Key RNN challenges

No paralel computing

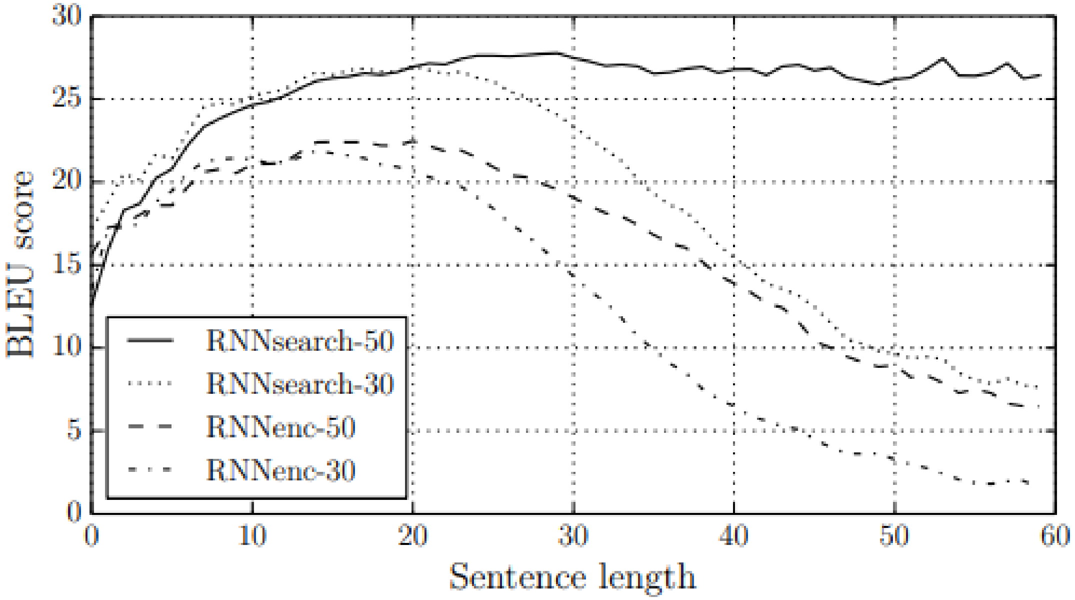
Loss of information

Vanishing gradients

How to implement attention to sequential models

- One way to fix RNNs issue with longer
- sequences is to apply
  attention mechanism
- Traditionally all previous hidden states
  are combined into one vector
- Attention mechanism allows to
  summarize all previous states in a more
  meaningfull way creating a context
  vector

Performance on BLEU translation benchmark by sentence lenghth



Source: „Neural Machine Translation by Jointly Learning to Align and Translate" Dzmitry Bahdanau

Transformers

- Transformer models are a type of Neural Network architecture designed to process sequential data first introduced in "Attention Is All You Need" by Vaswani et al. in 2017.
- Despite their initial use in NLP, they are very versatile and widely adapter in other areas of ML
- They are able to process input sequences in paralel
- Their attention mechanism allows to avoid loss of key information, even if is high distance apart within the sequence

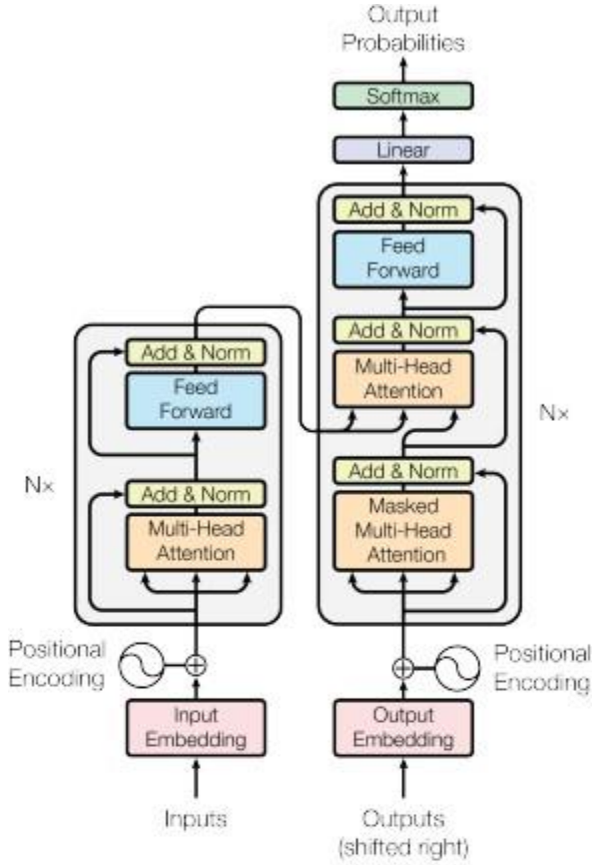


Figure 1: The Transformer - model architecture.

