IJHSR
ijhighschoolresearch.org
REVIEW ARTICLE

A Survey of Optimization and Compression Techniques for Natural Language Processing Models

Nathan C. Lam

Stuyvesant High School, 345 Chambers St, New York, NY, 10282, USA; nathanclam@icloud.com
Mentor: Siddharth Krishnan, Plinio Zanini

ABSTRACT: Natural Language Processing models have proven to be incredibly useful, boasting a tremendous number of practical applications spanning nearly all fields of study. The ability of NLP models to understand human language can be applied to a wide range of tasks, from generating medical reports to advanced translation. However, these abilities scale significantly with model size, so smaller models experience an exponential performance falloff when compared to large models. Large Language Models (LLMs) are at the forefront of NLP, and can have up to trillions of parameters. These programs are incredibly resource-intensive in their training and operation, and smaller organizations may be unable to properly utilize the full potential of current NLP technology. This paper aims to provide a survey of techniques that can make high-performance NLP models more accessible, whether it be by increasing the performance of weaker models or compressing larger, high-performance models to have more manageable requirements.

KEYWORDS: Robotics and Intelligent Machines, Machine Learning, Natural Language Processing, Large Language Model, Small Language Model, Model Compression, Model Optimization.

Introduction

Natural Language Processing is a field of AI centered around the 'natural' understanding and manipulation of human language by AI models. Natural Language Processing models (NLP models) are AI models that are built with this exact purpose in mind. They can generate and analyze text, translate text between supported languages much more accurately than non-AI machine translation, and even solve problems through logic.[1]

Some researchers believe that NLP models merely display the facade of intelligence through statistical prediction,[2-5] and others believe in the legitimacy of their reasoning abilities.[1,6-8] Despite this debate, this logical behavior is still incredibly useful in fields like medicine,[9] finance,[10] and computer science.[11]

The incredible potential of NLP models to be utilized in the world has one major obstacle, however. All of these characteristics are highly dependent on the size of the model, which is incredibly expensive to scale. The training, tuning, and operation of large-scale NLP models (Large Language Models/LLMs) takes an enormous amount of computational power, training data, and time. Additionally, most current research on optimization focuses heavily on larger models, as they are the current state-of-the-art (SOTA) technology in this field.[12-15]

The best solution to this problem, and the main focus of this paper, is the utilization of Small Language Models (SLMs). SLMs are smaller NLP models that have much lower resource requirements, but this comes at the cost of significantly lower performance in NLP tasks. Fortunately, there exists ample research around both the optimization of smaller NLPs' performance through techniques like prompt engineering and fine-tuning, and the compression of LLMs into SLMs with techniques like pruning and distillation. This paper aims to provide a survey of such methods and describe how they work; Examples of particular techniques will be provided in order to stimulate more interest in the specific field.

This paper will first examine the exact definitions of the terms LLM and SLM, the differences between the two, and how SLMs are a much better choice for the average person's utilization. Then it will review performance-enhancing methods for SLMs and compression methods for LLMs. Finally, it will provide an overview of promising future prospects, as well as a statement on the necessity of such innovation in order to further the state of the world.

SLMs and LLMs: Background Information

2.1. : What are LLMs and SLMs?

To fully understand what LLMs and SLMs are. It is necessary to first understand transformer models. Transformer models are a type of model that has an attention mechanism as the main component. A simplified explanation of how an attention mechanism works is that it allows the model to focus more on information that is deemed important by the mechanism. A much more mathematical and detailed explanation can be found in this paper by A. Vaswani *et al*. This unique characteristic of transformer models makes them very useful in a variety of fields,[16] but to understand this paper, all that you need to know is that almost all modern SOTA Natural Language Processing models are based on the transformer architecture.

Natural Language Processing models are a group of models designed to manipulate and 'understand' human language in a more human-like way. These models are incredibly versatile due to this function and are used in a variety of applications such as

DOI: 10.36838/IJHSR85.76

chatbots,[17,18] better translation of languages,[19] and analysis of text.[20] Large Language Models and Small Language Models are subsets of Natural Language Processing models, being the categorization of NLP models into larger and smaller models by their number of parameters.

LLMs are NLP models with incredibly large numbers of parameters. The current state-of-the-art is hundreds of billions to trillions of parameters. SLMs are NLP models with relatively low amounts of parameters, ranging from hundreds of millions to a few billion. The definitions aren't quite as concrete because of the rapid growth of the field, and now even models with as many as ten billion parameters can be considered SLMs as well, depending on sources.

