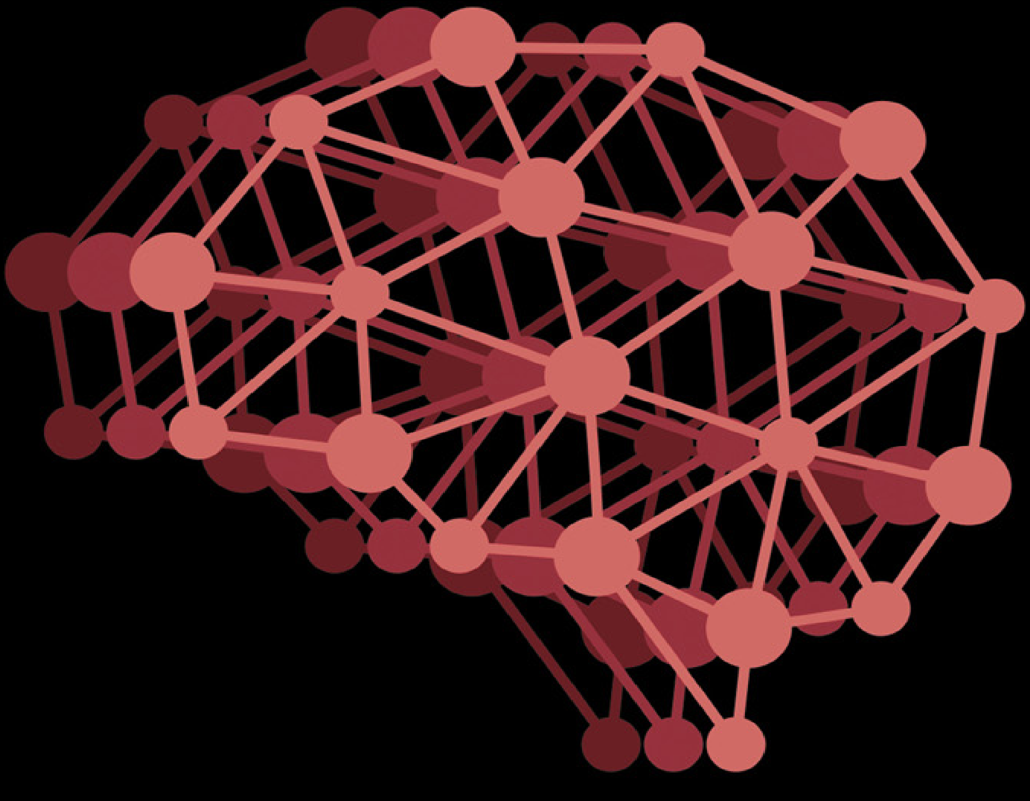
LARGE LANGUAGE MODELS

STEPHAN RAAIJMAKERS



THE MIT PRESS ESSENTIAL KNOWLEDGE SERIES

Large Language Models

Large Language Models

Stephan Raaijmakers

The MIT Press | Cambridge, Massachusetts | London, England

The MIT Press
Massachusetts Institute of Technology
77 Massachusetts Avenue,
Cambridge, MA 02139
mitpress.mit.edu

The MIT Press would like to thank the anonymous peer reviewers who provided comments on drafts of this book. The generous work of academic experts is essential for establishing the authority and quality of our publications. We acknowledge with gratitude the contributions of these otherwise uncredited readers.

This book was set in Chaparral Pro by New Best-set Typesetters Ltd.

Library of Congress Cataloging-in-Publication Data is available.

ISBN: 978-0-262-55269-1

10 9 8 7 6 5 4 3 2 1

To Zoë, Sanne, and Jasmijn.

Contents

Series Foreword

The MIT Press Essential Knowledge series offers accessible, concise, beautifully produced pocket-size books on topics of current interest. Written by leading thinkers, the books in this series deliver expert overviews of subjects that range from the cultural and the historical to the scientific and the technical.

In today’s era of instant information gratification, we have ready access to opinions, rationalizations, and superficial descriptions. Much harder to come by is the foundational knowledge that informs a principled understanding of the world. Essential Knowledge books fill that need. Synthesizing specialized subject matter for nonspecialists and engaging critical topics through fundamentals, each of these compact volumes offers readers a point of access to complex ideas.

