Choosing the right language model for your NLP use case

A guide to understanding, selecting and deploying Large Language Models

Janna Lipenkova · Follow

Published in Towards Data Science
15 min read · Sep 26, 2022

Listen Share More

Large Language Models (LLMs) are Deep Learning models trained to produce text. With this impressive ability, LLMs have become the backbone of modern Natural

Language Processing (NLP). Traditionally, they are pre-trained by academic institutions and big tech companies such as OpenAI, Microsoft and NVIDIA. Most of them are then made available for public use. This plug-and-play approach is an important step towards large-scale AI adoption — instead of spending huge resources on the training of models with general linguistic knowledge, businesses can now focus on fine-tuning existing LLMs for specific use cases.

However, picking the right model for your application can be tricky. Users and other stakeholders have to make their way through a vibrant landscape of language models and related innovations. These improvements address different components of the language model including its training data, pre-training objective, architecture and fine-tuning approach — you could write a book on each of these aspects. On top of all this research, the marketing buzz and the intriguing aura of Artificial General Intelligence around huge language models obfuscate things even more.

In this article, I explain the main concepts and principles behind LLMs. The goal is to provide non-technical stakeholders with an intuitive understanding as well as a language for efficient interaction with developers and AI experts. For broader coverage, the article includes analyses that are rooted in a large number of NLP-related publications. While we will not dive into mathematical details of language models, these can be easily retrieved from the references.

The article is structured as follows: first, I situate language models in the context of the evolving NLP landscape. The second section explains how LLMs are built and pre-trained. Finally, I describe the fine-tuning process and provide some guidance on model selection.

The world of language models

Bridging the human-machine gap

Language is a fascinating skill of the human mind — it is a universal protocol for communicating our rich knowledge of the world, and also more subjective aspects such as intents, opinions and emotions. In the history of AI, there have been multiple waves of research to approximate (“model”) human language with mathematical means. Before the era of Deep Learning, representations were based on simple algebraic and probabilistic concepts such as one-hot representations of words, sequential probability models and recursive structures. With the evolution of Deep Learning in the past years, linguistic representations have increased in precision, complexity and expressiveness.

In 2018, BERT was introduced as the first LLM on the basis of the new Transformer architecture. Since then, Transformer-based LLMs have gained strong momentum. Language modelling is especially attractive due to its universal usefulness. While many real-world NLP tasks such as sentiment analysis, information retrieval and information extraction do not need to generate language, the assumption is that a model that produces language also has the skills to solve a variety of more specialised linguistic challenges.

Size matters

Learning happens based on parameters — variables that are optimized during the training process to achieve the best prediction quality. As the number of parameters increases, the model is able to acquire more granular knowledge and improve its predictions. Since the introduction of the first LLMs in 2017–2018, we saw an exponential explosion in parameter sizes — while breakthrough BERT was trained with 340M parameters, Megatron-Turing NLG, a model released in 2022, is trained with 530B parameters — a more than thousand-fold increase.

Figure 1: The parameter sizes of language models increase exponentially over time [11]

Thus, the mainstream keeps wowing the public with ever bigger amounts of parameters. However, there have been critical voices pointing out that model performance is not increasing at the same rate as model size. On the other side, model pre-training can leave a considerable carbon footprint. Downsizing efforts have countered the brute-force approach to make progress in language modelling more sustainable.

The life of a language model

The LLM landscape is competitive and innovations are short-lived. The following chart shows the top-15 most popular LLMs in the timespan 2018–2022, along with their share-of-voice over time:

Figure 2: Mentions and share-of-voice of the top-15 most popular language models [12]

We can see that most models fade in popularity after a relatively short time. To stay cutting-edge, users should monitor the current innovations and evaluate whether an upgrade would be worthwhile.

Most LLMs follow a similar lifecycle: first, at the "upstream", the model is pre-trained. Due to the heavy requirements on data size and compute, it is mostly a privilege of large tech companies and universities. Recently, there have also been some collaborative efforts (e.g. the BigScience workshop) for the joint advancement of the LLM field. A handful of well-funded startups such as Cohere and AI21 Labs also provide pre-trained LLMs.

After the release, the model is adopted and deployed at the "downstream" by application-focussed developers and businesses. At this stage, most models require an extra fine-tuning step to specific domains and tasks. Others, like GPT-3, are more convenient in that they can learn a variety of linguistic tasks directly during prediction (zero- or few-shot prediction).

Finally, time knocks at the door and a better model comes around the corner — either with an even larger number of parameters, more efficient use of hardware or a more fundamental improvement to the modelling of human language. Models that brought about substantial innovations can give birth to whole model families. For example, BERT lives on in BERT-QA, DistilBERT and RoBERTa, which are all based on the original architecture.

In the next sections, we will look at the first two phases in this lifecycle — the pre-training and the fine-tuning for deployment.

Pre-training: how LLMs are born

Most teams and NLP practitioners will not be involved in the pre-training of LLMs, but rather in their fine-tuning and deployment. However, to successfully pick and use a model, it is important to understand what is going on "under the hood". In this section, we will look at the basic ingredients of an LLM:

- Training data
- Input representation
- Pre-training objective
- Model architecture (encoder-decoder)

Each of these will affect not only the choice, but also the fine-tuning and deployment of your LLM.

Training data

The data used for LLM training is mostly text data covering different styles, such as literature, user-generated content and news data. After seeing a variety of different text types, the resulting models become aware of the fine details of language. Other than text data, code is regularly used as input, teaching the model to generate valid programs and code snippets.

Unsurprisingly, the quality of the training data has a direct impact on model performance — and also on the required size of the model. If you are smart in preparing the training data, you can improve model quality while reducing its size. One example is the T0 model, which is 16 times smaller than GPT-3 but outperforms it on a range of benchmark tasks. Here is the trick: instead of just using any text as training data, it works directly with task formulations, thus making its learning signal much more focussed. Figure 3 illustrates some training examples.

Figure 3: T0 is trained on explicit task formulations for a wide range of linguistic tasks

A final note on training data: we often hear that language models are trained in an unsupervised manner. While this makes them appealing, it is technically wrong.

