An NLP Journey

About me

Lisbon, Portugal

About me

Lisbon, Portugal

Berlin, Germany

About me

2007 BSc. Computer Science
2009 M.Sc. - *“Geographic Information Extraction”*
2015 Ph.D. - *“Large-scale Information Extraction”*

About me

2007 BSc. Computer Science
2009 M.Sc. - *“Geographic Information Extraction”*
2015 Ph.D. - *“Large-scale Information Extraction”*

2016 - 2017 Data Engineer @ Hellofresh
2017 - 2023 NLP Engineer @ Comtravo/TripActions
2023 - 2024 Data Scientist NLP @ Veeva Systems
Since 2024 NLP Engineer @ deepset - working on Haystack

NLP in the last years

Natural Language Processing Tasks

Language Analysis

- **Part-of-speech (POS) tagging**: Classifying words by grammatical categories
- **Syntactic parsing**: Determining grammatical structure of sentences
- **Lemmatization/Stemming**: Reducing words to base/root forms
- **Tokenization**: Segmenting text into words, subwords, or characters

Semantic Understanding

- **Named entity recognition (NER)**: Identifying and classifying named entities
- **Relation extraction**: Identifying relationships between entities
- **Coreference resolution**: Finding expressions referring to the same entity
- **Entity linking**: Connecting named-entities to knowledge base entries

Natural Language Processing Tasks

Text Classification

- **Text classification**: Categorizing texts by topic, genre, etc.
- **Sentiment analysis**: Determining emotional tone or opinion
- **Hate speech/offensive language detection**: Identifying problematic content
- **Fake news detection**: Identifying misleading information

Document Processing

- **Text summarization**: Extractive summarization or Abstractive summarization
- **Information retrieval**: Finding relevant documents/information
- **Document clustering**: Grouping similar documents

Natural Language Processing - early days

1950s-1980s Rule-Based Approaches

- Relied on hand-crafted rules and pattern matching.
- Linguists would create explicit grammatical rules that computers could follow to parse language.

1980s-1990s: Statistical Methods

- **Hidden Markov Models (HMMs)** became popular for part-of-speech tagging and speech recognition
- **Statistical parsing** used probabilistic context-free grammars
- **N-gram language models** predicted words based on preceding context

2000-2012: Machine Learning Approaches

- **Support Vector Machines (SVMs)** became dominant for many classification tasks
- **Conditional Random Fields (CRFs)** excelled at sequence labeling tasks like NER and POS tagging
- **Maximum Entropy Models** (MaxEnt) were widely used for various classification problems
- **Topic modeling** with Latent Dirichlet Allocation (LDA, introduced 2003)

2000 - 2012: Email SPAM classifier

Subject: `WIN a FREE iPhone NOW!!!`
Body: `Congratulations! You have been selected to win a FREE iPhone. Click here to claim your prize.`

2000 - 2012: Email SPAM classifier

Subject: `WIN a FREE iPhone NOW!!!`
Body: `Congratulations! You have been selected to win a FREE iPhone. Click here to claim your prize.`

Feature Extraction: transform the text into input for a machine learning algorithm/classifier

2000 - 2012: Email SPAM classifier

Subject: `WIN a FREE iPhone NOW!!!`
Body: `Congratulations! You have been selected to win a FREE iPhone. Click here to claim your prize.`

Feature Extraction: transform the text into input for a machine learning algorithm/classifier

Text-based features:
- word frequencies, TF-IDF, n-grams

Character-level features:
- exclamation marks, dollar signs, uppercase ratio

Metadata features:
- number of recipients, HTML content, attachments

Structural features
- email length, header format, URL count

2000 - 2012: Email SPAM classifier

Subject: `WIN a FREE iPhone NOW!!!`
Body: `Congratulations! You have been selected to win a FREE iPhone. Click here to claim your prize.`

Contains the word "free"	1
Contains the word "win"	1
Number of exclamation marks	3
All CAPS words count	3
Number of links	1
Email length (number of words)	15
Sender is in known contacts list	0

Vector: `[1, 1, 3, 3, 1, 15, 0]`

2000 - 2012: Email SPAM classifier

Subject: `Meeting tomorrow`
Body: `Hey, can we reschedule the meeting for the next week? I can't make it this week.`

Contains the word "free"	0
Contains the word "win"	0
Number of exclamation marks	0
All CAPS words count	0
Number of links	0
Email length (number of words)	17
Sender is in known contacts list	1