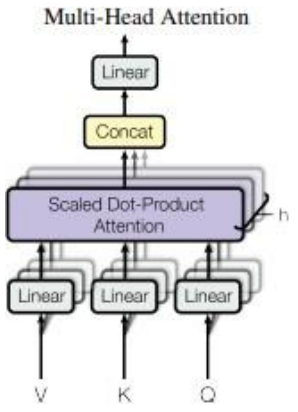
Source „Attention Is All You Need" by Vaswani et al. in 2017.

Transformers – high level overview

Encoder:
Processes the input data (e.g. a sentence) and converts it into a set of attention-based representations. These representations capture the context and relationships between different elements in the input.

Each Encoder layer consist of 2 sublayers:

1. Multi-head self-attention mechanism
2. Fully connected feed-forward Network



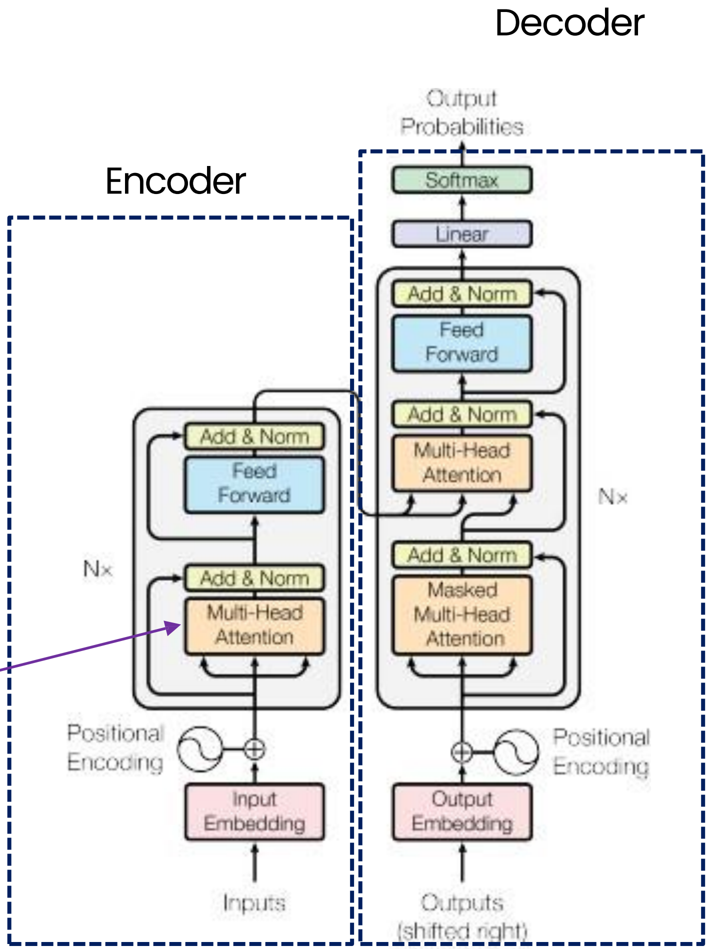


Figure 1: The Transformer - model architecture.

Decoder:
Generates the output data (e.g., the translated sentence in another language) step by step. It uses the representations from the encoder and the previously generated outputs to predict the next element in the sequence.

Each Encoder layer consist of 2 sublayers already present in Decoder + an additional sublayer:

3. Masked Multi-Head attention to prevent positions from attending to subsequent positions

The masking together with output embeddings offset guarantees that the model does not cheat by peeking at future tokens

Source „Attention Is All You Need" by Vaswani et al. in 2017.

Transformers – Self Attention

This mechanism allows the model to weigh the importance of different parts of the input when processing a particular element. For example, in a sentence, the importance or relevance of other words when considering a specific word.