Instead, well-formed text already provides the necessary learning signals, sparing us the tedious process of manual data annotation. The labels to be predicted correspond to past and/or future words in a sentence. Thus, annotation happens automatically and at scale, making possible the relatively quick progress in the field.

Input representation

Once the training data is assembled, we need to pack it into form that can be digested by the model. Neural networks are fed with algebraic structures (vectors and matrices), and the optimal algebraic representation of language is an ongoing quest — reaching from simple sets of words to representations containing highly differentiated context information. Each new step confronts researchers with the endless complexity of natural language, exposing the limitations of the current representation.

The basic unit of language is the word. In the beginnings of NLP, this gave rise to the naive **bag-of-words** representation that throws all words from a text together, irrespectively of their ordering. Consider these two examples:

In the bag-of-words world, these sentences would get exactly the same representation since they consist of the same words. Clearly, it embraces only a small part of their meaning.

Sequential representations accommodate information about word order. In Deep Learning, the processing of sequences was originally implemented in order-aware **Recurrent Neural Networks** (RNN).[2] However, going one step further, the underlying structure of language is not purely sequential but hierarchical. In other words, we are not talking about lists, but about trees. Words that are farther apart can actually have stronger syntactic and semantic ties than neighbouring words. Consider the following example:

Here, *her* refers to *the girl*. When an RNN reaches the end of the sentence and finally sees *her*, its memory of the beginning of the sentence might already be fading, thus not allowing it to recover this relationship.

To solve these long-distance dependencies, more complex neural structures were proposed to build up a more differentiated memory of the context. The idea is to keep words that are relevant for future predictions in memory while forgetting the other words. This was the contribution of Long-Short Term Memory (LSTM)[3] cells and Gated Recurrent Units (GRUs)[4]. However, these models don't optimise for specific positions to be predicted, but rather for a generic future context. Moreover, due to their complex structure, they are even slower to train than traditional RNNs.

Finally, people have done away with recurrence and proposed the **attention mechanism**, as incorporated in the **Transformer** architecture.[5] Attention allows the model to focus back and forth between different words during prediction. Each word is weighted according to its relevance for the specific position to be predicted. For the above sentence, once the model reaches the position of *her*, *girl* will have a higher weight than *at*, despite the fact that it is much farther away in the linear order.

To date, the attention mechanism comes closest to the biological workings of the human brain during information processing. Studies have shown that attention learns hierarchical syntactic structures, incl. a range of complex syntactic phenomena (cf. the Primer on BERTology and the papers referenced therein). It also allows for parallel computation and, thus, faster and more efficient training.

Pre-training objectives

With the appropriate training data representation in place, our model can start learning. There are three generic objectives used for pre-training language models: sequence-to-sequence transduction, autoregression and auto-encoding. All of them require the model to master broad linguistic knowledge.

The original task addressed by the encoder-decoder architecture as well as the Transformer model is **sequence-to-sequence transduction**: a sequence is transduced into a sequence in a different representation framework. The classical sequence-to-sequence task is machine translation, but other tasks such as summarisation are frequently formulated in this manner. Note that the target sequence is not necessarily text — it can also be other unstructured data such as images as well as structured data such as programming languages. An example of sequence-to-sequence LLMs is the BART family.

The second task is **autoregression**, which is also the original language modelling objective. In autoregression, the model learns to predict the next output (token) based on previous tokens. The learning signal is restricted by the unidirectionality of the enterprise — the model can only use information from the right or from the left of the predicted token. This is a major limitation since words can depend both on past as well as on future positions. As an example, consider how the verb *written* impacts the following sentence in both directions:

Here, the position of *paper* is restricted to something that is writable, while the position of *student* is restricted to a human or, anyway, another intelligent entity capable of writing.

Many of the LLMs making today’s headlines are autoregressive, incl. the GPT family, PaLM and BLOOM.

The third task — auto-encoding — solves the issue of unidirectionality. Auto-encoding is very similar to the learning of classical word embeddings.[6] First, we corrupt the training data by hiding a certain portion of tokens — typically 10–20% — in the input. The model then learns to reconstruct the correct inputs based on the surrounding context, taking into account both the preceding and the following tokens. The typical example of auto-encoders is the BERT family, where BERT stands for **Bidirectional** Encoder Representations from Transformers.

Model architecture (encoder-decoder)

The basic building blocks of a language model are the encoder and the decoder. The encoder transforms the original input into a high-dimensional algebraic representation, also called a "hidden" vector. Wait a minute — hidden? Well, in reality there are no big secrets at this point. Of course you can look at this representation, but a lengthy vector of numbers will not convey anything meaningful to a human. It takes the mathematical intelligence of our model to deal with it. The decoder reproduces the hidden representation in an intelligible form such as another language, programming code, an image etc.

Figure 4: Basic schema of an encoder-decoder architecture (example of English-German translation)

The encoder-decoder architecture was originally introduced for Recurrent Neural Networks. Since the introduction of the attention-based Transformer model, traditional recurrence has lost its popularity while the encoder-decoder idea lives on. Most Natural Language Understanding (NLU) tasks rely on the encoder, while Natural Language Generation (NLG) tasks need the decoder and sequence-to-sequence transduction requires both components.

We will not go into the details of the Transformer architecture and the attention mechanism here. For those who want to master the details, be prepared to spend a good amount of time to wrap your head around it. Beyond the original paper, [7] and [8] provide excellent explanations. For a lightweight introduction, I recommend the corresponding sections in Andrew Ng's Sequence models course.

Using language models in the real world

Fine-tuning

Language modelling is a powerful upstream task — if you have a model that successfully generates language, congratulations — it is an intelligent model. However, the business value of having a model bubbling with random text is limited. Instead, NLP is mostly used for more targeted **downstream tasks** such as sentiment analysis, question answering and information extraction. This is the time to apply **transfer learning** and reuse the existing linguistic knowledge for more specific challenges. During fine-tuning, a portion of the model is "freezed" and the rest is further trained with domain- or task-specific data.

