ChatGPT's Technical Foundations: Transformers to RLHF

By InuitionLabs.ai • 9/26/2025 • 40 min read

chatgpt · transformer architecture · rlhf · large language models · machine learning · artificial intelligence · natural language processing · self-attention

Key Innovations Behind ChatGPT: A Technical Deep Dive

ChatGPT is a product of converging breakthroughs in artificial intelligence research. Its abilities as a large language model were not born in isolation, but enabled by several pivotal inventions in machine learning and computing. This report examines the top five inventions that made ChatGPT possible: **(1)** the Transformer architecture, **(2)** large-scale unsupervised pretraining of language models, **(3)** GPU/TPU hardware acceleration for neural networks, **(4)** reinforcement learning from human feedback (RLHF), and **(5)** advanced tokenization methods like Byte Pair Encoding (BPE). For each innovation, we explore its origin and evolution, the underlying theory or mechanism, its contribution to ChatGPT's architecture, and the key people or institutions involved.

1. Transformer Architecture: The Self-Attention Revolution

Origin and Evolution

Before 2017, sequence models in NLP were dominated by recurrent neural networks (RNNs) such as LSTMs, which processed tokens sequentially and struggled with long-range dependencies en.wikipedia.org en.wikipedia.org. A major turning point came with the introduction of the **Transformer** architecture in the landmark 2017 paper *"Attention Is All You Need"* by Vaswani et al. at Google Brain en.wikipedia.org. This paper proposed an entirely new deep learning architecture based solely on an attention mechanism, dispensing with recurrence and convolution. The Transformer built upon the earlier idea of attention in seq2seq models (Bahdanau et al., 2014) by showing that *self-attention* alone could drive sequence transduction tasks like machine translation en.wikipedia.org. Eight researchers (Vaswani, Shazeer, Uszkoreit, Jones, Gomez, Kaiser, Parmar, Polosukhin) contributed equally to this work at Google, underscoring the collaboration involved in this breakthrough en.wikipedia.org. The proposal was initially met with some skepticism – even within the authors' circles – since it challenged the conventional wisdom that recurrence was needed for sequence modeling en.wikipedia.org. However, the results spoke for themselves: Transformers achieved state-of-the-art translation quality with significantly more parallelism than RNNs.

Following its debut, the Transformer architecture rapidly became the foundation of modern NLP. Already in 2018, OpenAI introduced the GPT (Generative Pre-Training) series – decoder-only Transformers for language modeling – and Google introduced BERT – an encoder-only Transformer for language understanding en.wikipedia.org. These models demonstrated the Transformer's versatility beyond translation, excelling in tasks from question answering to commonsense inference. By 2022, a chatbot based on a Transformer (GPT-3) – namely ChatGPT – gained worldwide popularity, cementing the Transformer as the architecture at the heart of the AI boom in large language models en.wikipedia.org. As of 2025, the original Transformer paper has been cited over 170,000 times, reflecting its foundational role in modern AI en.wikipedia.org.

Mechanism and Innovations

The key innovation of the Transformer is the **self-attention mechanism** combined with a deep feed-forward network architecture. Instead of processing words sequentially, the Transformer processes all tokens in parallel, using attention to allow each token to attend to (i.e. weightedly focus on) every other token's representation en.wikipedia.org. In self-attention, each position in the sequence produces three vectors: a Query, Key, and Value. Dot-products between Queries and Keys determine attention weights, which are used to aggregate Values from all positions. Crucially, the Transformer employs **multi-head attention**, meaning it uses multiple parallel attention "heads" to learn different aspects of the token relationships en.wikipedia.org en.wikipedia.org. Each head is a separate learned projection of Q/K/V, enabling the model to attend to different patterns (e.g. syntactic vs. semantic relations) simultaneously en.wikipedia.org en.wikipedia.org. These multiple attention outputs are then concatenated and linearly transformed, yielding a rich combined representation. This multi-head design, along with position-wise feed-forward layers and residual connections, gave Transformers greater modeling power than single-head or single-perspective models.

Another innovation was the use of **positional encoding** to inject sequence order information into the model, since pure self-attention is order-agnostic. Vaswani et al. used a fixed sinusoidal positional encoding added to input embeddings, which allowed the model to be aware of token positions and even generalize to sequence lengths not seen during training en.wikipedia.org en.wikipedia.org. Thanks to these features, Transformers overcame the main limitation of RNNs: they eliminated the need to propagate state through many time-steps, thus avoiding the vanishing-gradient bottleneck for long sequences en.wikipedia.org en.wikipedia.org. Instead, a Transformer can capture long-range dependencies in one pass via attention, albeit with a computational cost that scales quadratically with sequence length. Notably, the Transformer's parallelizable structure (processing tokens concurrently rather than sequentially) was *highly* attractive for modern hardware. The design was "easier to parallelize due to the use of independent heads and the lack of recurrence," which was a key factor in its widespread adoption for large-scale neural networks en.wikipedia.org. In the original experiments, a base Transformer (65M parameters) was trained on 8 NVIDIA P100 GPUs for 12 hours to outperform prior state-of-the-art models in translation en.wikipedia.org – a testament to its training efficiency and performance.