2.2. : Differ ence in Requirements:

The main difference between LLMs and SLMs is, of course, their size, with the largest LLMs having easily thousands of times the parameters of an average SLM. This is why LLMs perform much better than SLMs in language processing. However, this comes with a major downside, which is the cost.

Training a state-of-the-art level LLM can require up to thousands of petaFLOPS (10^15 floating point operations per second) of computational power. This is provided by hundreds to thousands of very expensive specialized GPUs, each costing in the range of tens of thousands of dollars. These GPUs can also need to run for up to several months in order to fully train such an LLM, and this incurs further tens to hundreds of thousands of dollars in electricity prices. On top of that, LLMs need terabytes of high-quality training data, which can be hard to procure and, of course, is very expensive.

Overall, LLMs need enormous amounts of resources to train, which is also reflected in requirements to optimize them using strategies like fine-tuning or reinforcement learning, where you are essentially just training the model more.

SLMs, on the other hand, are much more efficient in almost every way. While their performance is limited due to their smaller scale, the training requirements are equally reduced. SLMs can require thousands of times less computing power and can even be trained on the average laptop/desktop graphics card instead of specialized hardware. They also train much faster, even with this very reduced amount of computing power, and they take on the scale of weeks or days rather than months. They also require much less data to train, being on the scale of gigabytes rather than terabytes or even petabytes. All of this makes SLMs much easier to train and fine-tune. This leads to one of SLMs' main strengths over LLMs in terms of utility: they can be easily and quickly custom-trained to work well on one specific task, and cost thousands of times less.

■ Optimization Methods

3.1. : Fine Tuning:

Fine-tuning is a technique usable to optimize any kind of AI model. This technique involves taking a smaller, more specialized dataset and using it to further train a model to be better at a specific task. This technique was brought to the field of NLP by J. Howard & S. Ruder in 2018,[21] who took the concept from CV (computer vision) modeling and modified it to work with NLP models. Fine-tuning optimizes NLPs by focusing them on a particular set of data. Specifically for reasoning, it is done with data sets containing reasoning-related question sets and their answers. It can also be done with a chain of thought, examples like the ones given in CoT prompting.

Fine-tuning is a well-established method of focusing a model on a particular task or subject, and it is also much more efficient to apply to SLMs. This is because SLMs have many fewer parameters and are usually trained with less available data. Fine-tuning allows SLMs to perform comparably to much larger models using significantly less data. Some examples of fine tuning methods that are better for SLMs are: fine tuning only bias terms,[22] which involves only fine-tuning a small subset of a model's parameters for increased efficiency; Distillation,[23] which is a technique in its own right, involving the transfer of training from larger models to small models; and even fine tuning for multiple different tasks,[24] which involves fine tuning using more than one specialized dataset to allow fine-tuned models to not be as restricted to a single task. Fine-tuning is an excellent, data-efficient technique to optimize a Small Language Model's performance in any field.

3.2. : Prompt engineering:

Prompt engineering is a very broad term that can be used to describe any technique to optimize a model by modifying the input sequence. Whether it be providing examples of inputs and outputs to acclimate a model to a particular question type,[25,26] instructing a model to approach a question in a particular way, such as breaking a problem into simpler steps,[27] or telling the model to break up the problem on its own,[28] providing examples of how to break down a question into multiple steps.[29] A couple of other particularly important sections of prompt engineering include zero-shot prompt engineering and automated prompt engineering. Zero-shot prompt engineering aims to guide the model to better utilize the information it has from training, and can be achieved through simply telling the model to take the problem step-by-step,[30] telling the model to take a deep breath,[31] and even appealing emotionally.[32] Automated prompt engineering involves using a model to optimize prompts for you, whether it be automatically generated CoT and few-shot learning examples,[33,34] automatically generating questions that models can understand well,[35] or distilling necessary context in a prompt by removing distracting information.[36] In this paper, we will only go over a few of the many, many ways to engineer prompts.

In -Context Learning:

In-context learning is a prompt engineering technique that can be applied to almost any type of model and any type of task. It is done by simply giving the model examples of prompts and the desired answer. This can be done either manually with human-generated examples or with examples generated by another model. T. Brown *et al*. designed GPT-3 to test this technique in the realm of NLP,[25] and was able to beat the at-the-time SoTA model on the LAMBADA benchmark (Turing-NLG with 17B params at 68%) by about 10% accuracy with just 2.7B of its own parameters, advancing this