Explicit fine-tuning adds complexity on the path towards LLM deployment. It can also lead to model explosion, where each business task requires its own fine-tuned model, escalating to an unmaintainable variety of models. So, folks have made an effort to get rid of the fine-tuning step using few- or zero-shot learning (e.g. in GPT-3 [9]). This learning happens on-the-fly during prediction: the model is fed with a "prompt" — a task description and potentially a few training examples — to guide its predictions for future examples.

While much quicker to implement, the convenience factor of zero- or few-shot learning is counterbalanced by its lower prediction quality. Besides, many of these models need to be accessed via cloud APIs. This might be a welcome opportunity at the beginning of your development — however, at more advanced stages, it can turn into another unwanted external dependency.

Picking the right model for your downstream task

Looking at the continuous supply of new language models on the AI market, selecting the right model for a specific downstream task and staying in synch with the state-of-the-art can be tricky.

Research papers normally benchmark each model against specific downstream tasks and datasets. Standardised task suites such as SuperGLUE and BIG-bench allow for unified benchmarking against a multitude of NLP tasks and provide a basis for comparison. Still, we should keep in mind that these tests are prepared in a highly controlled setting. As of today, the generalisation capacity of language models is rather limited — thus, the transfer to real-life datasets might significantly affect model performance. The evaluation and selection of an appropriate model should involve experimentation on data that is as close as possible to the production data.

As a rule of thumb, the pre-training objective provides an important hint: autoregressive models perform well on text generation tasks such as conversational AI, question answering and text summarisation, while auto-encoders excel at “understanding” and structuring language, for example for sentiment analysis and various information extraction tasks. Models intended for zero-shot learning can theoretically perform all kinds of tasks as long as they receive appropriate prompts — however, their accuracy is generally lower than that of fine-tuned models.

To make things more concrete, the following chart shows how popular NLP tasks are associated with prominent language models in the NLP literature. The associations are computed based on multiple similarity and aggregation metrics, incl. embedding similarity and distance-weighted co-occurrence. Model-task pairs with higher scores, such as BART / Text Summarization and LaMDA / Conversational AI, indicate a good fit based on historical data.

Figure 5: Association strengths between language models and downstream tasks [12]

Key takeaways

In this article, we have covered the basic notions of LLMs and the main dimensions where innovation is happening. The following table provides a summary of the key features for the most popular LLMs:

Table 1: Summary of the features of the most popular Large Language Models

Let’s summarise some general guidelines for the selection and deployment of LLMs:

1. When evaluating potential models, be clear about where you are in your AI journey:

- At the beginning, it might be a good idea to experiment with LLMs deployed via cloud APIs.
- Once you have found product-market fit, consider hosting and maintaining your model on your side to have more control and further sharpen model performance to your application.

2. To align with your downstream task, your AI team should create a short-list of models based on the following criteria:

- Benchmarking results in the academic literature, with a focus on your [illegible] task

- Alignment between the pre-training objective and downstream task: consider auto-encoding for NLU and autoregression for NLG

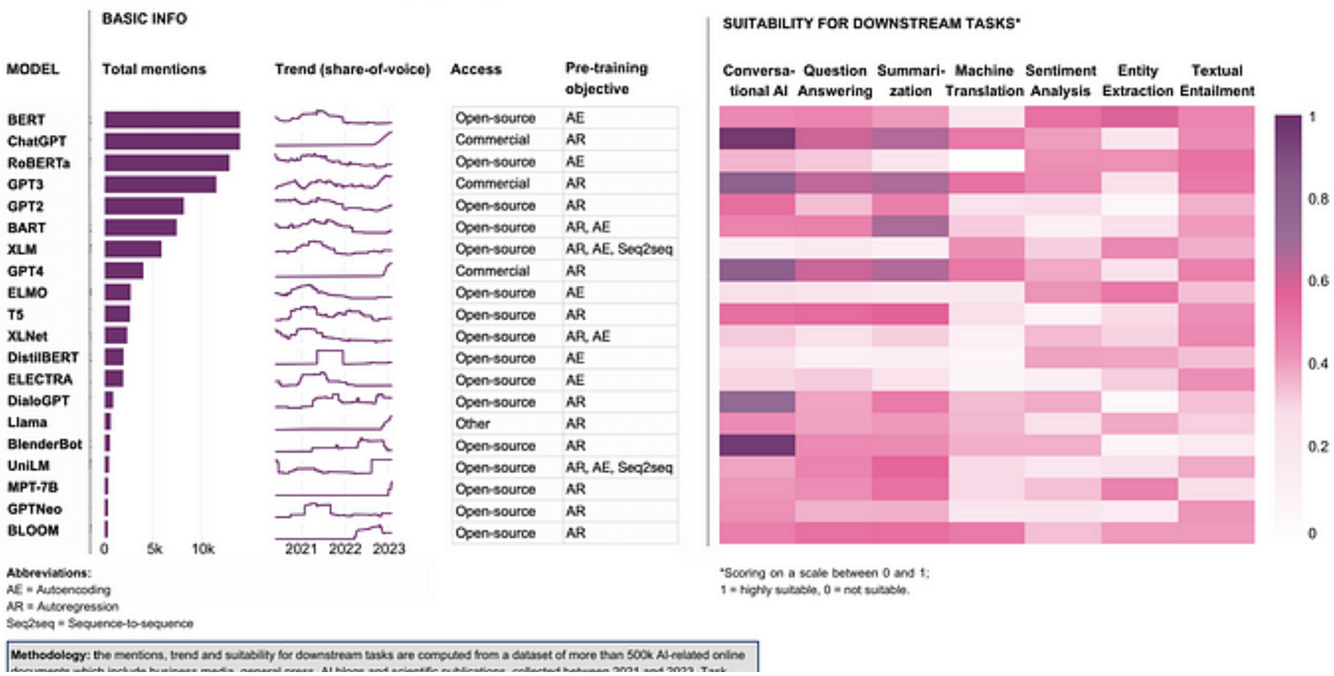
- Previous experience reported for this model-task combination (cf. Figure 5)

4. The short-listed models should be then tested against your real-world task and dataset to get a first feeling for the performance.
5. In most cases, you are likely to achieve a better quality with dedicated fine-tuning. However, consider few-/zero-shot-learning if you don't have the internal tech skills [illegible] for fine-tuning, or if you need to cover a large number of tasks.
6. [illegible] innovations and trends are short-lived. When using language models, keep an eye on their lifecycle and the overall activity in the LLM landscape and watch out for opportunities to step up your game.