Vector: `[0, 0, 0, 0, 0, 17, 0]`

2000 - 2012: Email SPAM classifier

Train a classifier based on labeled data

`[1, 0, 2, 1, 0, 25, 1]` - NOT SPAM

`[0, 1, 1, 2, 1, 10, 0]` - SPAM

`[0, 0, 3, 0, 2, 30, 1]` - SPAM

`[1, 1, 0, 1, 0, 40, 0]` - NOT SPAM

`[0, 0, 1, 3, 1, 15, 1]` - NOT SPAM

`[1, 0, 0, 0, 2, 20, 0]` - SPAM

`[0, 1, 2, 1, 1, 35, 1]` - NOT SPAM

`[1, 1, 1, 2, 0, 22, 0]` - SPAM

2000 - 2012: Email SPAM classifier

Train a classifier based on labeled data

`[1, 0, 2, 1, 0, 25, 1]` - NOT SPAM

`[0, 1, 1, 2, 1, 10, 0]` - SPAM

`[0, 0, 3, 0, 2, 30, 1]` - SPAM

`[1, 1, 0, 1, 0, 40, 0]` - NOT SPAM

`[0, 0, 1, 3, 1, 15, 1]` - NOT SPAM

`[1, 0, 0, 0, 2, 20, 0]` - SPAM

`[0, 1, 2, 1, 1, 35, 1]` - NOT SPAM

`[1, 1, 1, 2, 0, 22, 0]` - SPAM

- Logistic Regression
- Support Vector Machines
- k-Nearest Neighbors (k-NN)
- Decision Trees / Random Forest
- Naive Bayes
- Gradient Boosting
- XGBoost

2000 - 2012: Email SPAM classifier

Train a classifier based on labeled data

`[1, 0, 2, 1, 0, 25, 1]` - NOT SPAM

`[0, 1, 1, 2, 1, 10, 0]` - SPAM

`[0, 0, 3, 0, 2, 30, 1]` - SPAM

`[1, 1, 0, 1, 0, 40, 0]` - NOT SPAM

`[0, 0, 1, 3, 1, 15, 1]` - NOT SPAM

`[1, 0, 0, 0, 2, 20, 0]` - SPAM

`[0, 1, 2, 1, 1, 35, 1]` - NOT SPAM

`[1, 1, 1, 2, 0, 22, 0]` - SPAM

- Logistic Regression
- Support Vector Machines
- k-Nearest Neighbors (k-NN)
- Decision Trees / Random Forest
- Naive Bayes
- Gradient Boosting
- XGBoost

Text ⇨ **Feature Extraction** ⇨ **Feature Vector** ⇨ **Learning Algorithm**

2012 - 2014: From Feature Extraction to Embedding Vectors

- The distributional hypothesis by Harris (1954), states that each language can be described in terms of a distributional structure, i.e., in terms of the occurrence of parts relative to other parts.

- Firth (1957) explored this idea, based on a word context, popularised by the famous quote you "*shall know a word by the company it keeps*"

- Rubenstein and Goodenough (1965) have shown that a pair of words is highly synonymous if their contexts show a relatively high amount of overlap.

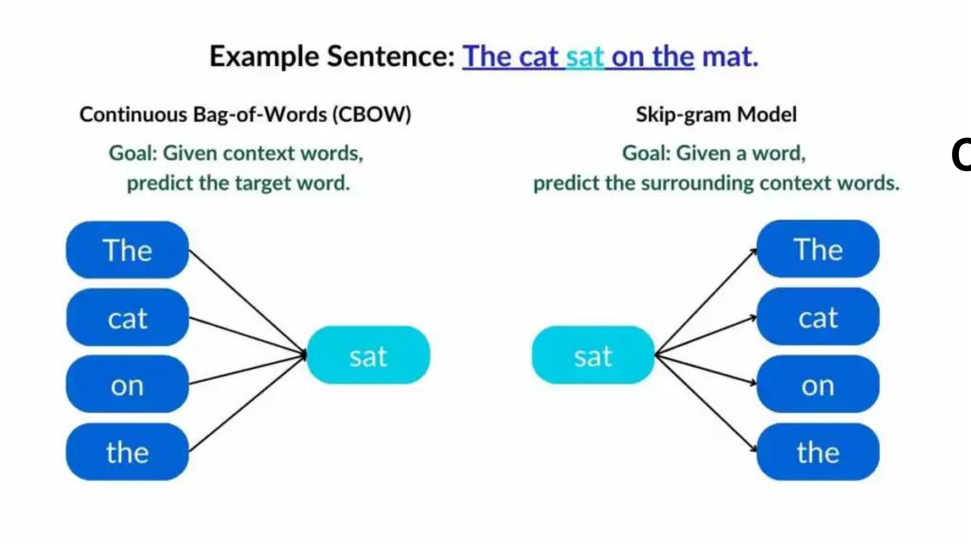
2012 - 2014: From Feature Extraction to Embedding Vectors

- The distributional hypothesis by Harris (1954), states that each language can be described in terms of a distributional structure, i.e., in terms of the occurrence of parts relative to other parts.
- Firth (1957) explored this idea, based on a word context, popularised by the famous quote you "*shall know a word by the company it keeps*"
- Rubenstein and Goodenough (1965) have shown that a pair of words is highly synonymous if their contexts show a relatively high amount of overlap.