to an 18% increase in accuracy with their 175B parameter version. This does show that the benefits of ICL do not scale linearly with size, as an almost 70-fold increase in parameters only improved the performance of the model by roughly 8%. On the other hand, S. Min *et al.* investigated the question: "Why does in-context learning work?"[26] They discovered that, bizarrely, the labels assigned to the demonstrations don't matter that much because when randomizing them, the performance was barely affected. They found that most of the benefits of ICL come from the example prompts and the example answers analyzed separately rather than paired, so as long as the example-answer sets are the same, you can pair them up in any fashion and have only a marginal reduction of benefits. In-Context Learning has an interesting effect when the size of the model is considered. Z. Shi *et al.* sought to find out how LLMs and SLMs process ICL examples differently,[37] and concluded that while they both have their skills, LLMs perform better due to their increased overall comprehension of the examples. However, SLMs were shown to notice smaller, more hidden trends in the examples that LLMs missed. This is a promising feature of them that could be used in conjunction with LLMs to improve overall performance.

Instruction-Based Prompting:

Instruction-based prompting is a prompt engineering technique based on giving a model a prompt that tells it to approach a problem in a specific way. One particular technique is plan and solve prompting,[28] which involves telling the model to make a step-by-step plan to solve the problem before actually following through on that plan. This strategy regularly beat standard zero-shot CoT ("think about it step by step"), and even barely outperformed manual few-shot CoT on the SingleEq, AddSub, and GSM8K benchmarks. Another technique is instructional prompt reframing.[27] This technique is centered around rewriting prompts to be better understood by the particular model(s) by breaking the problem down into steps, using more specific language, breaking up answer criteria into multiple instructions, turning 'don'ts' into 'dos' (NLP models struggle with negative commands), and adding restrictions to ensure a proper output format. In their experiment, they did this with the GPT models in mind and were able to improve GPT3's performance on several datasets by an average of 17% and 8% over the raw prompt for zero-shot and few-shot trials, respectively.

Chain of Thought Prompting:

Chain of Thought Prompting, or CoT prompting, is an umbrella term referring to any prompt engineering technique aimed at getting the model to break down a task into many steps. This technique is broken down into two different kinds. Few-shot CoT prompting is the first kind of CoT prompting. Developed by J. Wei *et al.*,[29] this version of CoT prompting involves giving the model a few example questions paired with the answers and the chain of thought used to get from the context to the correct solution. Their experimental results found a particular improvement of the GSM8K problem set with models like GPT-3, Codex, and PaLM-540B, having an increase of 31.3%, 43.4%, and 39% respectively. However, when tested on smaller models such as LaMDA 420M and GPT-3 350M, CoT prompting was found to have a negative effect on their performance, dropping their GSM8K accuracy rates by more than a factor of 4. This would suggest that CoT prompting has little to offer in the field of SLM optimization, but other studies have demonstrated potential. L. Ranaldi & A. Freitas use a method they call Instruction-Tuned- CoT, which involves using a teacher model to demonstrate chain-of-thought reasoning to SLMs.[38] They observed improvements of 5-10% on various benchmarks when using a teacher model of GPT-3.5 on the Llama-2 7B and Llama-2 13B models. While these improvements are not as significant as the ones that CoT has shown to have on LLMs, they show promise in the application of this idea to SLMs.

Zero -Shot Prompt Engineering:

Zero-shot Prompt Engineering is the field of prompt engineering surrounding zero-shot prompts, which are prompts that do not include any examples and require the models to answer solely with their training. One example of zero-shot prompt engineering is zero-shot CoT prompting. This technique, developed by

T. Kojima *et al.*,[30] involves prompting a model to take a problem step by step without any examples. They utilized the phrase "let's think step by step" to induce this behavior, and their model showed a significant improvement on both the MultiArith and GSM8K datasets (17.7-78.7% and 10.4-40.7% respectively, from zero-shot to zero-shot CoT). Zero-shot prompting can also be achieved in some strange ways, such as in a study done by Yang *et al.*.[31] In this study, they utilized LLMs as tools to generate optimized zero-shot prompts, and the best prompt phrase that their method generated was the phrase "Take a deep breath and work on this problem step-by-step." This prompt achieved an 80.2% accuracy on the GSM8K dataset on a pre-trained PaLM 2-L model. This is a significant improvement compared to just the question, which measured a 34% accuracy on the same dataset and model. Another of these strange zero-shot prompt engineering techniques is emotional prompting. C. Li *et al.* propose that LLMs can be boosted through emotionally charged prompts.[32] Their techniques induce self-monitoring behavior by either asking the model if it is sure, or providing encouragement with phrases usually associated with boosting self-esteem i.e., "Remember that progress is made one step at a time. Stay determined and keep moving forward." This study aimed to learn how LLMs process these kinds of human encouragement, and they tested their prompts on the TruthfulQA set, which is measured by both truthfulness and informativeness of a generated text. Their best result was using ChatGPT, and this was an improvement from 75% true, 53% informative, to 87% true and 67% informative.