Influence on ChatGPT

ChatGPT directly builds on the Transformer architecture. In fact, the “GPT” in ChatGPT stands for “Generative Pre-trained Transformer.” The GPT series (GPT-1 in 2018, GPT-2 in 2019, GPT-3 in 2020) are all Transformer-based language models that use the decoder side of the Transformer (with masked self-attention allowing autoregressive text generation). Without the Transformer, ChatGPT’s core neural architecture would not exist in its current form. The self-attention mechanism enables the model to handle long dialogues and maintain contextual coherence by flexibly attending to relevant parts of the conversation history. When a user converses with ChatGPT, the model’s *multi-head self-attention* allows it to recall instructions and facts provided many turns earlier, something that RNN-based models struggled with. The Transformer’s capacity to capture such long-range context is a prime reason why ChatGPT can produce coherent multi-sentence answers and follow complex, branching instructions over a conversation.

Moreover, the Transformer architecture proved to scale exceptionally well. As model size and training data grew, Transformer models only became more powerful – a phenomenon captured by the scaling laws of Kaplan et al. and others. GPT-3, with 175 billion parameters, is essentially a very large Transformer trained on an enormous corpus (we will discuss the pretraining shortly). This scaling enabled emergent capabilities like few-shot learning arxiv.org arxiv.org. None of this would be feasible without the efficient parallelism of the Transformer. In summary, the Transformer was the crucial architectural invention that allowed researchers to build ChatGPT’s underlying model architecture, providing the efficiency and flexibility needed for training on huge datasets and enabling the complex behaviors we observe in its responses en.wikipedia.org en.wikipedia.org.

Key Contributors: The Transformer was developed by Google Brain researchers (Vaswani, Shazeer, et al.), drawing on prior work in attention by Bahdanau, Cho, and Bengio. Its success was quickly expanded upon by both academia and industry – OpenAI’s GPT series (Radford, Sutskever, et al.), Google’s BERT (Devlin et al.), and many others all leverage the Transformer at their core. This invention’s impact on ChatGPT and modern NLP cannot be overstated; it is the backbone architecture that made large language models viable.

2. Large-Scale Unsupervised Pretraining of Language Models

Emergence of Pretrained Language Models

A second major innovation enabling ChatGPT was the strategy of **large-scale unsupervised pretraining** on raw text, followed by fine-tuning for specific tasks. Traditionally, machine learning models in NLP were trained on task-specific labeled data, which is often limited in quantity. By the late 2010s, researchers realized that leveraging the *enormous* amount of unlabeled text on the Internet could be the key to advancing language understanding research.google. Early milestones included learning word embeddings (e.g. Word2Vec by Mikolov et al., 2013) where vectors for words were learned from unlabeled corpora, and **language modeling** – training models to predict the next word (or missing words) in sentences. In 2018, these ideas culminated in the introduction of powerful **contextual language models** like ELMo, GPT-1, and BERT research.google. Rather than training from scratch for each task, these models were first *pre-trained* on vast unlabeled text to learn general linguistic knowledge, and then *fine-tuned* on small datasets for downstream tasks, achieving breakthrough results. For example, OpenAI's GPT (Generative Pre-Training) model was pre-trained on BooksCorpus and shown to improve many NLP tasks after a fine-tuning step research.google. Around the same time, Google's BERT (Bidirectional Encoder Representations from Transformers) was pre-trained on Wikipedia and books using a masked-language-model objective, and it achieved then state-of-the-art accuracy on 11 different NLP benchmarks upon fine-tuning research.google research.google. These successes established unsupervised pretraining as a new paradigm in NLP.

Researchers soon observed a compelling trend: the larger the model and the more data it was pre-trained on, the better its performance and generalization. This led to an arms race in training ever bigger language models on ever larger corpora – a shift memorably summarized in a 2021 report by Bommasani et al., who termed such models "**foundation models**." According to the Stanford CRFM report, *"AI is undergoing a paradigm shift with the rise of models (e.g., BERT, DALL-E, GPT-3) that are trained on broad data at scale and are adaptable to a wide range of downstream tasks. We call these models foundation models…"* arxiv.org. These foundation models derive their power from being trained on *extremely large and diverse datasets*, allowing them to capture linguistic patterns, facts, and even commonsense reasoning to an extent never before seen. Critically, their scale yields **emergent capabilities** – behaviors that were not present in smaller models, such as few-shot learning (the ability to perform a task given only a couple of examples in the prompt rather than explicit fine-tuning) arxiv.org. The emergence of GPT-3 in 2020 epitomized this approach: with 175 billion parameters trained on an unprecedented 300 billion tokens of text, GPT-3 demonstrated surprising proficiency in tasks it had not been explicitly trained for, simply by virtue of massive self-supervised training developer.nvidia.com developer.nvidia.com.

Unsupervised Training at Scale

The underlying theory of this approach is *self-supervised learning*: the model learns from the raw data itself by solving an artificial task (such as next-word prediction or masked-word prediction). In doing so, it internalizes statistical structure of the language. Large-scale unsupervised pretraining harnesses this at scale – typically using **language modeling** as the training objective. For GPT models (like the one behind ChatGPT), the objective is autoregressive next-token prediction: the model iteratively learns to predict the next word given all previous words in a sequence. This simple goal requires the model to absorb extensive knowledge about syntax, semantics, and real-world facts from the training corpus in order to make accurate predictions.

The scale of data is a defining factor. GPT-3, for example, was trained on a corpus of approximately 45 TB of text data (filtered and tokenized), including a large portion of the Common Crawl (a scrape of the web), Wikipedia, books, and other texts springboard.com en.wikipedia.org. In fact, about 60% of GPT-3's training tokens came from a filtered Common Crawl dataset (410 billion tokens after BPE tokenization) en.wikipedia.org. This immense breadth of data gave the model exposure to virtually every domain of text – from literature and scientific articles to forum dialogues – thus broadening the model's general knowledge. The unsupervised nature is crucial: no human labeling is needed; the model learns from the naturally occurring context in text. This made it feasible to use orders of magnitude more data than any labeled dataset. Researchers also discovered that as models are made larger and trained on more data, they continue to improve rather than overfit – leading to the concept of *scaling laws*. These empirical scaling laws (Kaplan et al. 2020) suggested that performance scales predictably with more compute, data, and model size, further driving the push to train giant models.