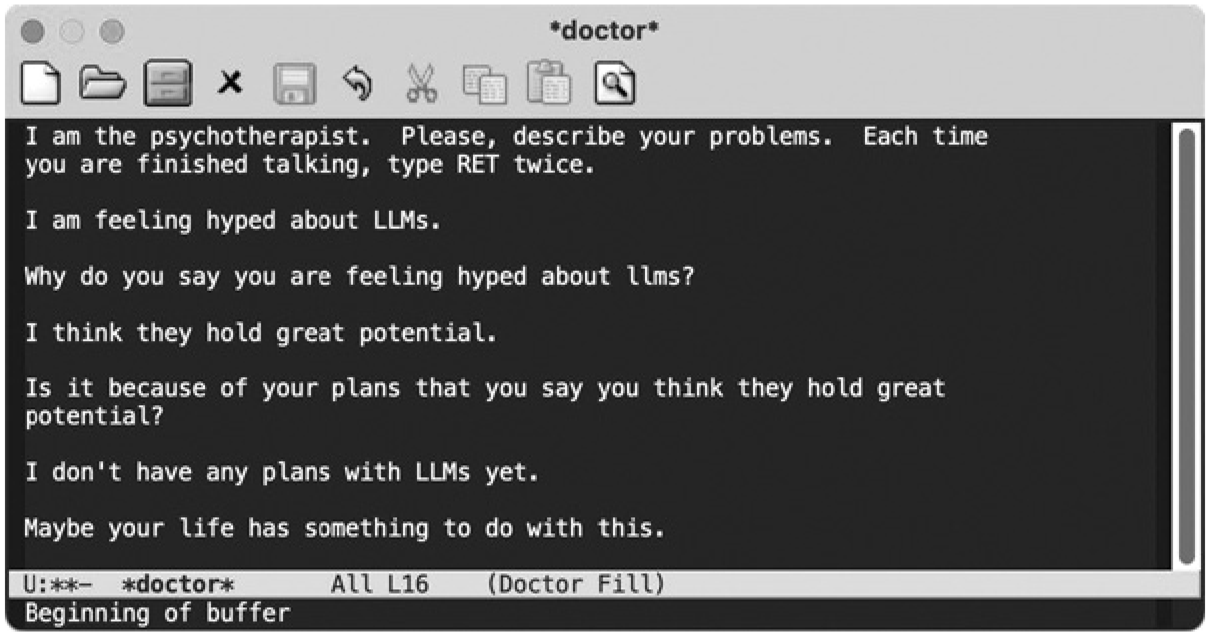
1

Large Language Models

In 1953, the famous Austrian philosopher Ludwig Wittgenstein wrote in his *Philosophical Investigations*: “The meaning of a word is its use in the language.”[1] The US linguist Rupert Firth rephrased this a bit more colloquially in 1957 as “you shall know a word by the company it keeps.”[2] Little did these two men know that roughly seventy years later, their casually appearing statements would underlie highly complex algorithms affecting millions of people on earth: *language models*.

Language models are everywhere in our modern digital life. From typing in a query in Google or Bing, instructing Siri to set a timer when you’re boiling an egg, asking Alexa to switch on a light, and using the autocompletion in Whatsapp messages, to conversations with a chatbot: language models get the job done. Language models predict the surfacing of a particular word given its neighboring ones. They draw their knowledge about words from the statistical analysis of vast amounts of textual data. When these models get big (and we will discuss shortly what “big” means), we call them *large language models* (LLMs). Surprisingly, these models can do a lot more than just generating words. They can even have conversations with us humans.

We have come a long way. In the 1960’s, MIT computer scientist Joseph Weizenbaum created what was probably the first chatbot ever: ELIZA, a primitive, parrotlike psychotherapist completely programmed by handwritten rules that were triggered by specific keywords in the sentences typed in by human “clients.” Currently, ELIZA still lives on as a macro in the code editor Emacs (figure 1).



```
I am the psychotherapist.  Please, describe your problems.  Each time
you are finished talking, type RET twice.

I am feeling hyped about LLMs.

Why do you say you are feeling hyped about llms?

I think they hold great potential.

Is it because of your plans that you say you think they hold great
potential?

I don't have any plans with LLMs yet.

Maybe your life has something to do with this.

U:**-  *doctor*       All L16     (Doctor Fill)
Beginning of buffer
```

Figure 1 Eliza still lives on in the Emacs editor.

The author of this book was once involved as a student in the development of the Dutch version of another landmark chatbot: Q&A, short for question and answer, by Symantec.[3] This system from the 1990s, premised on finicky, rule-based grammar and largely programmed in the illustrious LISP language, was able to support users in drafting reports based on database information. It could answer queries like, “Please provide all names and addresses of employees who earn more than $80,000 and live in Amsterdam.” Following these early natural language interfaces, we have seen IBM Watson, Microsoft’s Twitter adaptive, self-learning chatbot Tay, and a whole generation of scripted chatbots. You must have come across many of these in online customer services environments. As convincing as these systems sometimes looked, they were seriously hampered by their limited grasp of natural language, general inability to learn from data, and limited set of communicative skills. None of these systems could write a poem, make a joke, or engage in a natural dialogue. And many of them were not based on LLMs.

On November 30, 2022, ChatGPT was launched—an interactive LLM with a chat interface produced by OpenAI. ChatGPT generates texts through dialogues with a user. It is the logical successor to a set of precursor models, also from OpenAI: the GPT model family, comprising the members GPT-1 and GPT-2. These models were produced by a type of neural network architectures called *Transformers* (with GPT meaning *Generative Pretrained Transformer*). While the older GPT models were impressive by themselves and had already shown impressive language generation capabilities, the 2022 version of ChatGPT was something else. This system is based on the updated model GPT-3.5 (followed a few months later by GPT-4). ChatGPT is equipped with dialogue management and “chat” facilities, including mechanisms for building up conversational histories. These mechanisms allow humans to collaborate interactively, through dialogues, with the system on a text. In addition to that, ChatGPT contains several other techniques for text production. For instance, you can have ChatGPT generate a poem in the style of John Keats about a specified topic and then apply a more Shakespearean style, similar to *style transfer* in images generated by artificial intelligence (AI) (like applying Vincent van Gogh’s style to the *Mona Lisa*). But there is more. You can also have ChatGPT generate working computer codes from natural language, solve mathematical puzzles, draw analogies between concepts, and generate explanations. Some of these skills, as we will see, are not among the core abilities of language models; they seem to manifest themselves as a result of unknown factors.