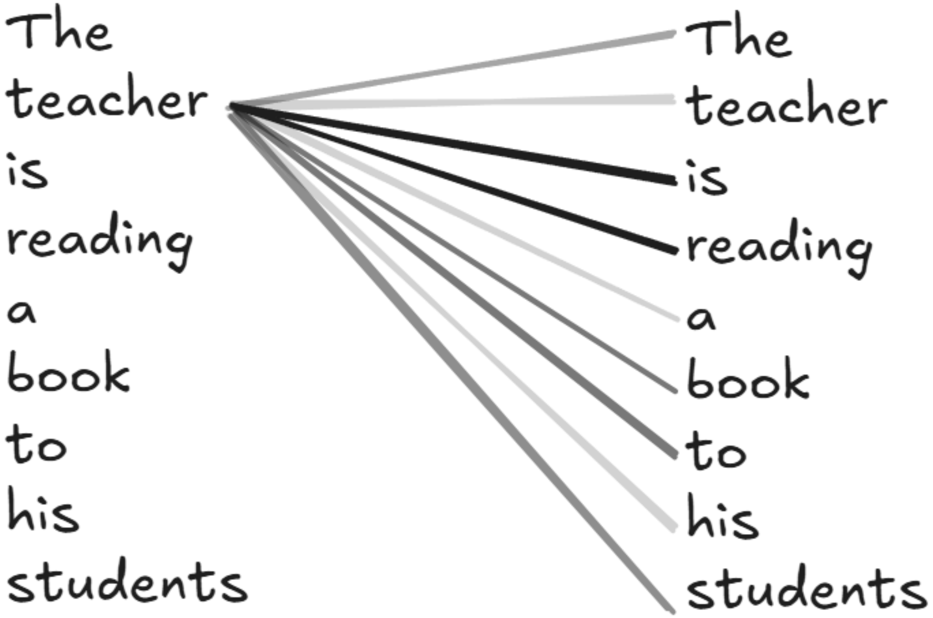
- **Attention Scores**: The model computes scores to determine how much focus to put on other parts of the input for each word in the sequence.

- **Attention Weights**: These scores are then normalized to form a distribution (using functions like softmax), so that they add up to one.

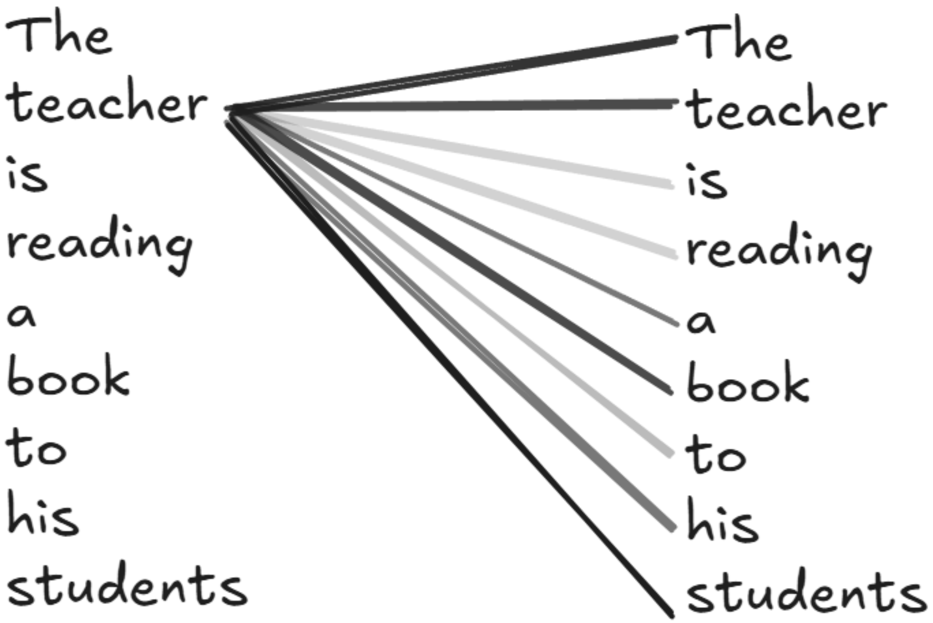
-**Contextual Representation**: Each word's representation is then updated by summing up the representations of all words, weighted by these attention weights. This process ensures that each word’s new representation is a blend of its own and others' based on their relevance.

Transformers – What does multi-head attention mean?