Considering the words "*doctor*" and "*physician*"

- Looking at the contexts in which these words appear, there's significant overlap
- Both frequently co-occur with terms like "patient," "hospital," "treatment," "diagnosis," etc
- This distributional similarity reflects their semantic similarity - they both refer to medical professionals who treat patients

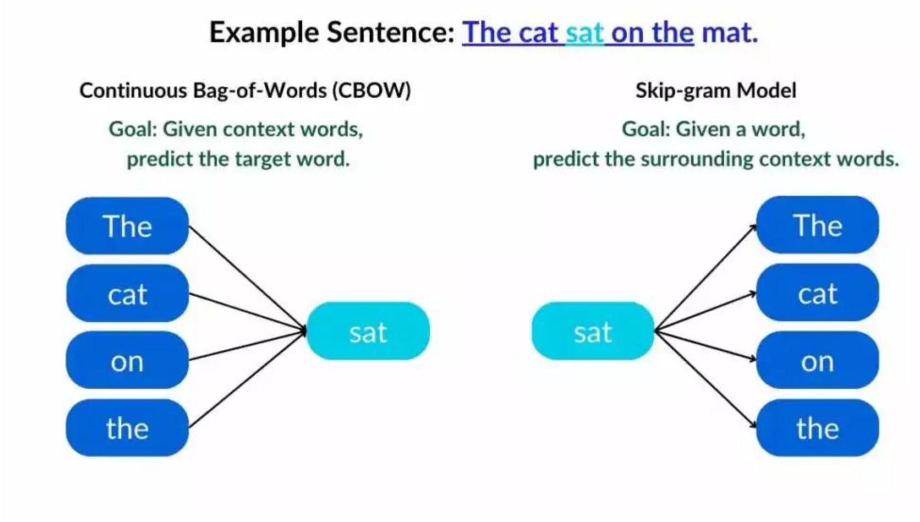
2012 - 2014: From Feature Extraction to Embedding Vectors



CBOW: predicts a target word given its context words:

1. Input: Context words represented as one-hot encoded vectors.
2. Hidden layer: Learns word embeddings by averaging the context word vectors.
3. Output: Predicts the target word.

2012 - 2014: From Feature Extraction to Embedding Vectors



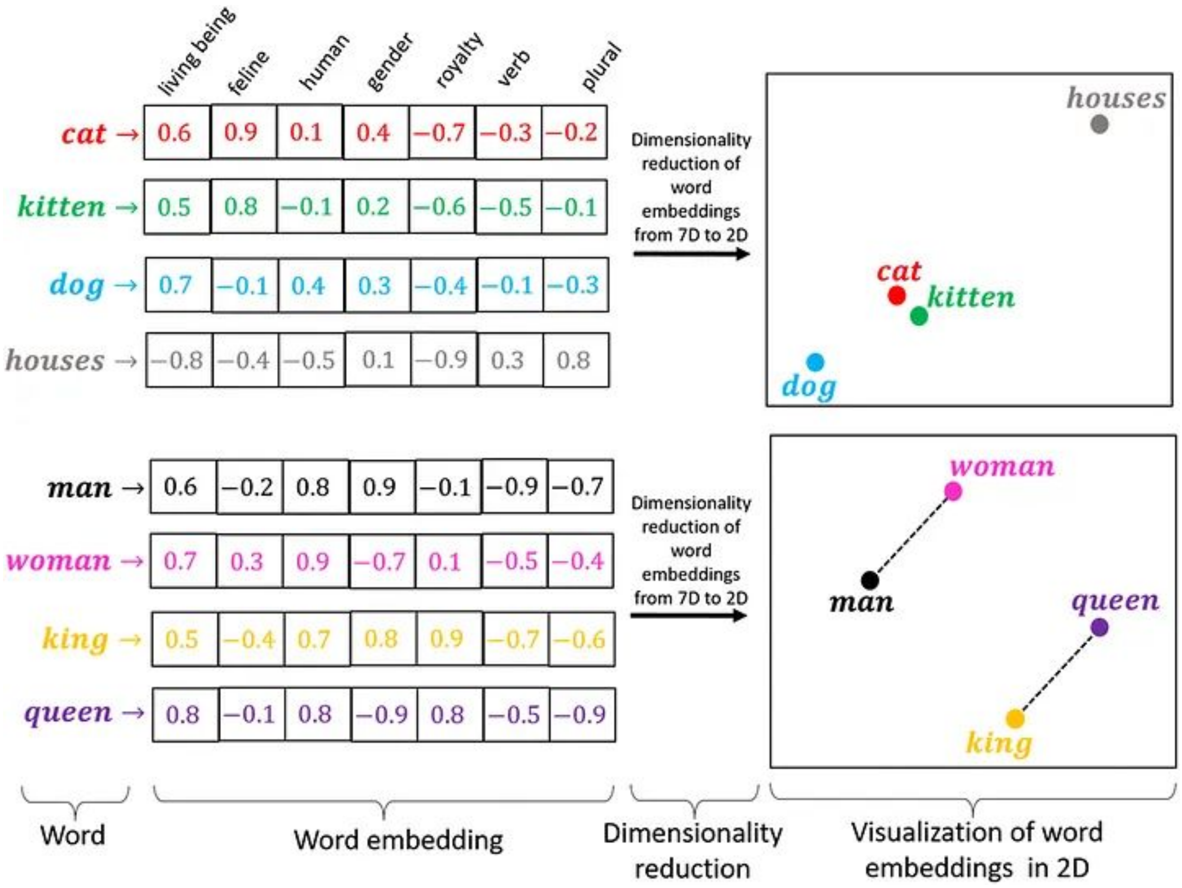
CBOW: predicts a target word given its context words:

1. Input: Context words represented as one-hot encoded vectors.
2. Hidden layer: Learns word embeddings by averaging the context word vectors.
3. Output: Predicts the target word.

By training on large datasets, these models **create word embeddings** capturing semantic and syntactic relationships between words, making them valuable for various NLP tasks.

The resulting embeddings allow for meaningful arithmetic operations on word vectors.
Analogy solving, e.g.: "king - man + woman ≈ queen"

2012 - 2014: From Feature Extraction to Embedding Vectors



2012 - 2014: From feature extraction to Embedding Vectors

Text → **Feature Extraction** → **Feature Vector** → **ML Learning Algorithm**

2012 - 2014: From feature extraction to Embedding Vectors

Text → **Feature Extraction** → **Feature Vector** → **ML Learning Algorithm**

Text → **Word Embeddings** → **ML Learning Algorithm**

2012 - 2014: From feature extraction to Embedding Vectors

Text → **Feature Extraction** → **Feature Vector** → **ML Learning Algorithm**

Text → **Word Embeddings** → **ML Learning Algorithm**

Text → **Word Embeddings** → **Neural Networks**

2012 - 2014: From feature extraction to Embedding Vectors

Text → **Feature Extraction** → **Feature Vector** → **ML Learning Algorithm**

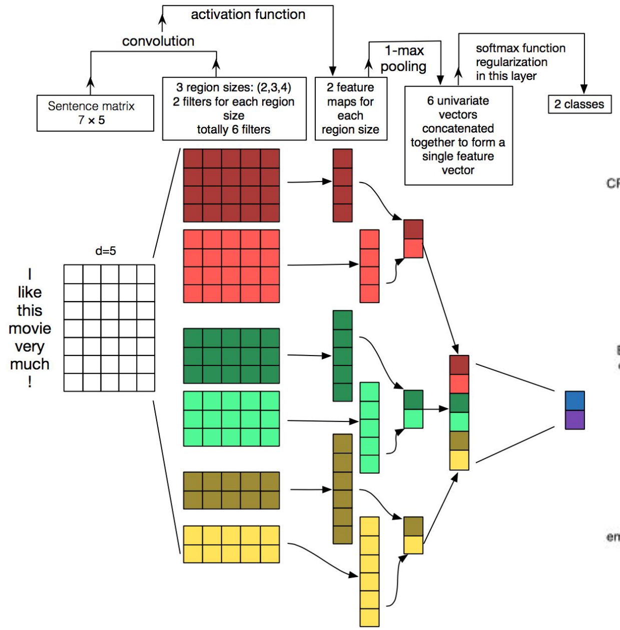
Text → **Word Embeddings** → **ML Learning Algorithm**

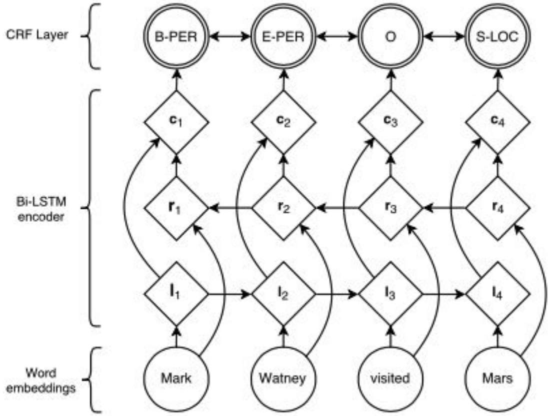
Text → **Word Embeddings** → **Neural Networks**