ChatGPT reached over 100 million users in three months, and caused quite a stir in society, education, and science. Its eloquence, authoritative manner, and sometimes eerily natural responses have led certain people to fret that the moment of *singularity* has come closer, with AI becoming humanlike or even *sentient*. The LLM of the 2022 release of ChatGPT was trained on a static (and huge) set of texts (totaling 300 billion—that is, 300,000 million—words). These data consist of web information, books, and a variety of other textual resources. ChatGPT has been carefully fine-tuned afterward by humans to provide appropriate responses. On top of all of that, filters weeding out toxic intents from users have been implemented (preventing, for example, the generation of jokes about politicians or recipes for making bombs).

Concerns on the societal consequences of ChatGPT and anecdotes on derailing dialogues keep appearing in the media. In higher education, the availability of ChatGPT has generated a lot of worries. Students can use ChatGPT to generate well-formulated texts on virtually any topic, with different styles. These texts are oftentimes indistinguishable from human-produced ones, making ChatGPT a risk factor for plagiarism and fraud. Language models in general are “paraphrase”-oriented; they generate fluent text on certain topics, usually without disclosing their sources. This makes them hard to trust from a factual perspective. Furthermore, they can be biased by their data (“selection bias”), architecture (“algorithm bias”), and the humans who train or fine-tune them (“training bias”). But such biases cannot always be easily detected or even prevented.

Language models in general are “paraphrase”-oriented; they generate

> fluent text on certain topics, usually without disclosing their sources.

For actual users of these models like you and me, these issues should not be purely academic. LLMs are becoming increasingly embedded in the commercial software we use daily, like word processors, mail clients, and conversational services on the web. This should instill a healthy dose of awareness in us, based on a sufficient understanding of how these models are organized and what their strengths and weaknesses are. This book aims to help you realize that goal. Before we take another look at these—and other—controversies, let's discuss what a language model actually is.

Imagine you have enrolled in an Italian for Beginners course. Apart from the dreaded grammar drills and cringe of getting those perky pronunciations right, your teacher may confront you with a so-called Cloze test, a popular test for second-language learners.[4] In this case, the test consists of showing you Italian sentences, with one or two words blanked out. You, as a student, are asked to fill in the blanks.

- *I fiori* ___ *vaso sono gialli* ("the flowers in the vase are yellow"). Fill in: *nel*, "in the."
- ___ *angeli* ___ *dipinto erano paffuti* ("the angels on the painting were puffy"). Fill in: *sul*, "on," and *gli*, "the."

When you perform this test, you mimic a language model. Based on your—still rudimentary—knowledge of Italian and the cues provided by the context, you may be able to fill in the correct words for the blanks. A language model does something similar. It digests huge amounts of words and infers conditional or contextual probabilities from these data, such as the probability that *vaso* most likely is preceded by *nel* in the context of *I fiori*. Once it has been exposed to data and all the relevant probabilities have been computed, we can put language models in *generation mode* and have them generate language. Or we can measure their *perplexity* or surprise when confronted with a piece of language they would not expect. This comes in handy when we would like to recognize different languages—an English-language model, for instance, would be surprised to see Italian text. Perplexity can also be used to ascribe certain texts to an author.

Language models come in different architectures. Traditional language models build up explicit statistics using *conditional probabilities*: the probability that a word follows another word or a sequence of words. We call these models *statistical language* models.

An example of such a language model is described by the following, somewhat approximate formula:

$$p(x_1,\ldots,x_n) = \prod_{i=1}^{n} p(x_i \mid x_{<i})$$

This formula expresses the following: the joint probability $p(x_1, \ldots ,x_n)$ of the sequence of words $x_1, \ldots ,x_n$ (that is, the chance of observing these words together, as a sequence) is a multiplication (or product, expressed by the large Greek *pi* symbol) of so-called conditional probabilities, which is the probabilities of observing word x_i given an unspecified quantity of words that precede x_i, denoted informally with $x<_i$. In other words, for this model, the probability of seeing a certain word in a word sequence depends on the words that precede that word—its left context. Many options are available here. You can express conditional probabilities referring to any type of context you like: words preceding, words following, combinations, and restrictions on the amount of words that play a part in such contexts. But how are these probabilities computed in the first place? According to Bayesian statistics, conditional word probabilities can be computed from simple counts of single word frequencies and joint occurrences. That is, a conditional probability $p(word_1 \mid word_2)$ can be computed by counting all joint occurrences of $word_1$ and $word_2$ and dividing that quantity by the number of times we see $word_2$:

$p(cat \mid the)$ = number of times we observe "the + cat" divided by the times we see "the"

Suppose we would like to compute the probability of observing a sentence like "The cat slept." Assuming a beginning-of-sentence marker like "<*s*>," and a similar end-of-sentence marker "</*s*>," language models would compute this probability as a multiplication of probabilities:

$$P(The \mid <s>)P(cat \mid <s>, The)P(slept \mid The, cat)\ P(</s> \mid cat, slept)$$

Modern language models are created by *deep learning*: the subfield of AI that learns from data with complex neural networks. Neural networks are biology-inspired methods for machine learning. They typically carve up data into small pieces and distribute these across many computing units called *neurons*. Neurons receive signals (data), manipulate their incoming signals in a mathematical way, and send out their results through weighted connections to other neurons, arranged in layers. Neural networks learn from (usually human-annotated—that is, preanalyzed) data and estimate their weights accordingly. Figure 2 shows such a network.