Attention layer 1: focus on verbs



Attention layer 2: focus on nouns and relations



Transformers – how are next words selected?

When a Transformer-based language model (such as GPT) generates text, it does so by predicting the probability distribution of the next token given the context

1.Context encoding: The model processes the input tokens (or the previously generated tokens in the conversation) through its attention layers, producing contextualized representations.

2.Token probability distribution: On the final layer, the model outputs a probability distribution across all possible tokens in its vocabulary for the next position.

3.Sampling or selection: A specific sampling or selection strategy is used to pick the next token from that probability distribution.

Top k sampling example

Machine Learning as an important …

word	proba
Topic	50%
Domain	30%
Concept	15%
Technology	10%

k – controls how many samples we include in the draw
t – controls how we approach probability distribution, when 0 it will always* select the most probabilistic token

Transformers can work in 3 different setups

Decoder Only

Examples
•**GPT** family (e.g., GPT-2, GPT-3, GPT-4)
•**LLaMA** (Meta)
•**BLOOM** (BigScience)
•**PaLM** (Google)

•They process text in a unidirectional manner.

•Used for generative language tasks like next-word prediction, story generation, or code generation.

Encoder-Only

Examples
•**BERT** (Google)
•**RoBERTa** (Meta)
•**DistilBERT** (Hugging Face)

•They typically process the entire input simultaneously (bidirectionally)

•Since there is no decoder, they they produce a contextualized representation of the input (vector), not text

Decoder Only

Examples
•**T5** (Google)
•**BART** (Facebook/Meta)

•Often used for sequence-to-sequence tasks like machine translation, summarization, question-answer generation

Transformer fine-tuning exercise

- Go to [Sentiment_classification_with_BERT.ipynb] where we will leverage Transfer Learning with BERT model in our own tweet sentiment classification

- **Transformers & Attention**

1 Imports

In [1]:
```
import pandas as pd
```

In [2]:
```
import torch
from transformers import DistilBertTokenizerFast, DistilBertForSequenceClassification, Trainer, TrainingArguments
from datasets import load_dataset, load_metric
from sklearn.model_selection import train_test_split
import pandas as pd
from torch.utils.data import Dataset
```

In [3]:
```
from sklearn.model_selection import train_test_split
```

In [50]:
```
import torch
print(torch.__version__)
```

```
1.12.1
```

In [48]:
```
torch.version.cuda
```

In [45]:
```
if torch.cuda.is_available():
    print("CUDA is available. Training on GPU.")
    device = torch.device("cuda")
else:
    print("CUDA is not available. Training on CPU.")
    device = torch.device("cpu")
```

```
CUDA is not available. Training on CPU.
```

In [ ]:

In [4]:
```
data = pd.read_feather("../data/movie_reviews_4k.feather")
```

In [5]:
```
data.shape
```

Out[5]:
```
(4000, 2)
```

In [6]:
```
data
```

Out[6]:

	text	label
0	I wanted to vote zero or lower. I loved the co...	0
1	Karen(Bobbie Phillips)mentions, after one of h...	0
2	This review applies for the cut of the film th...	0
3	The best film on the battle of San Antonio, Te...	1
4	In theory, 'Director's Commentary' should have...	0
...	...	...
3995	Excellent show. Instead of watching the same o...	1
3996	It's hard to believe an "action" packed Jet Li...	0
3997	Me and my girlfriend went to see this movie as...	0
3998	This movie is my all time favorite!!! You real...	1
3999	This movie is very funny. Amitabh Bachan and G...	1

4000 rows × 2 columns

Large Language Models
Training and finetuning

What are LLMs?

Large Language Models (LLMs) are Generative AI models designed to understand, generate, and interact with human language. They can process and generate text, answering questions, creating content, and even engaging in conversation.

LLMs are trained on vast datasets of text from the internet, including books, articles, and websites, using deep learning techniques. This training enables them to learn language patterns, grammar, and context. Despite their impressive abilities the training is still focused on next word prediction

They are used in a variety of applications such as chatbots, content creation, language translation, and sentiment analysis. LLMs are integral to enhancing human-computer interaction and automating complex language tasks.

Most popular LLMs include the pioneers in the domain such as Google's BERT and OpenAi's GPTs. Models from challengers such as Claude or open sourced models from Mistral AI are also catching up in performance.