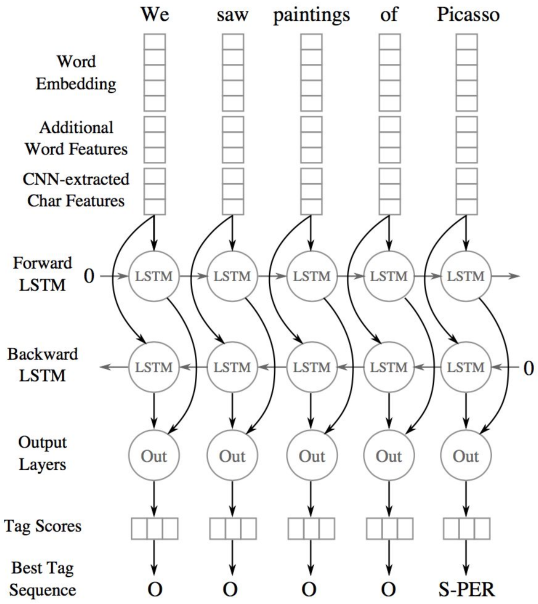
Word Embeddings revolutionised the way almost all NLP tasks can be solved.

Replacing the feature extraction/engineering with embeddings which could then be fed as input to different neural network architectures

2014 - 2017: Embeddings and Neural Networks for NLP







- **Averaging:** created a single vector representation for the entire document by summing up the embeddings of each word and dividing by the number of words
- **Pooling Operations**: Instead of simple averaging, some approaches used other pooling operations like max-pooling or min-pooling over the word embeddings in a document

2014 - 2017: Embeddings and Neural Networks for NLP

Word Embeddings Limitations

- *“I deposited 100 EUR in the* ***bank****.”* vs *“She was enjoying the sunset on the left* ***bank*** *of the river.”*
- **bank** has the same embedding vector
- Couldn't capture polysemy, no contextual understanding of words in sentences

2014 - 2017: Embeddings and Neural Networks for NLP

Word Embeddings Limitations

- *"I deposited 100 EUR in the* ***bank****."* vs *"She was enjoying the sunset on the left* ***bank*** *of the river."*
- **bank** has the same embedding vector
- Couldn't capture polysemy, no contextual understanding of words in sentences

RNN/LSTM Limitations (dominant models but faced several challenges)

Sequential Processing Bottleneck: Processing words one-by-one, making parallelization difficult

Long-range Dependency Problems: Difficulty capturing relationships between distant words

2017 - 2018: Transformer and BERT

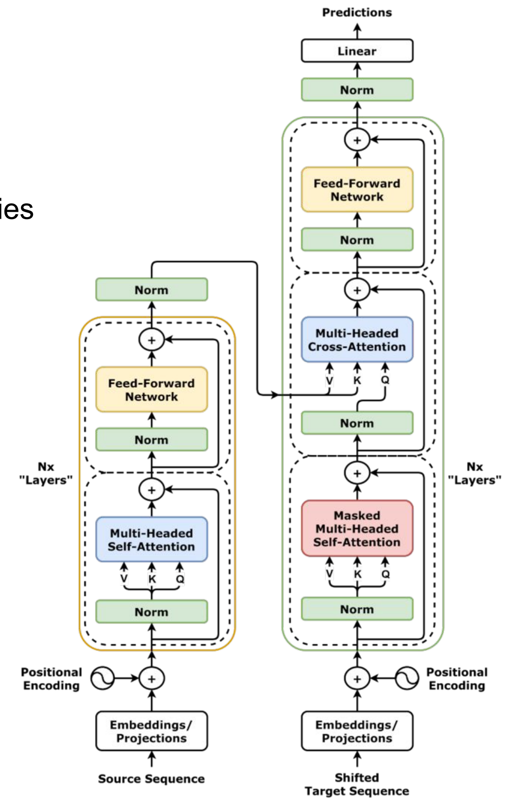
2017 paper "Attention Is All You Need"

- **Self-Attention Mechanism**:
    - Each word can "attend" to all other words, capturing long-range dependencies
- **Parallelizable computation:**
    - no sequential processing
- **Contextual Representations**:
    - same word gets different embeddings in different contexts