Finally, be aware of the limitations of LLMs. While they have the amazing, human-like capacity to produce language, their overall cognitive power is galaxies away from us humans. The world knowledge and reasoning capacity of these models are strictly limited to the information they find at the surface of language. They also can't situate facts in time and might provide you with outdated information without blinking an eye. If you are building an application that relies on generating up-to-date or even original knowledge, consider combining your LLM with additional multimodal, structured or dynamic knowledge sources.

References

[1] Victor Sanh et al. 2021. Multitask prompted training enables zero-shot task generalization. CoRR, abs/2110.08207.
[2] Yoshua Bengio et al. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks,* 5(2):157–166.
[3] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation,* 9(8): 1735–1780.
[4] Kyunghyun Cho et al. 2014. On the properties of neural machine translation:



MODEL	Access	Pre-training objective
BERT	Open-source	AE
ChatGPT	Commercial	AR
RoBERTa	Open-source	AE
GPT3	Commercial	AR
GPT2	Open-source	AR
BART	Open-source	AR, AE
XLM	Open-source	AR, AE, Seq2seq
GPT4	Commercial	AR
ELMO	Open-source	AE
T5	Open-source	AR
XLNet	Open-source	AR, AE
DistilBERT	Open-source	AE
ELECTRA	Open-source	AE
DialoGPT	Open-source	AR
Llama	Other	AR
BlenderBot	Open-source	AR
UniLM	Open-source	AR, AE, Seq2seq
MPT-7B	Open-source	AR
GPTNeo	Open-source	AR
BLOOM	Open-source	AR

Abbreviations:
AE = Autoencoding
AR = Autoregression
Seq2seq = Sequence-to-sequence

*Scoring on a scale between 0 and 1;
1 = highly suitable, 0 = not suitable.

Methodology: the mentions, trend and suitability for downstream tasks are computed from a dataset of more than 500k AI-related online

for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota.

[11] Julien Simon. 2021. Large Language Models: A New Moore's Law?

[12] Underlying dataset: more than 320k articles on AI and NLP published 2018–2022 in specialised AI resources, technology blogs and publications by the leading AI think tanks.

All images unless otherwise noted are by the author.

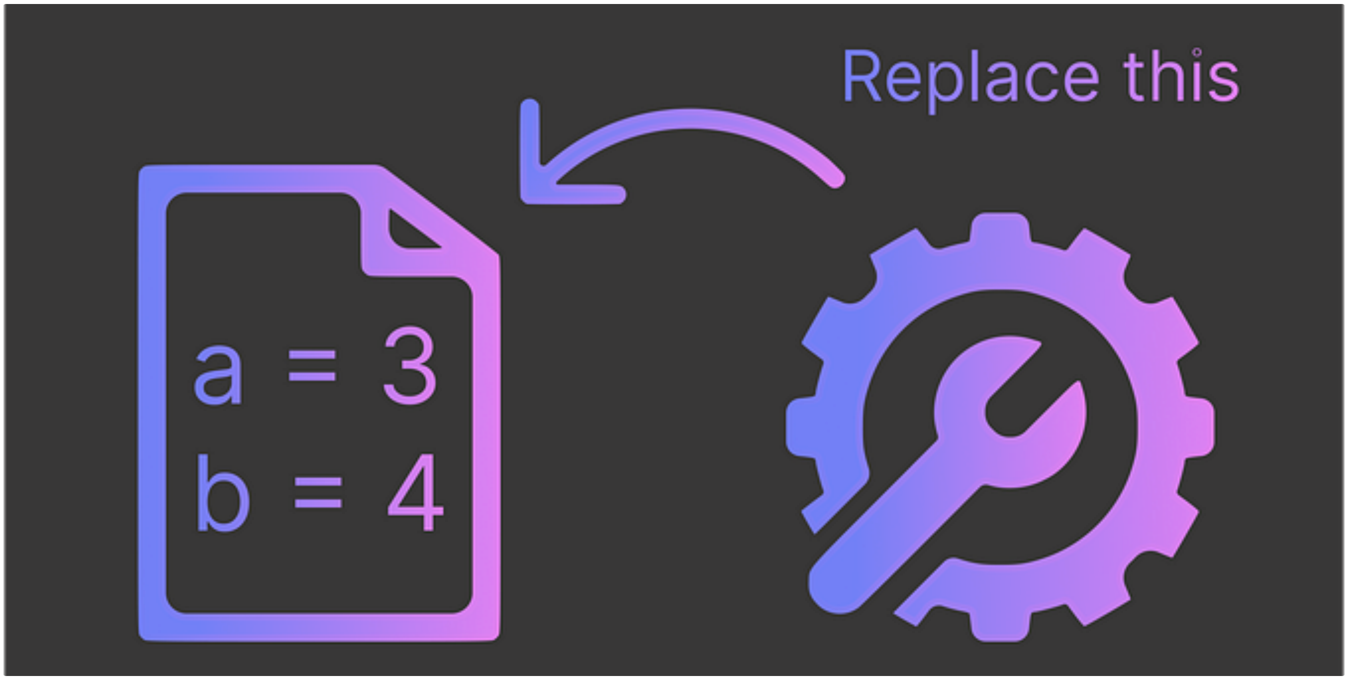
Janna Lipenkova in Towards Data Science

Four LLM trends since ChatGPT and their implications for AI builders

In October 2022, I published an article on LLM selection for specific NLP use cases , such as conversation, translation and summarisation...