Key LLM training steps

Data Preparation & Tokenization

- Collection of a vast and diverse datasets such as books, websites, and other textual materials.
- Preprocessing of this data to clean and format it for training
- Tokenizing text into format procesable by models

Model Architecture Design

- Transformer architecture allowed rapid growth of LLMs
- Different architectures and model sizes (params) have significant impact on performance
- BERT uses Bidirectional Encoder only architecture
- GPT uses Decoder only uni-directional architecture
- T5 keeps whole Encoder-Decoder setup

Pre Training

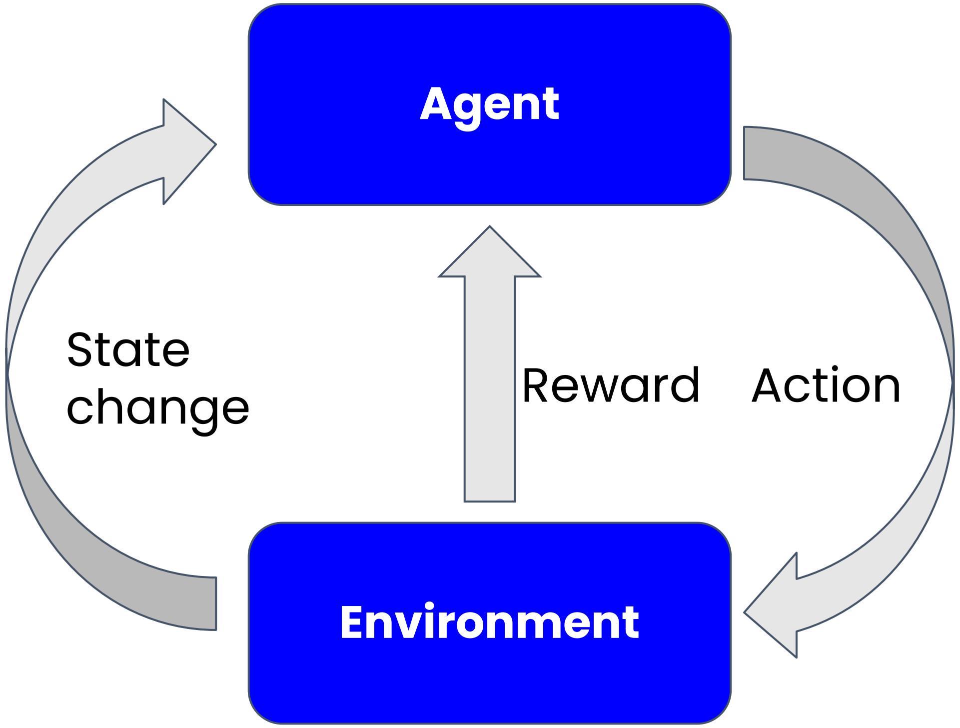
- The model undergoes unsupervised learning, where it learns to predict the next word in a sentence by being fed large amounts of text.
- This stage is critical for the model to learn language patterns, grammar, context, and general world knowledge.

Fine Tuning & RLFH

- Model is further trained on specific tasks and datasets in supervised learning setup
- The model receives feedback from human trainers to correct mistakes and improve its understanding.
- This iterative process helps in refining the model's responses and reducing biases.

Reinforced learning from human feedback

Reinforced learning logic



Reinforced learning logic

Intruct fine-tuned LLM

Human Feedback

- Rank different answers from trained models
- Maximize helpfulness and relevance
- Minimize harm
- Avoid dangerous topics

Human aligned LLM

Memory requirements

Approximate GPU RAM to train 1B params

Memory needed to train model - weights, optimizer, calculating backpropagation

24GB

Memory needed to store model

4GB

GPT 3.5:
175 billion
-> 4 200 GB

GPT 4:
1.76 trillion parameters
-> 42 240 GB

How LLMs changed ML

LLM Development	Classic ML
• No ML expertise needed • No training examples and clear loss function • Reasonable output without training • All communication with model based on natural language prompt • Model aims to follow prompt instructions - loss function is not that clear and easy to change	• ML expertise needed to get started • Training samples needed • Needs to be trained for a specific task • All communication with model based on natural language prompt • Model aims to minimize a loss function

Evaluating LLM performance

Human labeled benchmark datasets

GLUE (General Language Understanding Evaluation) and SuperGLUE Benchmarks:

- Designed to evaluate natural language understanding (NLU).
- Includes a series of tasks like sentiment analysis, question answering, and textual entailment.
- SuperGLUE is an advanced version of GLUE with more challenging tasks.