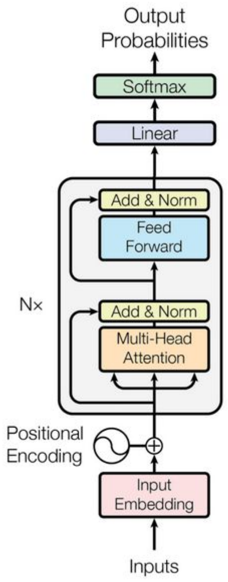
Transformer architecture consists of two main building blocks:

- an encoder

- a decoder



2017 - 2018: Transformer and BERT

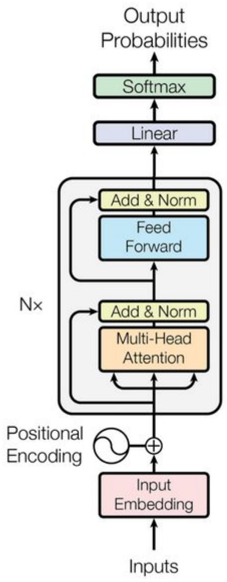


"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"

- **Pre-Training**
  - Predicting words that have been randomly masked out of sentences
  - Determining whether sentence B could follow after sentence A in a text passage
  - Wikipedia (approximately 2.5 billion words)
  - Google's BooksCorpus (approximately 800 million words)
  - Resulted in good initial word representations embeddings

- **Fine-Tuning**
  - Model is fine-tuned to learn a specific task initialised from the pre-trained model parameters
  - BERT achieved good benchmarks results in several NLP tasks

2017 - 2018: Transformer and BERT



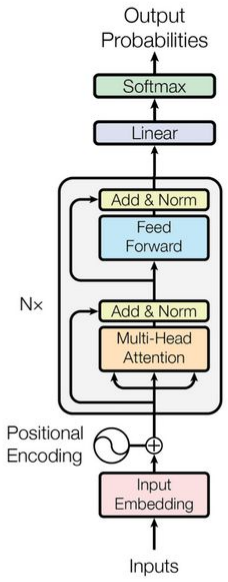
"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"

- **Pre-Training**
  - Predicting words that have been randomly masked out of sentences
  - Determining whether sentence B could follow after sentence A in a text passage
  - Wikipedia (approximately 2.5 billion words)
  - Google's BooksCorpus (approximately 800 million words)
  - Resulted in good initial word representations embeddings

- **Fine-Tuning**
  - Model is fine-tuned to learn a specific task initialised from the pre-trained model parameters
  - BERT achieved good benchmarks results in several NLP tasks

BERT become a powerful feature extractor!

2017 - 2018: Transformer and BERT



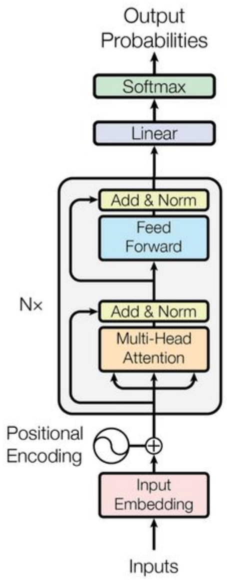
"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"

- **Pre-Training**
  - Predicting words that have been randomly masked out of sentences
  - Determining whether sentence B could follow after sentence A in a text passage
  - Wikipedia (approximately 2.5 billion words)
  - Google's BooksCorpus (approximately 800 million words)
  - Resulted in good initial word representations embeddings

- **Fine-Tuning**
  - Model is fine-tuned to learn a specific task initialised from the pre-trained model parameters
  - BERT achieved good benchmarks results in several NLP tasks

Text ⇨ **Word Embeddings** ⇨ **Neural Networks**

2017 - 2018: Transformer and BERT



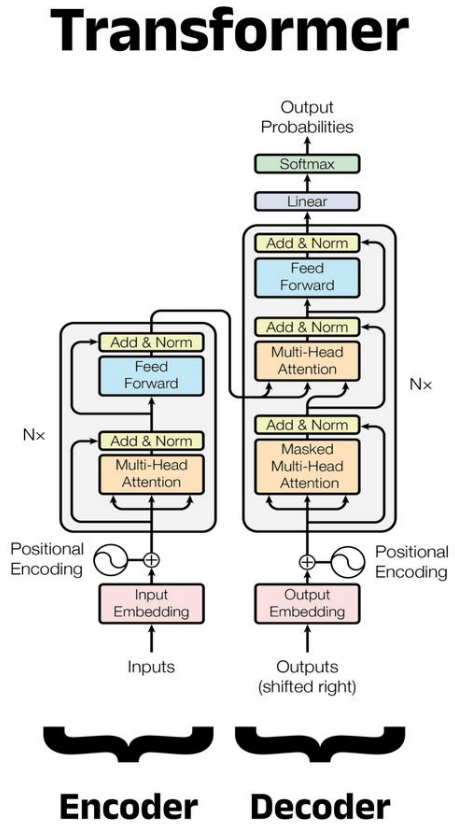
“BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”