Automated Prompt Engineering:

Automated prompt engineering is the field of prompt engineering that involves the utilization of a model in order to optimize prompts. For example, Yang *et al.* discuss the utilization of LLMs to optimize zero-shot prompts.[31] Their best

DOI: 10.36838/IJHSR85.76

generated prompt was "Take a deep breath and work on this problem step-by-step," which gave them an 80.2% accuracy on the GSM8K benchmark, a massive improvement from the baseline for no modifications, which was 34% accurate. Another couple of papers that demonstrate automatic prompt engineering are those by Z. Zhang *et al*. and A. Sevinc & A. Gumus.[33,34] These papers focus on the automatic generation of CoT prompts. Z. Zhang *et al*. achieved results comparable to manually generated CoT using examples generated by another LLM using a "let's think step by step" (zero-shot CoT) prompt. A. Sevinc & A. Gumus use a strategy that uses rationales generated by a better model (GPT-4) to improve the performance of a weaker/smaller model.[33] They observed an increase of 20% accuracy when tested using GPT-4 on the StrategyQA benchmark. Further automatic prompt engineering techniques include "System 2 Attention" or S2A by J. Weston & S. Sukhbaatar, which is a technique that utilizes a model to remove any distracting information and optimize a prompt for itself.[36] They found that their method could successfully remove distracting details from the GSM-IC dataset, which has problems with irrelevant and distracting sentences included within it. Their method successfully got the accuracy of the prompted model to be just about the same as that of the oracle prompt (a prompt for the question without the distractions). A final method of automatic prompt engineering is presented by J. Weston & S. Sukhbaatar.[36] They propose another method of generating high-quality prompts for any task by utilizing a masked language model (MLM), which is a model that is trained by 'masking' or covering up words in a text, and having the model predict them. These models specialize in filling in spots in a given text. Their study suggests that this method could be a replacement for fine-tuning in certain situations. They tested it on the GLUE benchmark using the BERT and RoBERTa models, achieving a 19.1% and 6.2% improvement, respectively.

3.3. : Test -Time Compute:

Test-time compute is an optimization field that involves the reallocation of resources during inference. This allows models to respond much more accurately to difficult questions at the cost of inference time. This technique is very good at optimizing SLMs, due to the fact that it allows SLMs to take a long time to answer questions to a degree of accuracy higher than some LLMs. One example of this utilization is demonstrated by E. Akyürek in abstract reasoning.[39] They use the benchmark of the Abstraction and Reasoning Corpus (ARC), and tested Llama-3 8B and Llama-3.2 1B, and 2B. Their findings are promising, as the 1B and 3B models solved 6x and 3x as many tasks, respectively, when TTT (test-time-training) was enabled. While the 8B Llama model performed well, TTT proved to be effective at improving the 1B and 3B parameter models to a level competitive with the 8B model. The downside to test-time compute is the extended inference time, which negates one of the advantages of SLMs: fast inference time. M. Alfarra *et al*. aim to combat this longer inference time by promoting methods that take less time by giving them more encouragement through data.[40]

■ Compression Methods

4.1. : Knowledge Distillation:

Knowledge Distillation is a technique that transfers knowledge and skills from one model to another model.[41-43] There are three different techniques that are used in distillation: offline distillation, online distillation, and self-distillation. Offline distillation is the traditional distillation, which involves a large, pre-trained teacher model teaching a much smaller student model.[44] It is a good method to train SLMs that can run on much more constrained devices while retaining a good amount of the teacher model's performance. Online distillation is a different approach, where both the teacher and student models are learning at the same time.[45] This is best when a pre-trained teacher model is not easily accessible. Self-distillation is a very drastically different technique where an already somewhat trained model trains itself and uses its deeper layers to improve its shallower layers.[46] This technique is best for optimizing a single model with little performance sacrificed when you have even more of a resource constraint.