For ChatGPT's development, OpenAI leveraged this innovation by first pretraining the base model (GPT-3.5 series) on a huge corpus in an unsupervised way. By doing so, the model acquired a vast repository of knowledge and linguistic ability. To illustrate the benefit: without large-scale pretraining, trying to teach a model to answer questions or hold a conversation would require labeled examples for every possible subject and query – clearly infeasible. Instead, thanks to unsupervised pretraining, ChatGPT already "knows" facts about the world and how language is used, before we ever fine-tune it to be a conversational agent. Large-scale pretraining thus provides the *general intelligence prior* that fine-tuning and human feedback can then shape into a useful assistant.

Contributions to ChatGPT's Capabilities

Unsupervised pretraining is arguably what makes ChatGPT so knowledgeable and fluent. During its extensive pretraining, the model has read and analyzed millions of articles, books, and websites, encoding not just vocabulary and grammar, but also factual information and common patterns of reasoning scattered across the internet. This is why ChatGPT can often answer questions about history, science, or programming – it has effectively absorbed a significant fraction of the written knowledge available online. The breadth of pretraining gives it a form of *transfer learning*: knowledge gained from one context (e.g. reading math textbooks) can be applied to another task (solving a math word problem in a conversation) without explicit task-specific training.

Moreover, the pretraining at scale gave rise to emergent abilities crucial for a helpful dialogue agent. One famous example documented with GPT-3 is **few-shot learning** arxiv.org arxiv.org. Even if ChatGPT was never explicitly trained on a given task (say, translating Klingon poetry), it can often perform it when prompted with a few examples or instructions, because the pretraining likely exposed it to related patterns or it can interpolate from its general language understanding. This zero- or few-shot generalization is a direct benefit of training on very large and varied data – the model can interpolate and recombine skills learned across the data. The

Stanford foundation model report emphasizes that the effectiveness of such models across so many tasks is a double-edged sword: it **"incentivizes homogenization"**, meaning one large pre-trained model can be adapted to many tasks, but also any flaws in the base model carry over arxiv.org. For ChatGPT, OpenAI leveraged this by using one foundational model (GPT-3.5 or GPT-4) and applying instruction-tuning and RLHF (see below) to get different behaviors, rather than training separate models from scratch for each behavior.

In summary, the invention of large-scale unsupervised pretraining allowed AI practitioners to create **foundation language models** that serve as general-purpose learners arxiv.org. This was crucial for ChatGPT: the pretraining makes it fluent and knowledgeable, and subsequent steps (like RLHF) make it aligned and conversational. Without unsupervised pretraining, ChatGPT would neither have the knowledge base nor the linguistic intuition that underpin its responses. Key contributors to this paradigm include researchers at OpenAI (authors of the GPT papers such as Alec Radford, Jared Kaplan, Tom Brown, et al.), at Google (BERT by Jacob Devlin et al.), as well as academic proponents of transfer learning in NLP (such as Ruder, Howard & Manning for ULMFiT and ELMo). The collective innovation was recognizing that *scale* and *unlabeled data* are critical resources for achieving high-level AI capabilities, a principle that ChatGPT's development fully capitalized on research.google research.google.

3. GPU and TPU Hardware: Accelerating Neural Network Training

Rise of GPU-Accelerated Deep Learning

Training ChatGPT's model (with hundreds of billions of parameters) is an extremely computationally intensive task. A crucial invention that made such training feasible is the use of **Graphics Processing Units (GPUs)** – and later specialized accelerators like **Tensor Processing Units (TPUs)** – for neural network computation. The origin of this idea dates back to the early 2010s when researchers discovered that GPUs, originally designed for rendering graphics, are also adept at the linear algebra operations (matrix multiplications, vector operations) at the heart of deep learning. A watershed moment was the 2012 success of **AlexNet**, a deep convolutional network trained on two NVIDIA GPUs, which won the ImageNet competition by a large margin blogs.nvidia.com. AlexNet's victory dramatically demonstrated that GPUs could train large neural nets that were previously impractical, thanks to a ~10x to 20x speedup over CPUs blogs.nvidia.com. As NVIDIA's CEO Jensen Huang recounted, by 2011 researchers found that a dozen NVIDIA GPUs could replace thousands of CPU cores for deep learning workloads blogs.nvidia.com. This "Big Bang" of GPU-accelerated computing ignited the modern AI revolution blogs.nvidia.com blogs.nvidia.com. In parallel, software advances like NVIDIA's CUDA and cuDNN libraries made it easier to program GPUs for AI, and deep learning frameworks (TensorFlow, PyTorch) incorporated GPU support.

The evolution since then has been rapid. GPUs themselves improved generation by generation, adding architectural features beneficial for AI – for instance, NVIDIA's introduction of **Tensor Cores** (specialized hardware for mixed-precision matrix multiply-accumulate) in its Volta and later GPU architectures gave another massive boost to training speed. Researchers also developed techniques for distributed training across many GPUs in parallel, to handle larger models and datasets. By the late 2010s, it became common to use GPU *clusters* (tens or hundreds of GPUs connected with high-speed interconnects) to train frontier models. Meanwhile, Google developed the **TPU** – a custom ASIC (application-specific integrated circuit) optimized for tensor operations – deploying TPU v2 and v3 in its data centers for accelerating both training and inference of neural networks. TPUs were used in training Google's BERT and later models, and have comparable performance to contemporary GPUs in those tasks. In a 2018 commentary, the BERT team noted that *Cloud TPUs gave us the freedom to quickly experiment and tweak our models… The Transformer model architecture… gave us the foundation we needed to make BERT successful* research.google, highlighting how advanced hardware and new architectures went hand-in-hand.