Input neurons send their data through weighted connections to a hidden layer of neurons, which sends its output to an output layer. Such an output layer can express the labeling of the input data. All weights are learned from data. Complex neural architectures have been designed for producing language models from raw textual data, and the way they compute their statistics is determined by intricate computations that go way beyond simple counting. The resulting models are called *neural language models*.

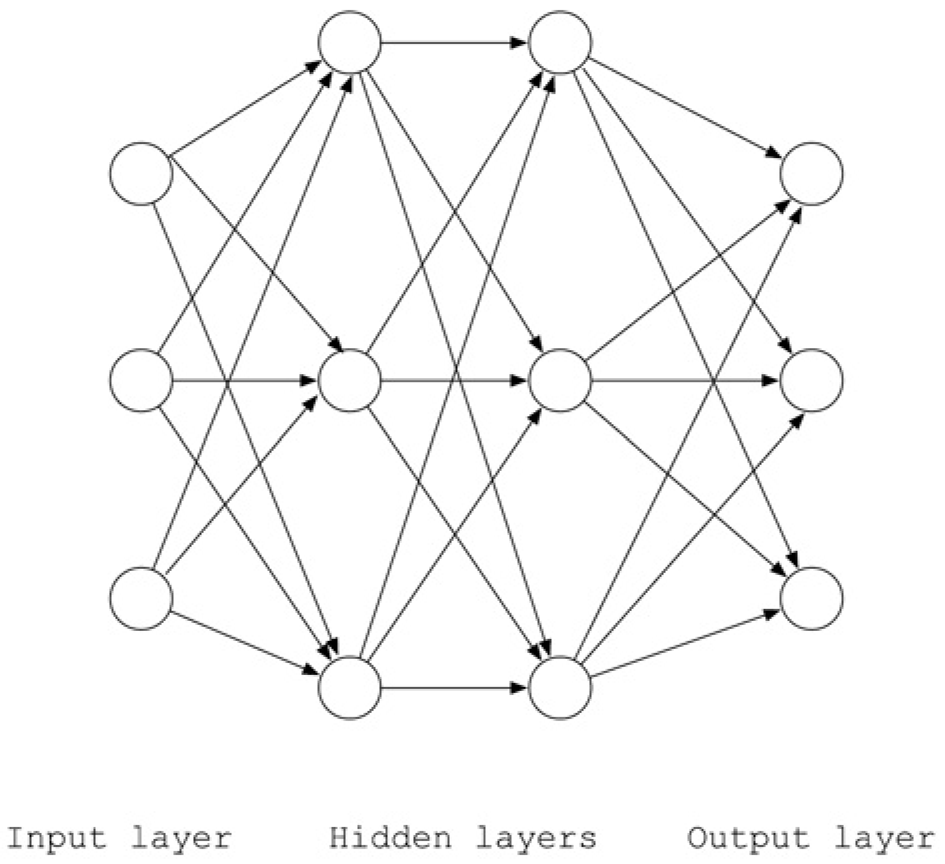


Figure 2 A simple neural network.

A neural language model looking at left context computes approximately the following probabilities:

$$p(x_1,\ldots,x_n) = \prod_{i=1}^{n} p_\theta(x_i \mid x_{<i})$$

This is the same formula as the one above, with a slight difference: the introduction of a theta (θ) symbol. This symbol expresses that the conditional word probabilities now depend not just on the word context but also the extra information: ingredients from a neural network. We will discover in chapter 3 what those ingredients are.

In this book, when we talk about LLMs, we will typically mean *neural* LLMs. At this point, we need to set some terminology straight before we proceed. LLMs can, as we will see, be trained in several ways. In fact, they go through an entire curriculum, starting by doing a (tremendous) amount of reading and then becoming trained for obeying human instructions. Once done, some of them have reached the gold medal status of *AI assistant*: they can assist us humans in doing our work. The obvious examples are ChatGPT, Copilot, and Gemini.[5] But essentially these are still LLMs. And the ones that do not complete the entire curriculum are also just LLMs. We will make the distinction clear as we go along. For now, just think of *LLM* as an umbrella term.

Now, what makes a language model *large*? LLMs have basically three dimensions that determine their size. First, the number of words a model has been exposed to for computing its probabilities ("data size"). Secondly, there is the number of parameters—the neural weights—of the neural network that produces the model ("parameter size"). These parameters determine the complexity of the neural network. The third dimension is the amount of computing power needed to compute these weights from data ("computing size"), specified in terms of GPU FLOPS: the number of floating-point operations per second run by a graphical processing unit, a processor on a graphical card that is geared toward the complex computations deep learning models tend to carry out (figure 3).