One example of a knowledge distillation technique is Spot-adaptive Knowledge Distillation by J. Song *et al*..[47] Their work demonstrates a novel technique that adaptively chooses layers to learn from. Traditional methods typically use a manually selected set of layers to use in distillation, but this can be very sub-optimal in circumstances that are not ideal. They demonstrated their method in CV benchmarks and recorded small but consistent improvements across various model sizes and benchmarks. Another example of a knowledge distillation technique is shown in the paper "Heterogeneous Knowledge Distillation Using Conceptual Learning" by Y. Yu & N. Kim.[48] Their technique involves teaching the smaller model higher-level concepts in the field of knowledge, rather than just the information required to solve the target problems. This has the benefit of allowing the distilled model to better generalize to a broader group of related tasks. They tested this on the 20 Newsgroups dataset, and, similarly to the other study, showed a small but consistent advantage over traditional knowledge distillation techniques. While these adaptations of distillation show relatively minor improvements over traditional distillation, traditional distillation is incredibly efficient at compressing the model's size, as shown by A. Rashid *et al*., with a recorded up to 30x reduction in size while only losing 8-25% of the original model's accuracy.[49]

The further development of distillation techniques is necessary because, while the improvements made over traditional distillation are small, distillation is still one of the best ways to provide close to LLM-level performance with a fraction of the required computational power.

4.2. : Pruning:

Pruning is a technique that involves carefully removing less important parts of models in order to reduce their operating requirements while not affecting performance too much. Pruning is separated into two main categories: structured pruning and unstructured pruning. Structured pruning is the process of removing large clusters of weights in a model, and unstructured pruning is the process of removing individual weights in

a model. These processes all aim to increase a measurement called sparsity, which is essentially the percentage of weights that are set to 0. This greatly reduces the required computing power as a large amount of calculations can be essentially skipped. The process typically targets weights that are close to zero already, so the model does not lose very much accuracy. One example of such is shown by R. Xu *et al.*, who developed a framework that utilizes the supervision of other models during the pruning process.[50] They achieve a 90% sparsity using structured pruning, and 97% sparsity using unstructured pruning.

They only lose 10.5-.7% and 7-.7% accuracy on various datasets such as QQP and SST-2 using structured and unstructured versions of their technique, respectively. Another example is in a paper by J.- H. Luo *et al.*, which demonstrates another framework for pruning that achieved only a 10% decrease in accuracy on the Indoor-67 dataset while achieving over 100-fold decrease in parameters.[51] While one of these studies is on computer vision deep neural networks rather than NLP models, they both demonstrate how pruning can decrease model sizes by orders of magnitude while still retaining much of their accuracy.

■ Discussion and Future Directions

5.1. : Prompt engineering

While prompt engineering techniques can provide little or even negative effects on SLMs, the core principles behind the field of prompt engineering can be implemented to improve SLM performance. The idea of Chain of Thought reasoning has been demonstrated to be teachable to smaller models,[37] and further developments in how prompts can be engineered to more effectively work on smaller models would make a massive impact on people's ability to utilize AI in small-scale settings.

5.2. : Fine Tuning:

Fine-tuning is a very well-used technique in SLMs, one of SLMs' main strengths being the ease of fine-tuning. This technique is generally beneficial, and important fields of improvement would be the versatility of fine-tuned models, the further increase of fine-tuning's effectiveness, and the decrease of computational requirements for fine-tuning. Fine-tuning is already a very effective and widely used technique to optimize SLMs for commercial usage, and further improvements are in progress as this is a more prevalent technique. Therefore, even more research into this field is likely not required, and efforts should be concentrated on newer, less studied techniques.

5.3. : Test Time Compute:

Test Time Compute is a technique in the field of NLP that has recently gained more widespread prevalence primarily due to OpenAI's o1 model line, and is overall a technique with much room for development. It is a field that needs much more research than the others due to its massive potential to make smaller models as good as larger models, and larger models even better. Of course, this technique is receiving the amount of attention it needs in the research space, but looking into applications and optimization for applying this technique on smaller models is somewhat less popular, and developments in that specific field would massively increase the accessibility of high-performance AI models, which can combine the strengths of both LLMs and SLMs.

5.4. : Distillation:

Distillation is a relatively well-established technique, and therefore has extensive existing research. Additionally, its sole purpose is to optimize and compress, optimizing a large model by compressing a larger model's knowledge into it. For our purposes, that makes any research into distillation beneficial towards the accessibility of AI models. However, it has become clear that most improvements upon distillation techniques are near marginal, only improving performance by a couple of percentage points. Therefore, a large breakthrough in the field is required before we make significant progress.

5.5. : Pruning:

Pruning is another extremely effective technique, and is well established as well. It is very similar to distillation in relevance to our goal of accessibility and research status. It also has much existing research, but also only minor improvements are being made over the original technique. Similarly to distillation, a massive breakthrough needs to be made in this field, as current performance returns on research invested are low.

■ Methodology and Limitations

This survey was primarily conducted using Google Scholar with the key phrases of "NLP Compression", "SLM Optimization", and the names of the specific sections of the 'Techniques' section. The exploration of relevant citations was also utilized. This study was limited by a strict time limit, and due to this, it was heavily focused on well-established and cited papers. This reflects the main purpose of this paper, which is not to provide a comprehensive review, but to incite interest in this field as it is of great importance.