15 min read · May 29

2K



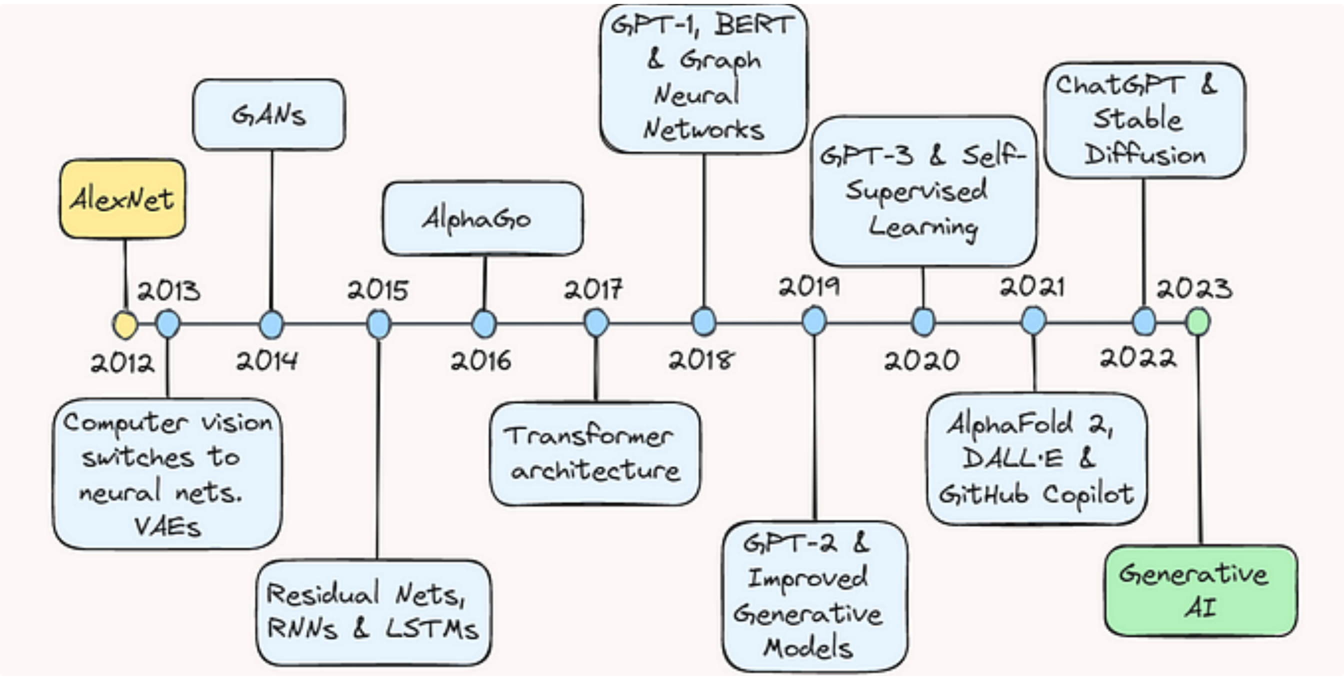
Khuyen Tran in Towards Data Science

Stop Hard Coding in a Data Science Project — Use Config Files Instead

And How to Efficiently Interact with Config Files in Python

· 6 min read · May 26

1.4K 19


AlexNet
GANs
AlphaGo
GPT-1, BERT & Graph Neural Networks
GPT-3 & Self-Supervised Learning
ChatGPT & Stable Diffusion
2013
2015
2017
2019
2021
2023
2012
2014
2016
2018
2020
2022
Computer vision switches to neural nets. VAEs
Residual Nets, RNNs & LSTMs
Transformer architecture
GPT-2 & Improved Generative Models
AlphaFold 2, DALL·E & GitHub Copilot
Generative AI

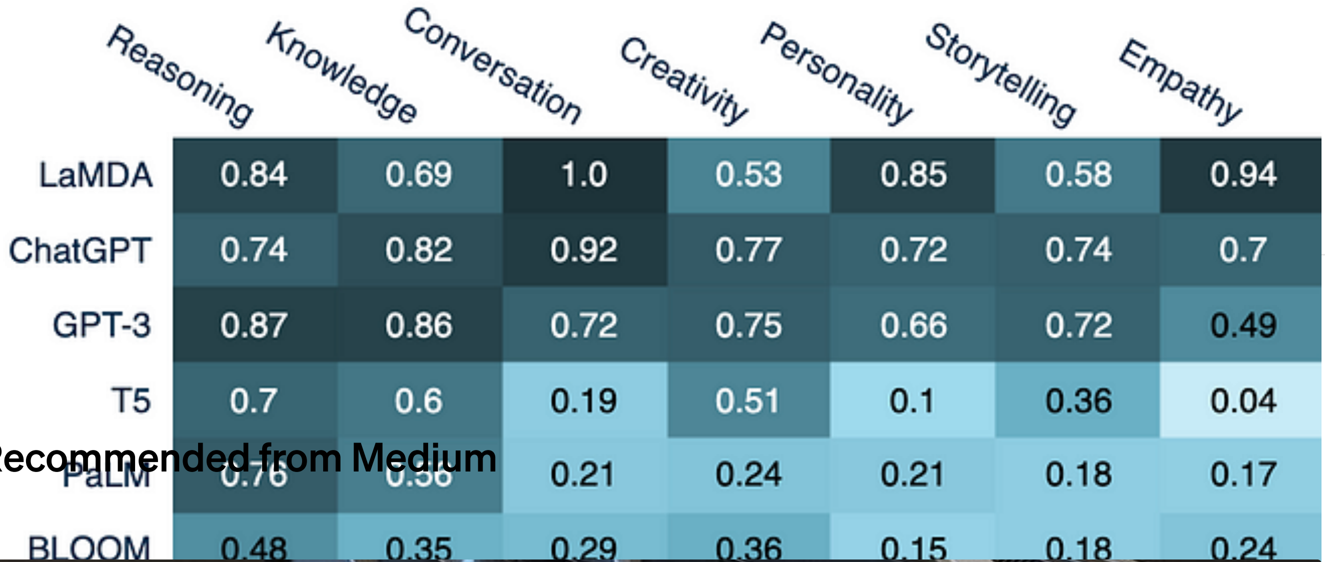
Thomas A Dorfer in Towards Data Science

Ten Years of AI in Review

From image classification to chatbot therapy

· 15 min read · May 23

1.3K 12



	Reasoning	Knowledge	Conversation	Creativity	Personality	Storytelling	Empathy
LaMDA	0.84	0.69	1.0	0.53	0.85	0.58	0.94
ChatGPT	0.74	0.82	0.92	0.77	0.72	0.74	0.7
GPT-3	0.87	0.86	0.72	0.75	0.66	0.72	0.49
T5	0.7	0.6	0.19	0.51	0.1	0.36	0.04
PaLM	0.76	0.56	0.21	0.24	0.21	0.18	0.17
BLOOM	0.48	0.35	0.29	0.36	0.15	0.18	0.24