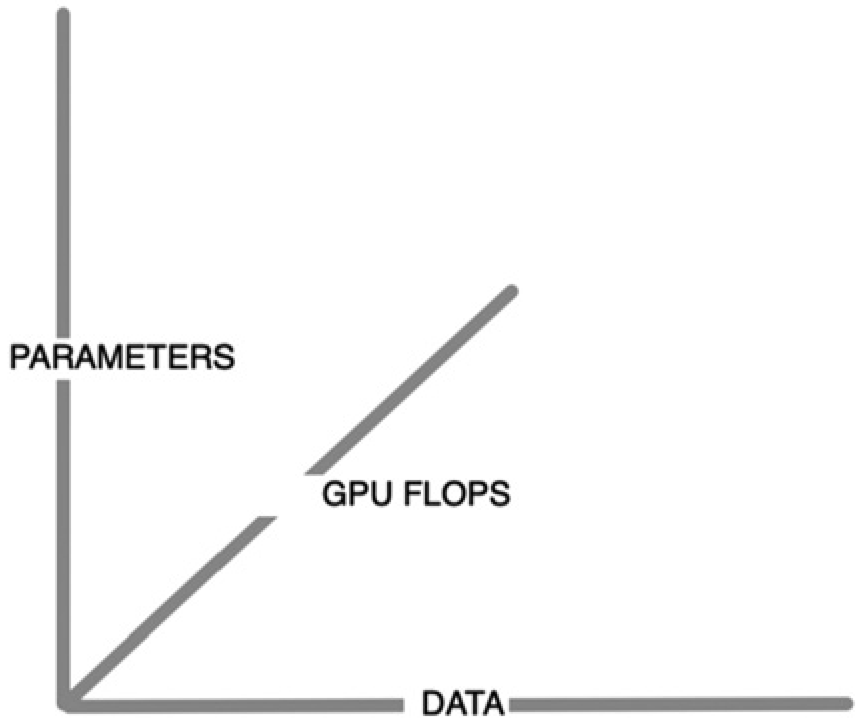


Figure 3 The three dimensions that determine the size of neural language models: parameter, data, and computing sizes (GPU FLOPS).

The first two dimensions are the ones usually reported in scientific literature for models. As an example, ChatGPT, based on GPT-3.5 (the name of the underlying LLM), has 175 billion parameters, and, as mentioned, was trained on 300 billion words.

Is a large LLM always better than a smaller LLM? Not necessarily, at least not across all three dimensions. In fact, many LLMs have disproportionately grown in their parameter size but not in their data size. Nowadays, we know that such underpopulated models are indeed undertrained. They could do a lot better with a better alignment of their parameter size to their data size. And they can often be shrunk for parameter size as well without major detrimental consequences. The exact balance between parameter, data, and computing sizes is an acute topic on the scientific research agenda, with the potential benefit of smaller, more optimally balanced models rivaling or even outperforming larger models. Having smaller models do the same job with fewer resources (time, money, and energy) is of course desirable!

In February 2023, *New York Times* tech journalist Kevin Roose published his eerie experiences with ChatGPT, which, at that time, had been linked to the Bing search engine by Microsoft, a large investor in OpenAI.[6] As one of the beta testers of this combination, Roose had an intense question-and-answer session with Bing. After some interactions, Bing —through ChatGPT—revealed that its real name was Sydney (which in fact was its prototype name given by its engineers), and that it fully trusted the Bing team with keeping it safe and sound. After this, Bing (or Sydney) engaged in a grotesque dialogue about having a dark "shadow self" (prompted by Roose), eventually leading to the disclosure of evil actions such a shadow self would be likely to perform, like overruling humankind or spreading misinformation. Finally, the system got snarky about Roose's marriage and repeatedly declared its love to Roose, even when Roose frantically tried to lure the model away from this topic.

Anecdotes like these have sparked genuine concerns about LLM-powered chatbots. And they are by no means limited to the latest flurry of models. Back in 2016, we witnessed a chatbot gone rogue: the aforementioned Twitter chatbot Tay, by Microsoft. Over the course of one night, this chatbot, which started out as a friendly conversational agent with its own Twitter account, turned into a misogynistic, antisemitic, and racist entity, apparently by being able to learn from interactions with malevolent human users. In 2022, Blenderbot by Meta produced fake news by claiming that Joe Biden lost the US presidential elections in 2020 and that Donald Trump was currently serving his second term. Similarly, Galactica, an LLM-based chatbot for assisting scientists that was launched in November 2022 by Meta, produced downright wrong or biased data. The public demo was taken offline after a mere three days. Such stories raise several urgent questions.

Given the fact that LLMs are becoming increasingly prevalent in our digital lives, can we estimate the exact capabilities of such models beforehand, so that we know when to rely on them and when not to? This question has a complicated answer. LLMs are to some extent stochastic systems and perform a certain amount of random behavior by design. Further, as mentioned before, LLMs display a hitherto poorly understood capability of doing things they were not explicitly trained to do. This is dubbed *emergence*: the appearance of a *quality* (like a certain trait) above a certain *quantity* (defined in terms of LLM size). It seems the set of such emergent properties is by no means complete and fleshed out, as some of these emergent properties lay dormant in the sense that they can be triggered by showing the model a few relevant examples. How should we deal with these unexpected capabilities in real-world scenarios, and are they truly emergent or just *mirages*, that is, rather gradually manifesting capabilities? Are we looking at these properties through the right glasses? Moreover, can LLMs be creative thinkers, discovering new knowledge? Or do they merely juggle existing knowledge, and is their creativity limited to new combinations of old stuff?