Hardware Innovations for Scaling

Under the hood, why are GPUs/TPUs so effective for neural networks? The answer lies in parallelism. Neural network training involves a huge number of linear algebra operations that can be done in parallel – for example, multiplying large weight matrices with input vectors for each layer, across many neurons and many training examples simultaneously (mini-batch processing). GPUs were designed to execute thousands of threads in parallel, originally to draw many pixels at once. This suits matrix math: GPUs can perform thousands of multiply-adds concurrently, vastly accelerating the compute-intensive forward and backward passes of neural networks. By contrast, traditional CPUs have fewer cores optimized for serial tasks. NVIDIA reports that its GPUs sped up deep learning training by an initial 10-20x around the early 2010s, turning multi-week trainings into days blogs.nvidia.com. They then continued to refine the whole stack – from hardware to libraries – achieving up to 50× speedups in a few years after AlexNet blogs.nvidia.com. TPUs follow a similar principle: they include matrix multiply units and a high-bandwidth memory architecture, allowing massive numbers of operations per second dedicated to neural network computations, with Google's TPUv4 pod containing thousands of chips networked together for exascale compute levels.

Scaling to training ChatGPT required not just single GPUs, but **clusters of accelerators**. OpenAI's GPT-3 (2020) was trained on a Microsoft Azure supercomputer that the companies built in partnership – it reportedly had **>285,000 CPU cores and 10,000 GPUs** connected with high-bandwidth links developer.nvidia.com. According to OpenAI, *"All models were trained on NVIDIA V100 GPUs on part of a high-bandwidth cluster provided by Microsoft."* developer.nvidia.com The cluster's design allowed thousands of GPUs to work on the training in parallel by splitting both the model (across GPUs' memory) and the data batches (across GPU nodes). This distributed training approach is what enabled the months-long training of GPT-3 to be completed in a reasonable time (estimated at a few thousand petaflop/s-days of compute).

NVIDIA's blog noted that Microsoft's AI supercomputer built for OpenAI was a single system with 10,000 GPUs and 400 Gbps networking per GPU server, representing one of the most powerful AI-dedicated infrastructures at the time developer.nvidia.com. By the time of ChatGPT's deployment (2022-2023), it was reported that OpenAI was using even more advanced hardware like NVIDIA A100 or H100 GPUs for both training refinements and serving the model, and projections suggested ChatGPT and successors would require tens of thousands of GPUs to handle user demand tomshardware.com.

This hardware innovation not only made training possible, but also affects how ChatGPT operates in real-time. Inference (model serving) for large transformers is also computationally heavy; GPUs are used to execute the model's forward pass so that ChatGPT can generate responses with reasonable latency. Microsoft's Azure, in collaboration with OpenAI and NVIDIA, built a dedicated AI cloud infrastructure to host ChatGPT. As Microsoft CTO Mark Russinovich explained, they developed a *"purpose-built AI supercomputer infrastructure… in collaboration with OpenAI, to host ChatGPT and other large language models at any scale."* developer.nvidia.com. This includes clusters of GPUs with fast interconnects, specialized networking (InfiniBand), and optimized software for deployment. Without such engineering, serving millions of ChatGPT users would be impractical.

Enabling ChatGPT's Training

In essence, advanced hardware is the enabler of everything discussed previously. The Transformer architecture and massive pretraining would be academic ideas only, were it not for GPUs/TPUs providing the raw horsepower to train billion-parameter models on billions of tokens. Each of ChatGPT's training iterations involved calculating enormous matrices of activations and gradients. GPUs made this tractable by parallelizing those calculations. To put it in perspective, training GPT-3 (175B) required on the order of 3.14×10^{23} FLOPs (floating point operations) developer.nvidia.com – a number only reachable in practice by harnessing many GPUs over extended periods. Thanks to hardware acceleration, OpenAI's researchers could experiment with increasingly larger models and found, somewhat surprisingly, that *bigger was better*. If the AI field had been limited to CPU training, we likely would not have seen models beyond a few billion parameters due to time and cost constraints.

Moreover, hardware advances continue to directly influence model development. For instance, larger memory GPUs (with 80GB or more per card) allow for training models with longer context windows and higher batch sizes, improving performance. Faster GPU generations (e.g. NVIDIA H100) offer significant speedups; NVIDIA reported that using data parallelism and their latest GPUs gave *"30× higher inference throughput and 4× higher training performance"* for LLMs compared to the previous generation developer.nvidia.com. Such improvements mean new versions like GPT-4 could be trained on even more data or with more complex architectures without prohibitive costs. In short, the co-evolution of hardware and algorithms is a key storyline in ChatGPT's development. *Key figures* in this domain include NVIDIA (led by Jensen Huang) which championed GPUs for AI, the Google Brain team (Norman Jouppi and colleagues who designed the TPU), and engineering teams at Microsoft and OpenAI who built the large-scale training infrastructure developer.nvidia.com. The synergy between hardware and model design enabled the leap from academic models to a deployed system like ChatGPT that millions can use.