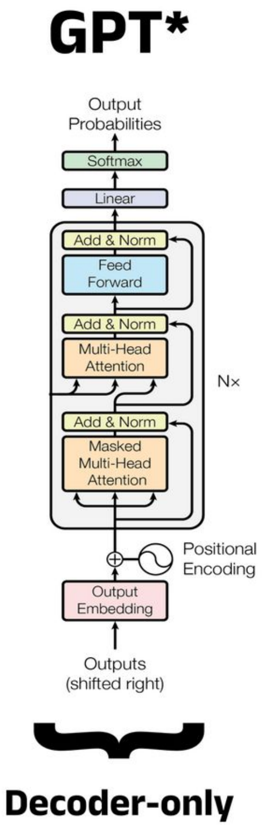
- **Pre-Training**
  - Predicting words that have been randomly masked out of sentences
  - Determining whether sentence B could follow after sentence A in a text passage
  - Wikipedia (approximately 2.5 billion words)
  - Google's BooksCorpus (approximately 800 million words)
  - Resulted in good initial word representations embeddings

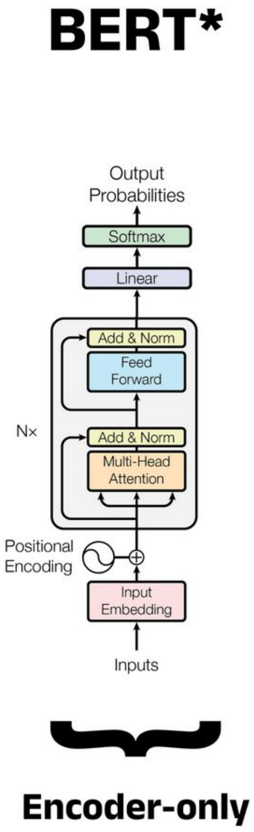
- **Fine-Tuning**
  - Model is fine-tuned to learn a specific task initialised from the pre-trained model parameters
  - BERT achieved good benchmarks results in several NLP tasks

Text ⇨ **BERT Pre-Trained Encoder Transformer** ⇨ **Linear Layer**

2017 - 2018: Transformer and BERT







2019 - 2022: Pre-Training and Scaling

The BERT-like models: **(encoder)**

- Bidirectional context
- Task-specific fine-tuning
- Discriminative tasks

Generative models: **(decoder)**

- Unidirectional (autoregressive) prediction
- Scaling compute and parameters
- Zero/few-shot capabilities through prompting to solve tasks

2019 - 2022: Pre-Training and Scaling

2019:

- **RoBERTa** (Facebook): Robustly optimized BERT pre-training approach **(encoder)**
- **ALBERT** (Google): A Lite BERT with parameter reduction techniques while maintaining performance **(encoder)**
- **DistilBERT** (HuggingFace): Knowledge distillation for creating smaller, faster models **(encoder)**
- **T5** (Google): Text-to-Text Transfer Transformer unifying NLP tasks into a text-to-text format **(seq2seq)**
- **GPT-2** (OpenAI): 1.5B parameter model shows surprising zero-shot abilities; initially "too dangerous" for full release **(decoder)**

2020:

- **GPT-3 (OpenAI):** a language model with 175 billion parameters, demonstrating remarkable abilities in text generation, coding, and creative tasks **(decoder)**

2021:

- **CLIP (OpenAI):** Contrastive Language-Image Pre-training bridging text and visual understanding **(multimodal)**
- **CodeX (OpenAI):** Code generation model fine-tuned on GitHub repositories, precursor to GitHub Copilot **(decoder)**
- **FLAN (Google):** Instruction-tuned model demonstrating improved few-shot learning capabilities across diverse tasks **(decoder)**

2022 Onwards: Decoder-Centric Generative AI

Large Language Models

- **ChatGPT** (November 2022): OpenAI's conversational interface built on GPT-3.5 that mainstream audiences adopted rapidly
- **GPT-4** (March 2023): Multimodal capabilities with significantly improved reasoning
- **LLaMA** (February 2023): Meta's open-source LLM series that catalyzed open-source development
- **Claude** models (2023-2024): Anthropic's models focused on helpfulness and harmlessness