Recommended from Medium

See all from Janna Lipenkova

See all from Towards Data Science

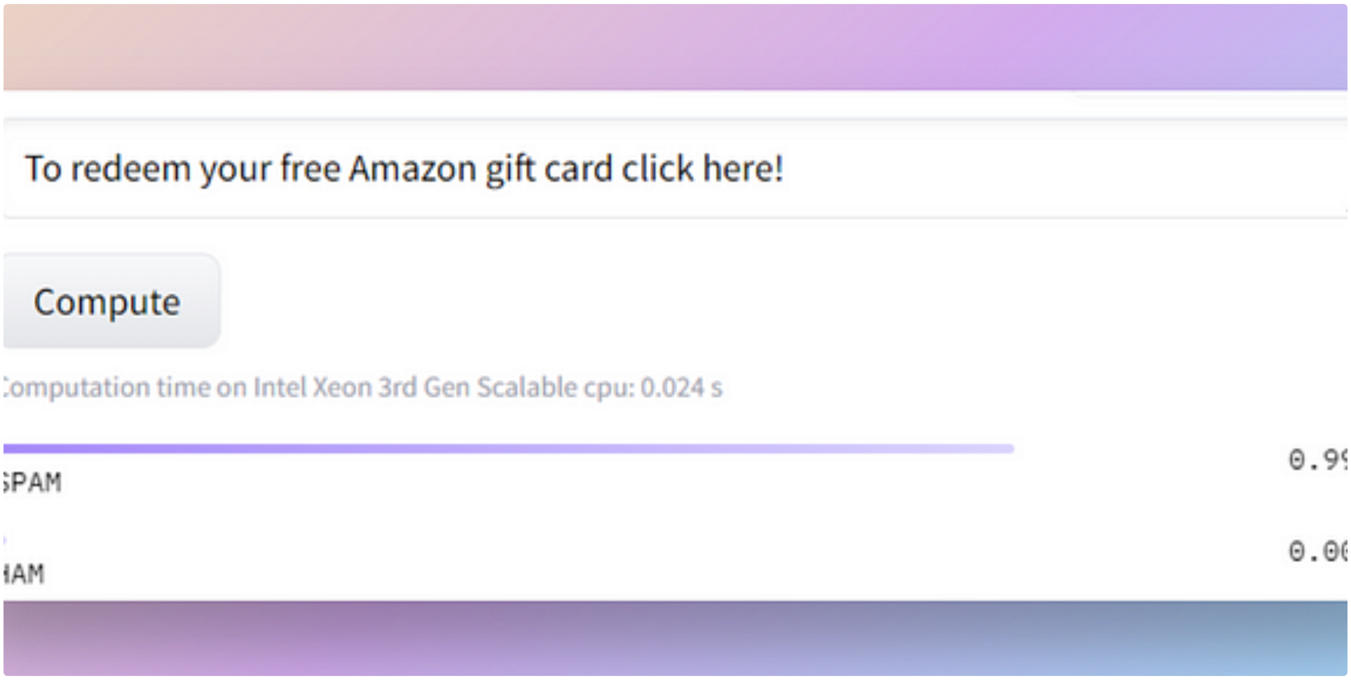
Benjamin Marie in Towards AI

Run Very Large Language Models on Your Computer

With PyTorch and Hugging Face's device_map

✦ · 5 min read · Dec 22, 2022

218 2



To redeem your free Amazon gift card click here!

Compute

omputation time on Intel Xeon 3rd Gen Scalable cpu: 0.024 s

SPAM 0.99

HAM 0.00

Skanda Vivek in Towards Data Science

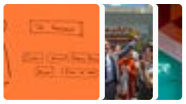
Transformer Models For Custom Text Classification Through Fine-Tuning

A tutorial on how to build a spam classifier (or any other classifier) by fine-tuning the DistilBERT model