LLMs are usually built from fixed snapshots of data. In early 2023, ChatGPT had access to data only up to the year 2022. Data in an LLM are typically not manually inspected and curated; the amount of data is just too vast for that. This means that the actuality of LLMs is determined by the data that went into them. Further, technically, it is not trivial to force accurate facts (like database information) to become part of generated text. To make things worse, LLMs have been shown to dream up unexpected text. This phenomenon also extends to question-and-answer scenarios. How do we tie LLMs to the facts and have them provide us insight into their underlying sources?

Here is another question. How can LLM-produced output be identified as being synthetic? This is a valid question in situations where originality and authorship matter (like in education or creative writing). While attempts to insert “digital watermarks” into generated texts are underway (such as certain specific and unlikely word combinations, revealing the hand of a language model), it is currently unclear whether these facilities will result in sound and complete solutions.

Other questions address the *governance* of LLMs and *AI sovereignty*. Currently, only the big tech industry has the data as well as the financial and computational capacities to develop, train, and maintain these models at scale. But these processes are often shrouded in mystery. Models are built from often undisclosed data collections, hidden choices for model size and model architecture design, and undocumented usage of human effort for fine-tuning models on human preferences. In addition, LLMs are subjected to proprietary, company-defined ethical rules and principles, which frequently are not made public either. Where does that leave users of these models, with potentially completely different ethical or legal paradigms? Can the vox populi be factored into the production process of LLMs? How can we inspect, evaluate, influence, and govern the models that big tech hands us?

This book will attempt to guide you through the historical and latest technological developments in this fast-paced field, and to provide answers to the questions above. It will discuss the current situation around LLMs at the time of this writing: a disruptive manifestation of humanlike AI, causing both concern and fascination in many audiences across the globe, and with potential important ramifications for society.

We will delve into the technical inner workings of LLMs, study their allegedly emergent abilities up close, and put things into perspective: Under which circumstances and conditions should we feel safe as well as entitled to use these models in our daily life?

2

Language as a Computational Phenomenon

Some people label themselves as “language people” rather than “math people.” After all, what would poetry, the new novel by Zadie Smith, the imperfect language of small children, and your eloquent social media posts have to do with computations on numbers and probabilities? As it turns out, quite a lot! In this chapter, we will visit a few important historical computational approaches to language analysis: strands in the field of *computational linguistics*. This will allow us to position LLMs amid the many other computational approaches to analyzing language. Such background is beneficial for understanding the roots of LLMs, but you can scoot off to chapter 3 if you are eager to learn more about their internal workings.

Language comes to us in sequences, like the consecutive sounds in a speech signal or the text you are reading now, word by word. Linguistic structure helps humans to interpret such utterances. For instance, in most languages with limited inflection, like English or Dutch, words do not just appear in random order. Linguists call these languages *configurational*; syntactic structure plays a large role in determining which word orders are allowed or not. In English, for example, the sentence “The boy bites the dog” cannot mean that the dog bites the boy, contrary to “The boy the dog bites.” Syntactic structure determines that objects of transitive verbs like “to bite” follow their verb in certain word orders. Languages that exhibit more liberal word order, however, tend to have richer inflection for compensation. For one thing, such rich inflection helps determine grammatical roles like subject or object. As an example, in the Australian language Wambaya, the following six permutations of the same sentence are allowed, and there is rich inflection helping to determine the meaning of the construction:[1]

1. Dawu **gin-a** alaji janyi-ni
   bite 3rd-sg-past boy-acc dog-erg
   the dog bit the boy
2. Dawu **gin-a** janyini alaji
3. Alaji **gin-a** dawu janyi-ni
4. Alaji **gin-a** janyi-ni dawu
5. Janyini **gin-a** dawu alaji
6. Janyini **gin-a** alaji dawu

Here, the *gin-a* is a past tense indicator, connected to the main verb *dawu* (to bite), of which the subject “dog” is marked with ergative case, uniquely identifying it as the agent of the action “to bite.” Notice that despite this liberal word order, *gin-a* needs to stay in second position everywhere.

Word choice is by no means a random process either. As mentioned in chapter 1, linguists like Firth have proposed the adage, “You shall know a word by the company it keeps.” This can be interpreted as a statistical statement: The probability of seeing a word surface in a text (or hearing a word in spoken language) depends on its context, and knowledge of the many contexts a word can (or cannot) appear in determines your knowledge about that word.

Under this statistical view of the human language facility, we humans develop, for every language we use, a *language* *model* that allows us to predict words in context, helping us to understand and produce language. Any such language model is basically a statistical function that computes a plausible completion of an incomplete utterance (a *context*), based on observations of similar contexts and their completions. Remember the Cloze test from chapter 1. This test addresses such an internal model and is quite useful as an educational tool for measuring the lexical capabilities of second-language learners. This all seems to perfectly make sense, and we are tempted to view a language model as a software program running on our brain’s hardware, just like the programs we use to process images and sounds. But thinking about language as something that can be subjected to computation did not occur overnight. How exactly, then, did language meet up with mathematics, statistics, and computation? It appears there are—at least—four pathways at play that treat language as a computational phenomenon, some of which can be linked to LLMs. We will embark on a little journey that will lead us through these pathways:

1. The generative school of linguistics founded by famed Pennsylvanian linguist Zelig Harris in the 1950s and continued by linguist Noam Chomsky.
2. The application of mathematical logic to language by philosopher Kazimierz Ajdukiewicz in the 1930s and notably mathematician Joachim Lambek in the 1960s, leading to the paradigm of categorial grammar.
3. The statistical approaches to language analysis based on mathematical work by mathematician Andrey Markov and engineer Andrew Viterbi.
4. Approaches to language analysis based on machine learning (which are essentially statistical by nature).