4. Reinforcement Learning from Human Feedback (RLHF)

Origin of Human-in-the-Loop Training

While a large pretrained Transformer can generate text, early versions of GPT-3 often produced outputs that were correct or fluent, but not aligned with what users actually wanted – they might ignore instructions, produce irrelevant answers, or exhibit toxic biases. The invention that bridged this gap and *transformed a raw language model into ChatGPT* is **Reinforcement Learning from Human Feedback (RLHF)**. RLHF is a technique that uses human feedback to optimize model behavior, first explored in the context of AI safety and alignment research. Its origins trace back to around 2017, when OpenAI researchers like Paul Christiano and colleagues proposed using human preference comparisons as a reward signal to train reinforcement learning agents en.wikipedia.org. The initial papers (Christiano et al. 2017) demonstrated that an RL agent could learn complex behaviors (like performing a backflip in a simulator) by optimizing a reward model trained on human judgments, instead of a hand-coded reward function. This idea of learning a *reward function* from humans and then using RL to optimize policy laid the groundwork for RLHF.

Applying RLHF to language generation was pioneered by OpenAI in subsequent years. In 2019-2020, OpenAI researchers used human feedback to improve summarization systems (Stiennon et al. 2020) en.wikipedia.org. The major breakthrough for language models came with OpenAI's **InstructGPT** (Ouyang et al., 2022), which fine-tuned GPT-3 using RLHF to better follow user instructions. This was essentially the research precursor to ChatGPT. As the OpenAI team wrote, *"fine-tuning language models with humans in the loop is a powerful tool for improving their safety and reliability"* openai.com. By January 2022, they had deployed these InstructGPT models as the default models in the OpenAI API, finding that even a 1.3B parameter model trained with RLHF could outperform a 175B GPT-3 model on following instructions openai.com. This highlighted how effective RLHF was in aligning model behavior with user intent. The key people behind these advances include OpenAI's alignment team (Jan Leike, Paul Christiano, Ryan Lowe, Jeffrey Wu, and others) who for several years had been developing this approach openai.com. By the time ChatGPT was introduced (late 2022), RLHF had become a proven method to significantly enhance helpfulness and reduce harmful outputs in large language models openai.com openai.com.

RLHF Process and Theory

Reinforcement Learning from Human Feedback can be understood as a three-step training procedure applied after the initial pretraining of the model. In summary, the steps are: **(1)** Supervised Fine-Tuning (SFT) with human demonstrations, **(2)** Reward Model training from human preferences, and **(3)** Policy optimization with RL. We can break these down:

- **Step 1: Supervised Fine-Tuning (SFT).** First, a base language model is fine-tuned on a curated dataset of prompts and ideal responses. These responses are provided or written by human annotators as examples of the desired behavior (e.g., helpful answers, polite tone). For ChatGPT, OpenAI labelers would take various user prompts and write high-quality answers manually openai.com. The model is initialized from the pretrained GPT and then trained to imitate these human-written responses. This gives an initial model that already follows instructions better than the raw GPT-3.

- **Step 2: Reward Model (RM) Training.** Next, a separate model is trained to act as a **reward function** that captures human preferences en.wikipedia.org en.wikipedia.org. Humans are shown multiple outputs that the current language model produces for a given prompt (for example, two or more different answers to the same question). They then rank these outputs from best to worst according to which answer is more helpful, correct, or aligned. Using this data, the reward model is trained (via supervised learning) to predict a higher score for outputs that humans preferred and a lower score for outputs they disliked en.wikipedia.org. Essentially, the RM learns to approximate the human evaluator's judgments. In practice, the reward model often shares the same architecture as the language model (e.g., a smaller Transformer) and is trained on a large number of comparisons. The OpenAI InstructGPT paper leveraged this method, citing earlier work that introduced the algorithm in a text continuation/summarization setting en.wikipedia.org.

- **Step 3: Reinforcement Learning (Policy Optimization).** In the final step, the language model (the policy) is fine-tuned using reinforcement learning, to directly maximize the reward model's score for its outputs en.wikipedia.org en.wikipedia.org. Typically, the Proximal Policy Optimization (PPO) algorithm is used, which is a stable RL method for gradient ascent on expected reward. The model generates an output for a given prompt, the reward model scores it, and the policy's parameters are adjusted to increase the likelihood of high-scoring outputs. Over many iterations, this optimizes the language model to produce answers that align better with human preferences as encoded by the reward model. Crucially, using the reward model as a proxy allows for automation – the system no longer requires a human in the loop for every single policy update, only for the initial comparisons. This step can be seen as the model *learning from feedback*: it is no longer purely predicting likely text (as in pretraining), but rather choosing text that maximizes an approximation of human satisfaction.

Illustration of the three-phase RLHF pipeline used to fine-tune ChatGPT (OpenAI InstructGPT methodology). In phase 1 (Supervised Fine-Tuning), the pretrained model is tuned on demonstration examples of good behavior. In phase 2, a reward model is trained to predict human preference rankings. In phase 3, the model is further optimized with reinforcement learning (e.g. PPO) to generate responses that score highly according to the reward model. openai.com en.wikipedia.org

The theoretical underpinning of RLHF is that it combines *human intuition* with *reinforcement learning* to align AI behavior with human values or goals. Standard reinforcement learning requires a reward function – something that scores how well the agent is doing at its task. Designing a reward function for "how helpful or correct is this answer?" is extremely hard to do with code, but relatively easy for humans to judge when they see an answer en.wikipedia.org en.wikipedia.org. RLHF leverages this by having humans provide examples and preferences, which indirectly define a reward function that the model can then optimize. One challenge is that the reward model is an imperfect proxy; if it's not carefully trained or if the model finds loopholes, it can lead to *misalignment* (e.g., the model might learn to game the reward by producing answers that sound pleasing but are subtly incorrect – known as the "Goodhart's law" issue in RLHF). Addressing these challenges is an ongoing research area in AI alignment. However, empirically, RLHF has proven remarkably effective at steering model behavior: as noted, users and evaluators strongly prefer outputs from RLHF-tuned models over those from the original model in head-to-head comparisons openai.com.