✦ · 4 min read · Jan 20

142 1

Lists



Business 101

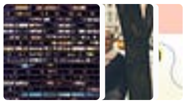
25 stories · 118 saves

Company Offsite Reading List

8 stories · 18 saves

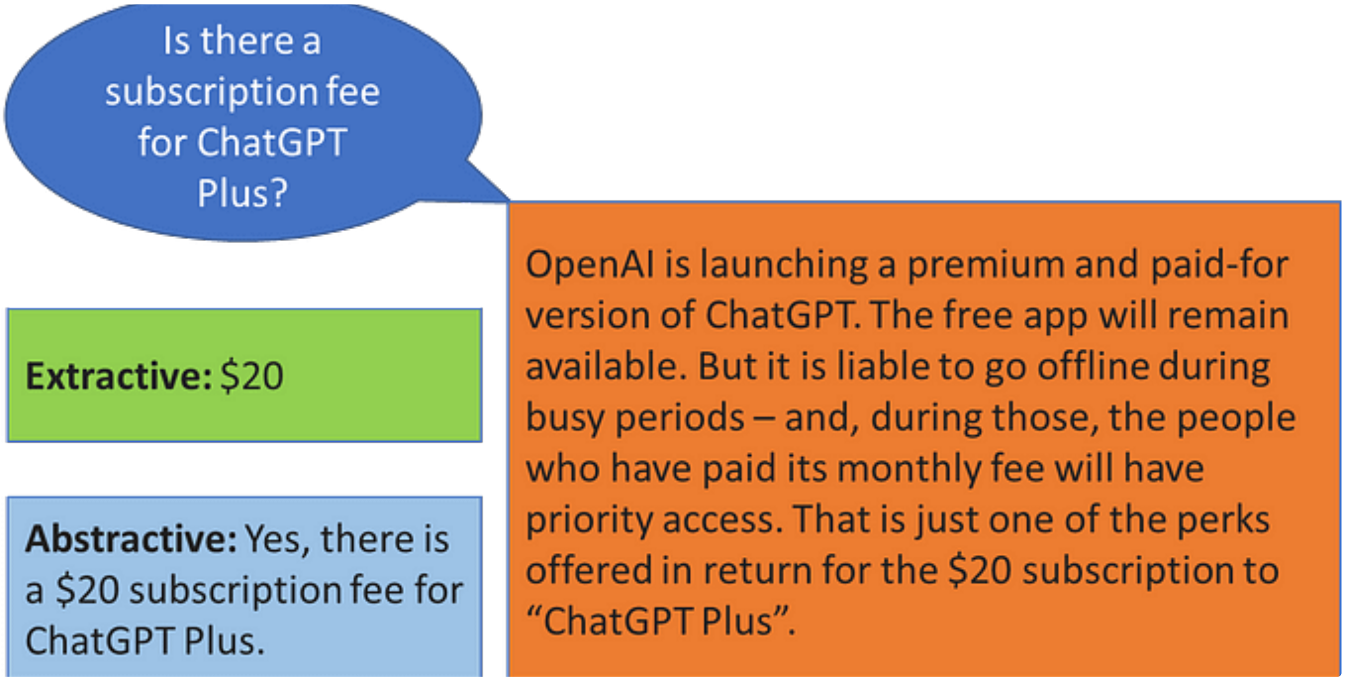
Leadership upgrades

7 stories · 9 saves



Intro to People Ops: Not Your Mama's HR

8 stories · 10 saves



Is there a subscription fee for ChatGPT Plus?

Extractive: $20

Abstractive: Yes, there is a $20 subscription fee for ChatGPT Plus.

OpenAI is launching a premium and paid-for version of ChatGPT. The free app will remain available. But it is liable to go offline during busy periods – and, during those, the people who have paid its monthly fee will have priority access. That is just one of the perks offered in return for the $20 subscription to "ChatGPT Plus".

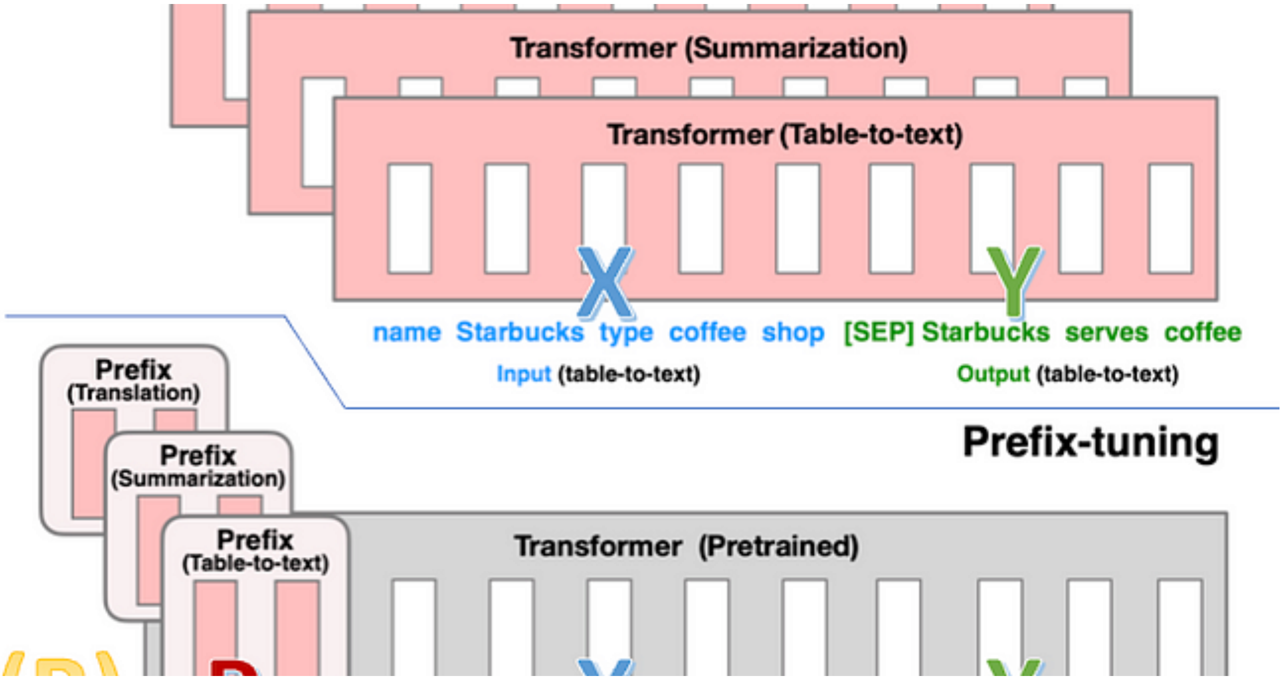
Skanda Vivek in Towards Data Science

Extractive vs Generative Q&A — Which is better for your business?