■ Conclusion

In this survey, we have gone over various methods that can be used to optimize small NLP models, as well as methods that can compress larger ones. This is done to focus on the practical utilization of NLP models in real-life applications. The development of these techniques is incredibly important for AI deployment in a small-scale setting. The increased accessibility of NLP models and AI in general for the general public is corresponding with increased public interest and investment in further development of AI. Therefore, it is just as important to work on smaller, more accessible models for public use as it is to work on massive, state-of-the-art models because this development of public AI is crucial to foster interest in the field of AI and eventually will lead to an increase in research and funding in the field as a whole.

Increasing accessibility of these models would also boost efficiency and improve the quality of life for countless small businesses and organizations.

DOI: 10.36838/IJHSR85.76

Acknowledgments

I would like to thank Professor Siddarth Krishnan and Dr. Plinio Zanini for their mentoring during the process of writing this paper. Thanks to them, my own view of research as a whole has been completely changed.

Citiations

1. Huang, J.; Chang, K. C.-C. Towards Reasoning in Large Language Models: A Survey. arXiv May 26, 2023. https://doi.org/10.48550/arXiv.2212.10403.
2. Lewis, M.; Mitchell, M. Evaluating the Robustness of Analogical Reasoning in Large Language Models. arXiv November 21, 2024. https://doi.org/10.48550/arXiv.2411.14215.
3. Kambhampati, S. Can Large Language Models Reason and Plan? *Annals of the New York Academy of Sciences* 2024, *1534* (1), 15–18. https://doi.org/10.1111/nyas.15125.
4. Valmeekam, K.; Olmo, A.; Sreedharan, S.; Kambhampati, S. Large Language Models Still Can't Plan (A Benchmark for LLMs on Planning and Reasoning about Change).
5. Cerf, V. G. Large Language Models. *Commun. ACM* **2023**, *66* (8), 7–7. https://doi.org/ 10.1145/3606337.
6. Dasgupta, I.; Lampinen, A. K.; Chan, S. C. Y.; Sheahan, H. R.; Creswell, A.; Kumaran, D.; McClelland, J. L.; Hill, F. Language Models Show Human-like Content Effects on Reasoning Tasks. arXiv July 17, 2024. https://doi.org/10.48550/arXiv.2207.07051.
7. Xu, B.; Poo, M. Large Language Models and Brain-Inspired General Intelligence. *National Science Review* **2023**, *10* (10), nwad267. https://doi.org/10.1093/nsr/nwad267.
8. Zhang, Z.; Zhang, A.; Li, M.; Zhao, H.; Karypis, G.; Smola, A. Multimodal Chain-of-Thought Reasoning in Language Models. arXiv May 20, 2024. https://doi.org/10.48550/arXiv.2302.00923.
9. Jiang, F.; Jiang, Y.; Zhi, H.; Dong, Y.; Li, H.; Ma, S.; Wang, Y.; Dong, Q.; Shen, H.; Wang, Y. Artificial Intelligence in Healthcare: Past, Present and Future. *Stroke Vasc Neurol* **2017**, *2* (4), 230–243. https:// doi.org/10.1136/svn-2017-000101.
10. Cao, L. AI in Finance: Challenges, Techniques and Opportunities. **2021**, *1* (1).
11. Barenkamp, M.; Rebstadt, J.; Thomas, O. Applications of AI in Classical Software Engineering. *AI Perspectives* **2020**, *2* (1), 1. https://doi.org/10.1186/s42467-020-00005-4.
12. Minaee, S.; Mikolov, T.; Nikzad, N.; Chenaghlu, M.; Socher, R.; Amatriain, X.; Gao, J. Large Language Models: A Survey. arXiv February 20, 2024. https://doi.org/10.48550/arXiv.2402.06196.
13. Contreras Kallens, P.; Kristensen-McLachlan, R. D.; Christiansen, M. H. Large Language Models Demonstrate the Potential of Statistical Learning in Language. *Cognitive Science* **2023**, *47* (3), e13256. https://doi.org/10.1111/cogs.13256.
14. Wu, L.; Zheng, Z.; Qiu, Z.; Wang, H.; Gu, H.; Shen, T.; Qin, C.; Zhu, C.; Zhu, H.; Liu, Q.; Xiong, H.; Chen, E. A Survey on Large Language Models for Recommendation. arXiv June 18, 2024. https:// doi.org/10.48550/arXiv.2305.19860.