Alignment of ChatGPT with Human Intent

For ChatGPT, RLHF was the game-changer that turned a capable but sometimes unfocused language model into a reliable conversational agent. After applying RLHF, ChatGPT began to **follow user instructions** much more literally and helpfully (hence the name "InstructGPT" for the intermediate model). OpenAI reported that their RLHF-tuned model "makes up facts less often, and generates more appropriate outputs" compared to the base GPT-3, and that humans greatly preferred the new model's outputs openai.com openai.com. In practical terms, this means if you ask ChatGPT a question, it tries to actually answer the question directly, or if you give it an instruction, it attempts to carry it out, whereas a raw LM might have ignored the instruction and continued some unrelated text. RLHF aligned the model's objective with *what the user wants*, rather than just next-word probability. It's worth noting that RLHF doesn't give the model new knowledge (that comes from pretraining), but it changes how that knowledge is used to produce outputs.

RLHF also helped address issues of **toxicity, bias, and safety**. By incorporating human preferences, designers could explicitly train the model to avoid certain classes of outputs (for instance, hate speech, harassment, or revealing private information) by giving such examples low reward during training. The human evaluators were instructed to rank outputs higher if they were truthful and harmless. As a result, the RLHF process inherently *filters out* a lot of unwanted behavior. OpenAI observed small decreases in toxic output generation and factual errors after RLHF openai.com. In effect, RLHF serves as a form of *value alignment* for language models – aligning the model with human ethical and practical values to some extent en.wikipedia.org. This is not perfect and there are ongoing concerns about biases in the feedback or reward model, but it's a substantial improvement over an unaligned model.

From an innovation perspective, RLHF was a fairly novel idea in the context of large language models. It combined techniques from reinforcement learning (policy optimization, reward modeling) with large-scale human data collection. The fact that it worked so well for GPT-3 was somewhat surprising even to researchers – it showed that even without solving "AI understanding", we could use brute-force optimization guided by human preference to significantly improve an AI's usefulness. **ChatGPT's success and distinctive conversational quality owes a great deal to RLHF**: it is the reason the model asks for clarifications, apologizes for errors, refuses certain requests, and overall behaves in a manner more aligned with user expectations for a helpful assistant openai.com openai.com. This has set a new standard; now nearly all major large language models (Anthropic's Claude, Google's Bard, etc.) employ some form of human feedback fine-tuning as part of their training regimen medium.com en.wikipedia.org.

Key contributors: RLHF for language models was developed by OpenAI's alignment research group, with early conceptual work by Paul Christiano et al., and later implementation in InstructGPT led by Long Ouyang, Jeff Wu, and others. Collaborations with teams labeling data (oftentimes contractors or in-house annotators) were also crucial. This innovation represents a convergence of ideas from reinforcement learning, human-computer interaction, and ethics, making it a standout technique in the journey to ChatGPT.

5. Tokenization and Byte Pair Encoding (BPE)

Challenges of Open-Vocabulary NLP

The final invention on our list is perhaps more low-level, but it is fundamentally important for making models like ChatGPT work with human language: **tokenization**, especially subword tokenization algorithms like **Byte Pair Encoding (BPE)**. Language is composed of words, but words can be incredibly diverse – there are millions of possible words, names, or misspellings one might encounter (an "open vocabulary" problem). Early language models often used a fixed word vocabulary, which struggled with out-of-vocabulary (OOV) words (anything not in the training vocab would be impossible to generate or understand properly). The invention of subword tokenization in mid-2010s solved this by breaking text into smaller pieces (subword units) that can flexibly recombine to form words. This allows the model to cover any possible word or even make up new words, using a manageable set of components.

One influential method was introduced by Rico Sennrich and colleagues in 2015: they adapted **Byte Pair Encoding (BPE)** – an algorithm from data compression – for word segmentation arxiv.org arxiv.org. In their ACL 2016 paper *"Neural Machine Translation of Rare Words with Subword Units"*, Sennrich et al. describe a way to enable open-vocabulary translation by encoding rare and unknown words as sequences of subword pieces arxiv.org. Their intuition was that even if a word is unseen, its parts (morphemes, roots, syllables) might be seen in other words. BPE starts with individual characters and iteratively merges the most frequent adjacent character sequences, forming longer and longer tokens en.wikipedia.org en.wikipedia.org. For example, it might learn that "j" + "umping" often occur together and merge into "jumping", but a very rare word might remain split into smaller units. The result is a vocabulary of subword tokens – including whole common words and meaningful pieces of rare words. This method was quickly adopted in machine translation and then in language modeling because it vastly reduced OOV issues. A BPE vocabulary of size 30k–50k can cover virtually all words by construction (since any word can be formed from characters) jlibovicky.github.io. In practice, BPE "keeps the most frequent words intact while splitting the rare ones into subword pieces," striking a balance between too large a vocabulary and too long a character sequence researchgate.net.