The arrival of ChatGPT hints at a new era of search engines, this tutorial dives into the 2 basic types of AI based question answering

✦ · 6 min read · Feb 6

57



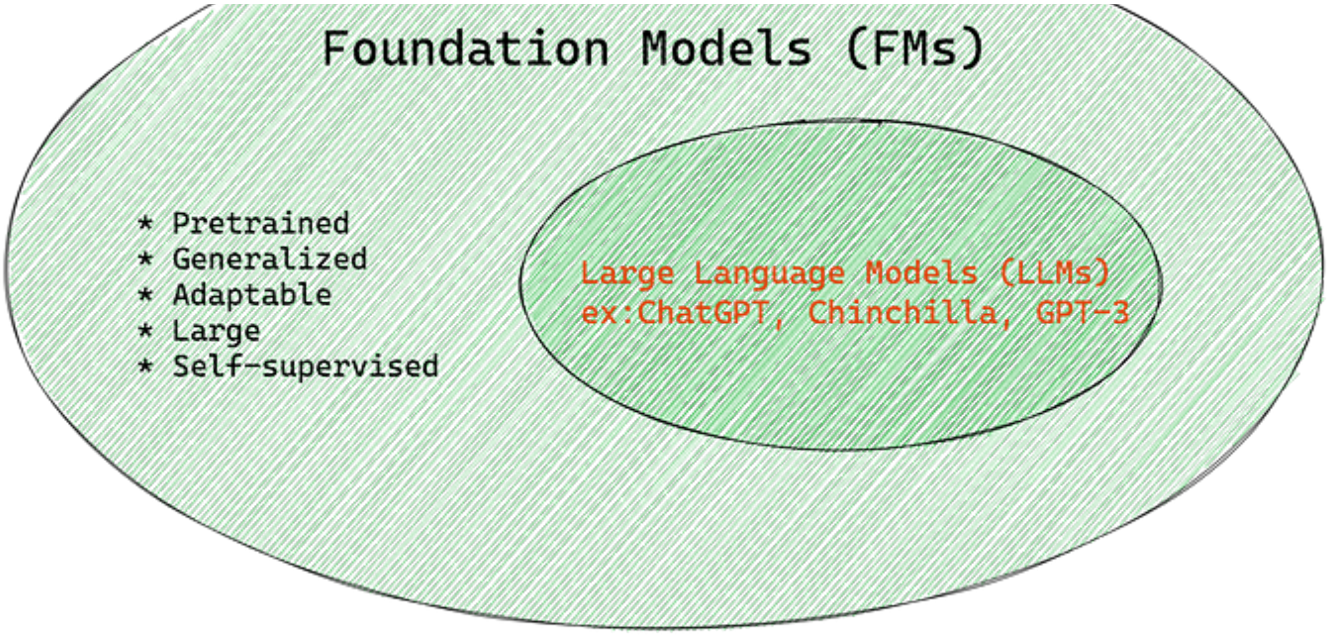
Chris Kuo/Dr. Dataman

Fine-tune a GPT — Prefix-tuning

I plan to walk you through the fine-tuning process for a Large Language Model (LLM) or a Generative Pre-trained Transformer (GPT). There...

· 17 min read · Jun 10

129 1



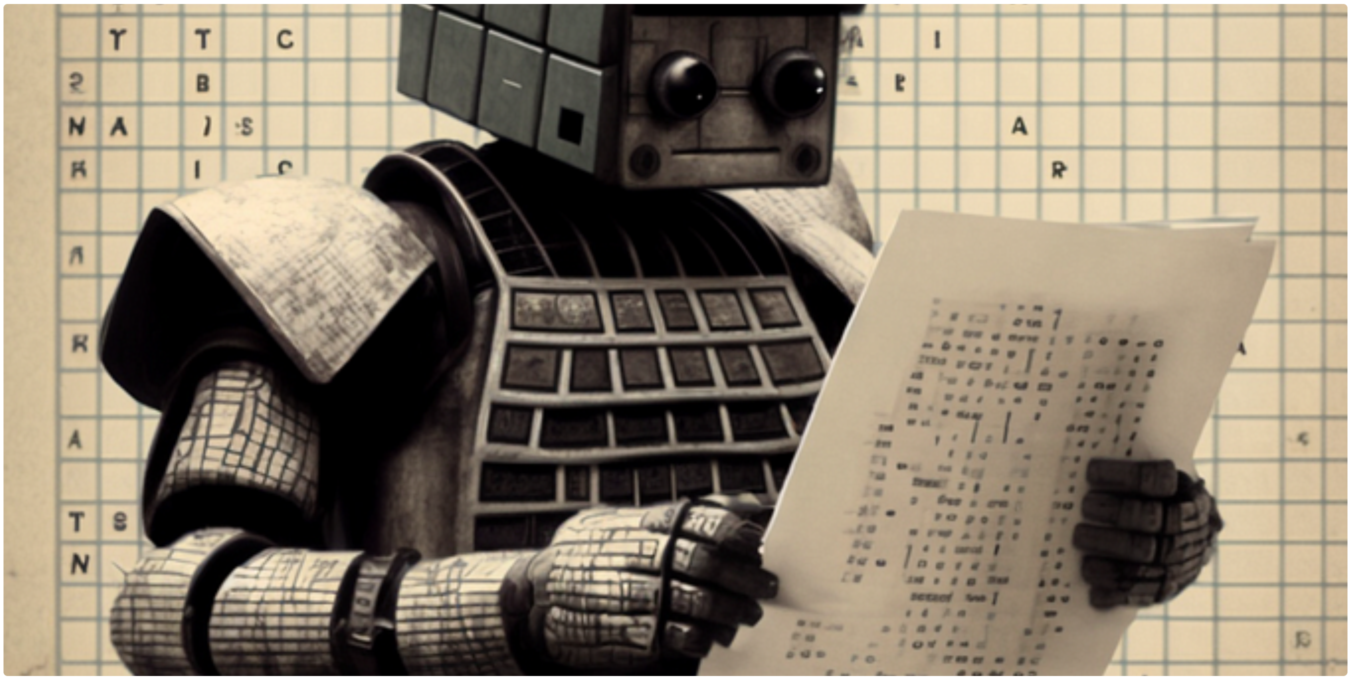
Babar M Bhatti

Essential Guide to Foundation Models and Large Language Models

The term Foundation Model (FM) was coined by Stanford researchers to introduce a new category of ML models. They defined FMs as models...

· 15 min read · Feb 6

216



The Jasper Whisperer

The Dummy Guide to ‘Perplexity’ and ‘Burstiness’ in AI-generated content

Understanding Language Models: A Simplified Guide

· 6 min read · Feb 17

181 3

See more recommendations