BLEU (Bilingual Evaluation Understudy) Score for Translation Tasks:

- Commonly used for evaluating the quality of machine-translated text compared to human translations.
- Focuses on how many words and phrases in the machine translation appear in the human translation.
- Commonly used for text translation

ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

- Set of metrics for evaluating automatic summarization and machine translation software in natural language processing.
- It compares an automatically produced summary or translation against a set of reference summaries, typically human-generated, using measures such as the overlap in unigrams, bigrams, trigrams, and longest common subsequences.
- Commonly used for summarization tasks

Massive models bechmark

Massive Multitask Language Understanding (MMLU)

- Comprehensive evaluation framework designed to assess the performance of language models across a wide range of subjects and tasks.
- It includes over 50 different tasks covering a diverse set of topics such as science, humanities, social sciences, and professional domains, aimed at testing the depth and breadth of a model's understanding.
- MMLU is known for its challenging nature, requiring models to not only understand the nuances of human language but also to demonstrate knowledge and reasoning abilities across various disciplines.

Big-Bench

- BIG-bench (Beyond the Imitation Game Benchmark) is an extensive benchmark designed to evaluate and push the limits of large-scale language models in areas like reasoning, creativity, and understanding.
- It encompasses a diverse range of tasks, over 200 in total, that cover a wide array of domains including mathematics, common sense reasoning, linguistics, and even ethical judgment.
- BIG-bench is unique in its focus on tasks that are challenging for current models, aiming to identify the limitations of existing AI and guide future research in natural language understanding and generation.

Chatbot Arena

Chatbot ELO (2024-01-07)

OpenAI

ANTHROP\C

MISTRAL AI_

Model	⭐ Arena Elo rating	📈 MT-bench (score)	MMLU	License
GPT-4-Turbo	1243	9.32		Proprietary
GPT-4-0314	1192	8.96	86.4	Proprietary
GPT-4-0613	1158	9.18		Proprietary
Claude-1	1149	7.9	77	Proprietary
Claude-2.0	1131	8.06	78.5	Proprietary
Mixtral-8x7b-Instruct-v0.1	1121	8.3	70.6	Apache 2.0
Claude-2.1	1117	8.18		Proprietary
GPT-3.5-Turbo-0613	1117	8.39		Proprietary
Gemini Pro	1111		71.8	Proprietary
Claude-Instant-1	1110	7.85	73.4	Proprietary
Tulu-2-DPO-70B	1110	7.89		AI2 ImpACT Low-risk
Yi-34B-Chat	1110		73.5	Yi License

Chatbot Arena

Chatbot ELO (2025-03-06)

Rank* (UB)	Delta	Model	Arena Score	95% CI	Votes	Organization
1	0	GPT-4.5-Preview	1370	+10/-11	3242	OpenAI
2	-1	Grok-3-Preview-02-24	1334	+10/-12	3364	xAI
2	-1	chocolate (Early Grok-3)	1332	+5/-5	13660	xAI
2	2	ChatGPT-4o-latest (2025-01-29)	1341	+4/-6	17221	OpenAI
3	4	DeepSeek-R1	1320	+7/-6	8580	DeepSeek
4	0	Gemini-2.0-Pro-Exp-02-05	1321	+5/-5	15466	Google
4	4	o1-2024-12-17	1323	+4/-4	19785	OpenAI
6	-2	Gemini-2.0-Flash-Thinking-Exp-01-21	1311	+6/-5	17487	Google
6	7	Claude 3.7 Sonnet	1308	+9/-8	4254	Anthropic
8	2	o1-preview	1303	+4/-3	33167	OpenAI
11	-4	Gemini-2.0-Flash-001	1286	+4/-5	13257	Google
11	-1	o3-mini-high	1290	+6/-7	9102	OpenAI
11	-1	Qwen2.5-Max	1284	+4/-6	11930	Alibaba
11	11	Claude 3.5 Sonnet (20241022)	1286	+3/-3	59139	Anthropic