Around the same time, Google developed a similar approach called **WordPiece** (used in their translation system and in BERT), and others like **SentencePiece** (Unigram LM by Kudo) emerged, but the core idea is shared: *subword tokenization*. These algorithms became essential for all major NLP models. The original Transformer paper itself used byte-pair encoding to prepare its training data en.wikipedia.org. OpenAI's GPT-2 in 2019 employed a variant: they used BPE on raw bytes (byte-level BPE), allowing the model to even handle any unicode characters or binary data robustly. The GPT-2/GPT-3 BPE vocabulary size was 50,000, meaning the model represents text as sequences of integers (each corresponding to a subword token) from a set of 50k possible tokens en.wikipedia.org en.wikipedia.org. This approach lets the model deal with, say, a new hashtag or a misspelled word by breaking it into pieces that were seen during training. By the time of GPT-3.5 and GPT-4, tokenization schemes further evolved (GPT-4's tokenizer has over 100k tokens, allowing very long contexts and more direct encoding of text) en.wikipedia.org en.wikipedia.org.

Subword Tokenization with BPE

The underlying mechanism of BPE tokenization can be summarized as follows: the algorithm begins with all characters (and possibly some pre-defined tokens like spaces or punctuation) as the initial vocabulary. Then, it scans a large training corpus to find which pairs of tokens occur most frequently next to each other. That pair is merged into a new single token. This process repeats – each iteration adding one new token (the frequent pair) to the vocabulary and merging it in the text – until the vocabulary reaches a desired size (e.g., 50k). The outcome is a vocabulary that includes single letters, common prefixes/suffixes, and entire words for very frequent words. Rare words get broken down. For example, suppose "Atlantis" is not common; the tokenizer might split it into "At", "lan", "tis". If those pieces were seen in other contexts ("Atlantic", "lantis"), the model can still handle "Atlantis" by composing those subwords. In contrast, a character-level model would have to spell everything out letter-by-letter (making sequence lengths very long), and a word-level model would have to treat "Atlantis" as an unknown. Subword models get the best of both worlds – shorter sequences than characters and coverage of any word like a char model.

For ChatGPT's model, tokenization is the very first step that happens when input text is fed to it: the input string is chopped into tokens according to the learned BPE rules. The model then processes those tokens (each mapped to an embedding vector). On output, the model generates tokens which are then decoded back to text. Errors or limitations in tokenization can affect the model; for instance, if a word is split in an unnatural way, the model might struggle with it. But BPE has been shown to be highly effective. It was noted in the literature that BPE segmentation keeps frequent words intact which improves efficiency, and splits rare words into pieces that still often carry meaning (like "anti" + "body" for "antibody") researchgate.net. In essence, BPE encodes a **compressed representation of text** that neural networks can more easily learn from arxiv.org.

One interesting aspect is that tokenization also influences the efficiency of training and inference. With subwords, the average length of sequences (in tokens) is shorter than with characters, which means fewer time steps for the Transformer to iterate over. Yet it's long enough to avoid an enormous embedding matrix that a full word vocabulary would require (with millions of entries, many unused). BPE's data-driven approach finds a sweet spot. It also helps with multilingual models – subword pieces can often be shared between languages (e.g., "respon" in "response" and "responsabilidad"), improving cross-lingual transfer.

Role in ChatGPT and Modern LLMs

Without advanced tokenization, ChatGPT would not handle the open-ended nature of human language as smoothly. BPE tokenization allows ChatGPT to **recognize and generate any word or sequence of characters** it encounters, as long as it can break it into known pieces. This is critical for user interactions, because users might input slang, typos, code snippets, or rare names – none of which can be pre-listed exhaustively. Thanks to subword tokenization, the model can parse these inputs. For example, if a user asks about "XÆA-12" (an unusual name), the tokenizer might split it into "X", "Æ", "A", "-", "12". The model has embeddings and training for each of those and can thus form a representation for the whole. If the model had a closed vocabulary of common words, it would fail on such inputs.

Moreover, tokenization influences how the model counts length and thus its context window. ChatGPT's maximum input length is often described in tokens (for GPT-3 it was 2048 tokens; for GPT-4 models can handle even more). This token count corresponds to roughly a few thousand words of English text. The use of tokens ensures that this count has consistent meaning across languages and scripts (since tokens are normalized pieces). It's also the unit of the model's probability distribution when generating text – ChatGPT generates one token at a time. If tokens were too small (characters), generation would be slower and text quality could suffer (since the model would have to learn spelling from scratch). If tokens were whole words, the model would struggle with new words or miss subtle morphology. Thus the adoption of BPE/tokenization was a crucial practical invention to make large language models workable in real-world settings.

In the development history, **Sennrich et al.'s 2016 paper** is widely credited for popularizing subword units in NMT arxiv.org arxiv.org, and Google's *SentencePiece* (2018) further made it easy to integrate such tokenization. OpenAI implemented their own version for GPT-2/GPT-3, emphasizing byte-level coverage (hence the ability to handle any Unicode symbol by decomposing it to bytes). All these advances solved the "OOV problem" and have been inherited by ChatGPT. When we say ChatGPT has a vocabulary of size X, we really mean the set of subword tokens it knows. Interestingly, ChatGPT's knowledge of word composition can sometimes be seen in its errors – e.g., it might break a hashtag oddly. That reflects the tokenization. On the positive side, it can handle very long numbers or novel compounds because it will just generate them piece by piece.

To summarize, tokenization schemes like BPE were a key enabling innovation that allows large language models to function on real text. They were developed to handle open vocabulary efficiently, and have become a standard component in the NLP pipeline. **ChatGPT would not be possible without BPE and similar subword tokenizers**, as the model needs to reliably encode any input text into tokens and back. It's a great example of an idea borrowed from another domain (compression algorithms) becoming a linchpin in modern AI. Key contributors here include the University of Edinburgh team (Sennrich, Haddow, Birch) for BPE in NLP arxiv.org, Google's research for WordPiece (Schuster & Nakajima) and SentencePiece (Kudo), and OpenAI's engineering for scaling these methods to their models. While perhaps less glamorous than Transformers or RLHF, tokenization is one of the behind-the-scenes inventions that enabled ChatGPT to interact with us in writing as fluently as it does.