2022 Onwards: Decoder-Centric Generative AI

Large Language Models

- **ChatGPT** (November 2022): OpenAI's conversational interface built on GPT-3.5 that mainstream audiences adopted rapidly
- **GPT-4** (March 2023): Multimodal capabilities with significantly improved reasoning
- **LLaMA** (February 2023): Meta's open-source LLM series that catalyzed open-source development
- **Claude** models (2023-2024): Anthropic's models focused on helpfulness and harmlessness

Multimodal Generative Models

- **DALL-E 2** (April 2022) and **DALL-E 3** (2023): Text-to-image generation with improved coherence
- **Stable Diffusion** (August 2022): Open-source text-to-image model that revolutionized accessibility
- **Midjourney** (2022-2023): Text-to-image service with distinctive aesthetic quality

2022 Onwards: Decoder-Centric Generative AI

Large Language Models

- **ChatGPT** (November 2022): OpenAI's conversational interface built on GPT-3.5 that mainstream audiences adopted rapidly
- **GPT-4** (March 2023): Multimodal capabilities with significantly improved reasoning
- **LLaMA** (February 2023): Meta's open-source LLM series that catalyzed open-source development
- **Claude** models (2023-2024): Anthropic's models focused on helpfulness and harmlessness

Multimodal Generative Models

- **DALL-E 2** (April 2022) and **DALL-E 3** (2023): Text-to-image generation with improved coherence
- **Stable Diffusion** (August 2022): Open-source text-to-image model that revolutionized accessibility
- **Midjourney** (2022-2023): Text-to-image service with distinctive aesthetic quality

2023-2024: emerging trends - what's next?

- **Tool Use**: Models effectively leveraging external tools and APIs to extend capabilities
- **Agentic Systems**: LLMs orchestrating complex tasks with planning capabilities
- **Local Deployment**: Smaller, more efficient models running on personal devices

Thank you :)

davidsbatista.net

linkedin.com/in/dsbatista

github.com/davidsbatista

Haystack: RAG and Agents framework

2021~2022 - Retrieval-Augmented Generation (RAG): Combining generation with external knowledge retrieval

1. **Retrieval-Based Systems**: fetch relevant documents from a DB based on a query.

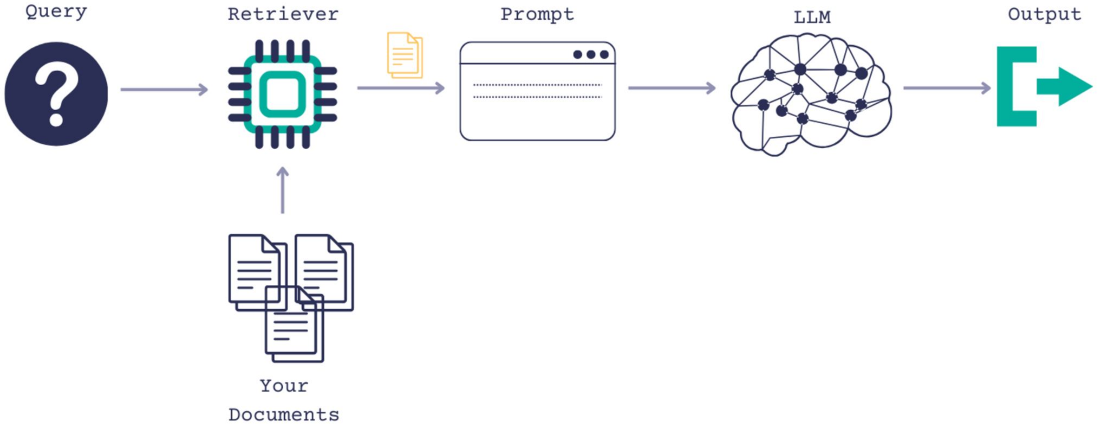
2. **LLMs**: generate responses based on the input query using the language model.

3. **Retrieval-Augmented Generation (RAG)**: RAG combines the strengths of both approaches. It first retrieves relevant documents or passages based on the query and then uses these retrieved pieces of information to generate a more informed and accurate response. This helps in grounding the generated responses in factual information, reducing hallucinations, and improving overall accuracy.

Haystack: RAG and Agents framework

2021~2022 - Retrieval-Augmented Generation (RAG): Combining generation with external knowledge retrieval

1. **Retrieval-Based Systems**: fetch relevant documents from a DB based on a query.
2. **LLMs**: generate responses based on the input query using the language model.
3. **Retrieval-Augmented Generation (RAG)**: RAG combines the strengths of both approaches. It first retrieves relevant documents or passages based on the query and then uses these retrieved pieces of information to generate a more informed and accurate response. This helps in grounding the generated responses in factual information, reducing hallucinations, and improving overall accuracy.



2012 - 2014: From Feature Extraction to Embedding Vectors