33

Tuning LLMs

Model Fine-tuning

Single task fine tuning

- Retraining all model params on task specific data
- Possible with as little as 1k examples
- Requires significant compute resources and creates a completely separate model for each task
- May lead to catastrophic forgetting, which is basically an equivalent of overfitting

Parameter Efficient Fine Tuning (PEFT)

- Retraining specific part of model params, with keeping majority of model frozen
- Significantly less compute intensive, 90% of params remain frozen and the remaining <10% can be stored for each task and swapped at inference
- Combines general knowledge with new task, reduces risk of catastrophic forgetting

PEFT methods

Reparametrization

- Retrain part of models params using lower dimension
- LoRA is one of most popular use-cases combining model base params, with ones trained for a specific task

Additive

- Add trainable layers or parameters to model
- In “Soft Prompts” prompt tuning additional training happens at input level
- Adapters add additional model layers fine-tuned for specific task, while the backbone models are frozen

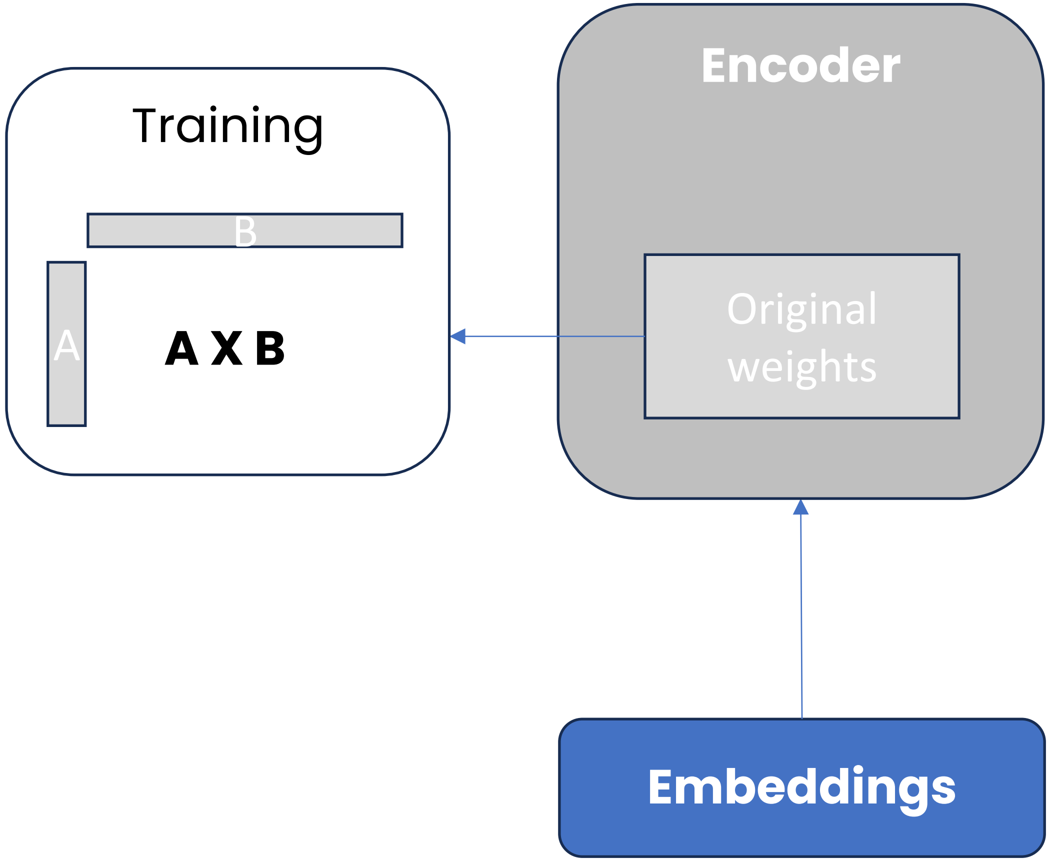
LoRA: Low Rank Adaption of LLMs

Training

- Freeze original model weights
- Replace part of original weight with 2 rank decomposition matrices (with lower dimensionality)
- Train weights only for the smaller matrices

Inference:

- Multiply low rank matrices, to get a matrix with same dimenstions as original weights
- Add product of this multiplication to original weights



Soft prompts fine tuning

- Add additional embeddings, which do not correspond to any token representation
- They will form context embeddings, which help to guide input prompt toward desired outcomes
- Analysing their vector representation in relation to actual words can provide some basic context