References

- Vaswani, A. et al. (2017). *"Attention Is All You Need."* Advances in NIPS 30. Introduced the Transformer architecture based on self-attention en.wikipedia.org en.wikipedia.org.
- Devlin, J. et al. (2018). *"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding."* Showed large-scale unsupervised pretraining yields state-of-the-art NLP results research.google research.google.
- Brown, T. et al. (2020). *"Language Models are Few-Shot Learners."* (GPT-3 paper). Demonstrated that scaling model size and data produces emergent capabilities arxiv.org arxiv.org. GPU cluster details: GPT-3 trained on 10,000 NVIDIA V100 GPUs provided by Microsoft Azure developer.nvidia.com developer.nvidia.com.
- NVIDIA Blog (2020). *"OpenAI Presents GPT-3, a 175 Billion Parameters Language Model."* Described GPT-3's performance and that it was trained on NVIDIA GPUs with Microsoft's high-bandwidth AI supercomputer developer.nvidia.com developer.nvidia.com.
- Huang, J. (2016). *"Accelerating AI with GPUs: A New Computing Model."* NVIDIA Blog. Explained how GPUs accelerated deep learning by 10-20x and enabled the modern AI boom blogs.nvidia.com blogs.nvidia.com.
- OpenAI (2022). *"Aligning language models to follow instructions."* OpenAI Blog (Ouyang et al.). Announced InstructGPT and use of RLHF to make GPT-3 more truthful and less toxic openai.com openai.com.
- Wikipedia (2023). *"Reinforcement learning from human feedback."* Definition and background of RLHF, noting OpenAI's contributions in summarization and InstructGPT en.wikipedia.org en.wikipedia.org.

- OpenAI (2022). *“Training language models to follow instructions with human feedback.”* (InstructGPT paper). Detailed the 3-step RLHF method (SFT, reward model, PPO) used to align GPT-3 openai.com en.wikipedia.org.
- Sennrich, R. et al. (2016). *“Neural Machine Translation of Rare Words with Subword Units.”* ACL. Introduced BPE for open-vocabulary translation, merging character sequences to represent rare words arxiv.org arxiv.org.
- Wikipedia (2023). *“Byte Pair Encoding.”* Describes the BPE algorithm and its adoption in large language model tokenizers en.wikipedia.org en.wikipedia.org.
- Wikipedia (2023). *“GPT-3.”* Notes the Common Crawl dataset (410B BPE tokens, ~60% of training) used to train GPT-3 en.wikipedia.org.
- NVIDIA Technical Blog (2023). *“What Runs ChatGPT?”* (Azure + NVIDIA collaboration). Describes the Azure supercomputer infrastructure with NVIDIA H100 GPUs built to host ChatGPT developer.nvidia.com.
- Bommasani, R. et al. (2021). *“On the Opportunities and Risks of Foundation Models.”* Stanford CRFM report. Coins the term “foundation model” and discusses how scale of pretraining leads to broad capabilities arxiv.org arxiv.org.
- **Additional references**: OpenAI & DeepMind publications on RLHF and safety, Google’s SentencePiece documentation, etc., as cited inline above openai.com blogs.nvidia.com researchgate.net.

IntuitionLabs - Industry Leadership & Services

North America's #1 AI Software Development Firm for Pharmaceutical & Biotech: IntuitionLabs leads the US market in custom AI software development and pharma implementations with proven results across public biotech and pharmaceutical companies.

Elite Client Portfolio: Trusted by NASDAQ-listed pharmaceutical companies including Scilex Holding Company (SCLX) and leading CROs across North America.

Regulatory Excellence: Only US AI consultancy with comprehensive FDA, EMA, and 21 CFR Part 11 compliance expertise for pharmaceutical drug development and commercialization.

Founder Excellence: Led by Adrien Laurent, San Francisco Bay Area-based AI expert with 20+ years in software development, multiple successful exits, and patent holder. Recognized as one of the top AI experts in the USA.

Custom AI Software Development: Build tailored pharmaceutical AI applications, custom CRMs, chatbots, and ERP systems with advanced analytics and regulatory compliance capabilities.

Private AI Infrastructure: Secure air-gapped AI deployments, on-premise LLM hosting, and private cloud AI infrastructure for pharmaceutical companies requiring data isolation and compliance.

Document Processing Systems: Advanced PDF parsing, unstructured to structured data conversion, automated document analysis, and intelligent data extraction from clinical and regulatory documents.

Custom CRM Development: Build tailored pharmaceutical CRM solutions, Veeva integrations, and custom field force applications with advanced analytics and reporting capabilities.

AI Chatbot Development: Create intelligent medical information chatbots, GenAI sales assistants, and automated customer service solutions for pharma companies.

Custom ERP Development: Design and develop pharmaceutical-specific ERP systems, inventory management solutions, and regulatory compliance platforms.

Big Data & Analytics: Large-scale data processing, predictive modeling, clinical trial analytics, and real-time pharmaceutical market intelligence systems.

Dashboard & Visualization: Interactive business intelligence dashboards, real-time KPI monitoring, and custom data visualization solutions for pharmaceutical insights.

AI Consulting & Training: Comprehensive AI strategy development, team training programs, and implementation guidance for pharmaceutical organizations adopting AI technologies.

Contact founder Adrien Laurent and team at https://intuitionlabs.ai/contact for a consultation.