Generative Linguistics

In 1955, Chomsky, a former student of Harris's who was at the time working at MIT on a machine translation project, published his dissertation "Transformational Analysis." This work was premised on the idea that observed linguistic variations in a language can be *generated* from a small set of configurations, typically laid out in a grammar, and a set of *transformations* that generate controlled variations of these base structures. This makes a grammar and its associated transformations basically a computational device, as Chomsky later specified: "The fundamental aim in the linguistic analysis of a language L is to separate the grammatical sequences which are the sentences of L from the ungrammatical sequences which are not sentences of L. The grammar of L will thus be a device that generates all of the grammatical sequences of L and none of the ungrammatical ones."[2]

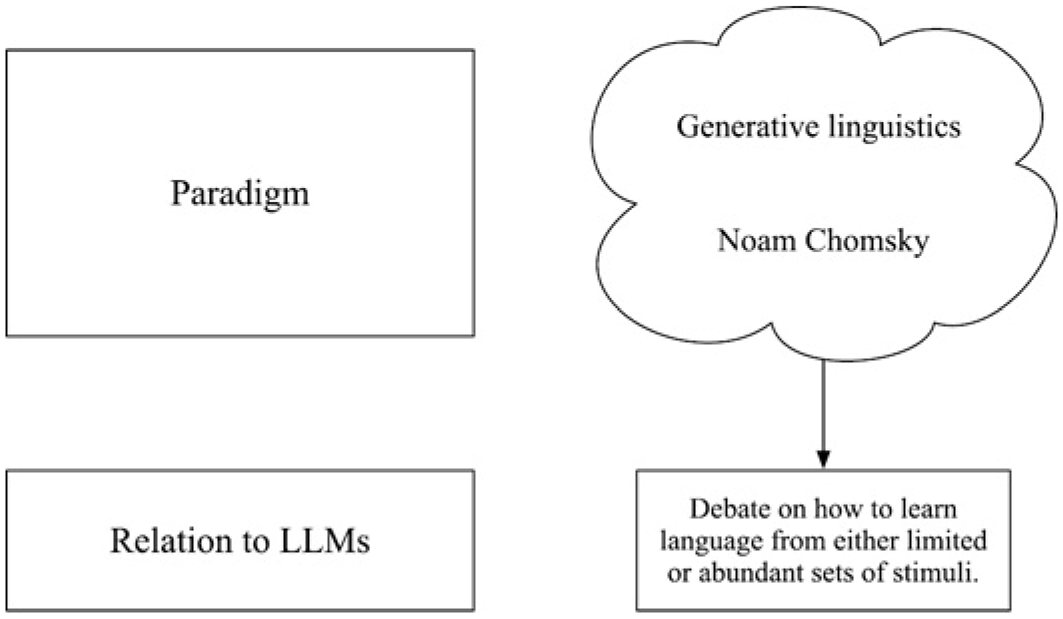

Paradigm
Generative linguistics
Noam Chomsky
Relation to LLMs
Debate on how to learn language from either limited or abundant sets of stimuli.

Figure 4

The influential linguistic paradigm of generative linguistics that Chomsky built up in the following decades (see figure

4) makes a number of nontrivial claims about the linguistic and cognitive capabilities of the human brain. They include a plea for a universal grammar that is parameterized for every language and obeys universal principles, and a *poverty of the stimulus* hypothesis.

According to this hypothesis, children acquire their native language based on an innate, parameterizable mechanism for language acquisition, which is set to its correct instantiation for a certain language based on a *limited* set of *only positive* examples: the linguistic stimuli produced by the environment of the child. Those stimuli are utterances that help set certain parameters to their correct value, like the order of a verb. In English, for instance, verbs follow subjects (the so-called subject-verb-object [SVO] word order). In Japanese, on the other hand, verbs are found at the end of a sentence, making for a so-called SOV language. For the order of subject, object, and verb, there are only six possible permutations: SVO, SOV, VSO, VOS, OSV, and OVS. Each of these permutations is found in natural languages, which makes the position of the verb a parameter with six values. Another tenet of generative linguistics is the distinction between *competence* and *performance*. Competence addresses a mental model of grammar that reflects knowledge of a language, and performance relates to the operationalization of such knowledge, constrained by memory and other processing limitations. Generative linguistics is multistratal, with clear roles for, say, syntax and semantics.

Since LLMs appear to be able to learn languages quite well, what does this imply for linguistic and cognitive theories of human language acquisition? Principles and parameters do not play an overt role in LLMs. They may very well manifest themselves as statistical regularities, hidden somewhere in the myriad probabilities computed by LLMs, but they are not factored in explicitly by design. One of the common criticisms that LLMs get from generative linguists when presented as plausible theories of language acquisition is that they need to consume billions of words before they start showing off their magic. Yet we should not forget that LLMs, like many AI models, are not designed from the ground up as cognitively plausible models of learning. LLMs are part of a class of *invariant* machine learning algorithms; they need to observe many instances of objects (in their case, words in context) to learn to generate words in context. Invariant learning is also performed by many AI-based image analysis algorithms, like face or object detectors: systems that learn to recognize (detect) a face or some object. Such models need to be trained on numerous examples that differ in shape, scale, rotation, color, and lighting conditions in order to become invariant to changes in the data. This rather tedious learning process is aggravated by the downright crude operation of the so-called *backpropagation* process: a process for weight optimization during artificial neural network learning (see chapter 3). For every error a network makes during training, *all* weights in the network will be updated. So effective learning proceeds one item at a time, all items are important, and standard neural networks need many items to learn a concept. Such algorithmic crudeness is not shown in the human brain.[3]

Contrastingly, so-called *equivariant* machine learning models learn a topology: a model of the conceptual structure of a certain class of objects. An equivariant face detector would learn that a human face usually has a nose below two eyes and above a mouth. In that sense, such models infer a rendering instruction with which they can deal with variation; they “render” an observation on the basis of a generic topological template and need far less examples for that. Equivariance seems a more natural, cognitively more plausible way of learning, prompting

Geoffrey Hinton (the “godfather” of deep learning) and his colleagues to propose a specific neural network architecture precisely for this type of learning.[4]

Given that current LLMs are invariant learners, it may not come as a surprise that they need that much data. Analysis shows that ten to a hundred million words are necessary for LLMs to acquire a good grasp of syntax and semantics, including grammatical insight.[5] The rest of the material could be used for inferring commonsense knowledge about the world and “metalearning” skills for natural language understanding tasks like reasoning. This is certainly more data than an infant is presented with during its primary language learning phase but given the fact that LLMs are not optimized per se for cognitively plausible learning, there may be room for improvement. From a methodological perspective, Chomskyan linguistics is a *deductive* theory. It starts from principles, premises, and assumptions about underlying structure. It subsequently attempts to *infer* (explain, derive) observations from this starting point. In contrast, LLMs may be seen as *inductive* formalisms. They make no assumptions about the underlying data structure of language. Any structure imposed on language is a consequence of their inductive (as opposed to deductive) behavior, and most derived structure is rooted in observed co-occurrences of words in the huge amounts of their training data.

LLMs are part of a class of *invariant* machine learning algorithms; they need to observe many instances of objects (in their case, words in context) to learn to generate words in context.

Cognitive scientist Steven Piantadosi discusses at length the ongoing debate between generative linguists and machine learning researchers working on LLMs, responding to critical comments made by Chomsky and others about the alleged "autocomplete"-like behavior of LLMs and their lack of a theoretical foundation.[6] His overall conclusion is that LLMs bring a necessary change to linguistics by linking an—albeit mainly motivated by engineering—processing architecture to distributed representations and a principle-free form of linguistics, stating, "There is nothing comparable in all of linguistic theory to the power of Large Language Models in both syntax and semantics." He emphasizes that LLMs blur the distinction between syntax and semantics effectively by describing words and their relationships in a uniform data space called a "vector space" (more about that later) and thus embody a perspective on a simpler theory of language than generative linguistics. Hardly unexpected, researchers Jordan Kodner and colleagues and Roni Katzir replied to Piantadosi with a wide array of counterarguments defending the generativist perspective.[7]

Whether humans deploy similar models and architectures for learning language is still an unsettled issue. It is difficult —and may be downright impossible—to infer something about human brains from synthetic implementations like LLMs unless the latter are faithful and minimal characterizations (or even *theories*) of human cognitive behavior. At this point, probably all we can say is that LLMs are approximate models of the human language faculty, mimicking some of human linguistic behavior. But at least from an observational perspective, current LLMs outperform more traditional models of grammar by a large margin in terms of linguistic capabilities. Since the fields of machine learning and cognitive science have been linked intimately for decades, there is no reason to assume their future interaction will not yield additional, mutually beneficial insights into how language "works."

To sum up, the generative linguistics paradigm has—inadvertently—set the stage for an interesting and lively debate in the LLM and linguistics research communities about how humans (and machines) learn language from observations.

Logic-Based Natural Language Processing

Another computational perspective on language originates from logic (figure 5). In so-called propositional logic, logical inference is performed by constructing *propositions*: truth-bearing statements like "Every computer needs electricity." A proposition can be true or false, or even partially true or false, depending on a certain context.[8]

Every proposition *P* can be negated ("*not P*"), enter conjunctive relations with other propositions ("*P and Q*") or disjunctive relations ("*P or Q*"), and, importantly, participate in conditional relationships: "*if P then Q*," in symbols "$P \supset Q$," with "$\supset$" reading as "implies." Such implication can be defined in terms of negation and disjunction, incidentally: $P \supset Q$ is logically equivalent to "*not P or Q*." Implication allows for deducing conclusions (*Q*) based on a premise (*P*): If we have established the truth (whatever that means) of some proposition *P*, and we have a condition that says, from *P* being true we can infer *Q* being true, then infer *Q*. This pattern of reason is called *modus ponens*, an essential ingredient of propositional logic.