15. Zhao, W. X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; Du, Y.; Yang, C.; Chen, Y.; Chen, Z.; Jiang, J.; Ren, R.; Li, Y.; Tang, X.; Liu, Z.; Liu, P.; Nie, J.-Y.; Wen, J.-R. A Survey of Large Language Models. arXiv October 13, 2024. https://doi.org/10.48550/ arXiv.2303.18223.
16. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. arXiv August 2, 2023. https://doi.org/10.48550/arXiv.1706.03762.
17. Islam, S.; Elmekki, H.; Elsebai, A.; Bentahar, J.; Drawel, N.; Rjoub, G.; Pedrycz, W. A Comprehensive Survey on Applications of Transformers for Deep Learning Tasks. *Expert Systems with Applications* **2024**, *241*, 122666. https://doi.org/10.1016/j.eswa.2023.122666.
18. Bird, J. J.; Ekárt, A.; Faria, D. R. Chatbot Interaction with Artificial Intelligence: Human Data Augmentation with T5 and Language Transformer Ensemble for Text Classification. *J Ambient Intell Human Comput* **2023**, *14* (4), 3129–3144. https://doi.org/10.1007/s12652-021-03439-8.
19. Bawden, R.; Yvon, F. Investigating the Translation Performance of a Large Multilingual Language Model: The Case of BLOOM. arXiv May 9, 2023. https://doi.org/10.48550/arXiv.2303.01911.
20. Basyal, L.; Sanghvi, M. Text Summarization Using Large Language Models: A Comparative Study of MPT-7b-Instruct, Falcon-7b-Instruct, and OpenAI Chat-GPT Models. arXiv October 17, 2023. https:// doi.org/10.48550/arXiv.2310.10449.
21. Howard, J.; Ruder, S. Universal Language Model Fine-Tuning for Text Classification. arXiv May 23, 2018. https://doi.org/10.48550/arXiv.1801.06146.
22. Ben Zaken, E.; Goldberg, Y.; Ravfogel, S. BitFit: Simple Parameter-Efficient Fine-Tuning for Transformer-Based Masked Language-Models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*; Association for Computational Linguistics: Dublin, Ireland, 2022; pp 1–9. https://doi.org/10.18653/v1/2022.acl-short.1.
23. Hsieh, C.-Y.; Li, C.-L.; Yeh, C.-K.; Nakhost, H.; Fujii, Y.; Ratner, A.; Krishna, R.; Lee, C.-Y.; Pfister, T. Distilling Step-by-Step! Outperforming Larger Language Models with Less Training Data and Smaller Model Sizes. arXiv July 5, 2023. https://doi.org/10.48550/arXiv.2305.02301.
24. Chung, H. W.; Hou, L.; Longpre, S.; Zoph, B.; Tay, Y.; Fedus, W.; Li, Y.; Wang, X.; Dehghani, M.; Brahma, S.; Webson, A.; Gu, S. S.; Dai, Z.; Suzgun, M.; Chen, X.; Chowdhery, A.; Castro-Ros, A.; Pellat, M.; Robinson, K.; Valter, D.; Narang, S.; Mishra, G.; Yu, A.; Zhao, V.; Huang, Y.; Dai, A.; Yu, H.; Petrov, S.; Chi, E. H.; Dean, J.; Devlin, J.; Roberts, A.; Zhou, D.; Le, Q. V.; Wei, J. Scaling Instruction- Finetuned Language Models. arXiv December 6, 2022. https://doi.org/10.48550/arXiv.2210.11416.
25. Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; Amodei, D. Language Models Are Few-Shot Learners. arXiv July 22, 2020. https://doi.org/10.48550/arXiv.2005.14165.
26. Min, S.; Lyu, X.; Holtzman, A.; Artetxe, M.; Lewis, M.; Hajishirzi, H.; Zettlemoyer, L. Rethinking the Role of Demonstrations: What Makes In-Context Learning Work? arXiv October 20, 2022. https:// doi.org/10.48550/arXiv.2202.12837.
27. Mishra, S.; Khashabi, D.; Baral, C.; Choi, Y.; Hajishirzi, H. Reframing Instructional Prompts to GPTk's Language. In *Findings of the Association for Computational Linguistics*: ACL 2022; Association for Computational Linguistics: Dublin, Ireland, 2022; pp 589–612. https://doi.org/10.18653/ v1/2022.findings-acl.50.
28. Wang, L.; Xu, W.; Lan, Y.; Hu, Z.; Lan, Y.; Lee, R. K.-W.; Lim, E.-P. Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models. arXiv May 26, 2023. https://doi.org/10.48550/arXiv.2305.04091.
29. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; Zhou, D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv January 10, 2023. https:// doi.org/10.48550/arXiv.2201.11903.

30. Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; Iwasawa, Y. Large Language Models Are Zero-Shot Reasoners. arXiv January 29, 2023. https://doi.org/10.48550/arXiv.2205.11916.
31. Yang, C.; Wang, X.; Lu, Y.; Liu, H.; Le, Q. V.; Zhou, D.; Chen, X. Large Language Models as Optimizers. arXiv April 15, 2024. https://doi.org/10.48550/arXiv.2309.03409.
32. Li, C.; Wang, J.; Zhang, Y.; Zhu, K.; Hou, W.; Lian, J.; Luo, F.; Yang, Q.; Xie, X. Large Language Models Understand and Can Be Enhanced by Emotional Stimuli. arXiv November 12, 2023. https:// doi.org/10.48550/arXiv.2307.11760.
33. Sevinc, A.; Gumus, A. AutoReason: Automatic Few-Shot Reasoning Decomposition. arXiv December 9, 2024. https://doi.org/10.48550/arXiv.2412.06975.
34. Zhang, Z.; Zhang, A.; Li, M.; Smola, A. Automatic Chain of Thought Prompting in Large Language Models. arXiv October 7, 2022. https://doi.org/10.48550/arXiv.2210.03493.
35. Zhou, Y.; Muresanu, A. I.; Han, Z.; Paster, K.; Pitis, S.; Chan, H.; Ba, J. Large Language Models Are Human-Level Prompt Engineers. arXiv March 10, 2023. https://doi.org/10.48550/arXiv.2211.01910.
36. Weston, J.; Sukhbaatar, S. System 2 Attention (Is Something You Might Need Too). arXiv November 20, 2023. https://doi.org/10.48550/arXiv.2311.11829.
37. Shi, Z.; Wei, J.; Xu, Z.; Liang, Y. Why Larger Language Models Do In-Context Learning Differently? arXiv May 30, 2024. https://doi.org/10.48550/arXiv.2405.19592.
38. Ranaldi, L.; Freitas, A. Aligning Large and Small Language Models via Chain-of-Thought Reasoning. **2024**.
39. Akyürek, E. The Surprising Effectiveness of Test-Time Training for Abstract Reasoning.
40. Alfarra, M.; Itani, H.; Pardo, A.; Alhuwaider, S.; Ramazanova, M.; Pérez, J. C.; Cai, Z.; Müller, M.; Ghanem, B. Evaluation of Test-Time Adaptation Under Computational Time Constraints. arXiv May 23, 2024. https://doi.org/10.48550/arXiv.2304.04795.
41. Gu, Y.; Dong, L.; Wei, F.; Huang, M. MiniLLM: Knowledge Distillation of Large Language Models. arXiv April 10, 2024. https://doi.org/10.48550/arXiv.2306.08543.
42. Shridhar, K.; Stolfo, A.; Sachan, M. Distilling Reasoning Capabilities into Smaller Language Models. In *Findings of the Association for Computational Linguistics: ACL 2023*; Association for Computational Linguistics: Toronto, Canada, 2023; pp 7059–7073. https://doi.org/10.18653/v1/2023.findings-acl.441.
43. West, P.; Bhagavatula, C.; Hessel, J.; Hwang, J.; Jiang, L.; Le Bras, R.; Lu, X.; Welleck, S.; Choi, Y. Symbolic Knowledge Distillation: From General Language Models to Commonsense Models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*; Association for Computational Linguistics: Seattle, United States, 2022; pp 4602–4625. https://doi.org/10.18653/v1/2022.naacl-main.341.
44. Furlanello, T.; Lipton, Z. C.; Tschannen, M.; Itti, L.; Anandkumar, A. Born Again Neural Networks. arXiv June 29, 2018. https://doi.org/10.48550/arXiv.1805.04770.
45. Guo, Q.; Wang, X.; Wu, Y.; Yu, Z.; Liang, D.; Hu, X.; Luo, P. Online Knowledge Distillation via Collaborative Learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE: Seattle, WA, USA, 2020; pp 11017– 11026. https://doi.org/10.1109/ CVPR42600.2020.01103.
46. Handong Global University, Pohang, South Korea; Hahn, S.; Choi, H. Self-Knowledge Distillation in Natural Language Processing. In *Proceedings - Natural Language Processing in a Deep Learning World*; IncomaLtd., Shoumen, Bulgaria, 2019; pp 423-430. https://doi.org/ 10.26615/978-954-452-056-4_050.
47. Song, J.; Chen, Y.; Ye, J.; Song, M. Spot-Adaptive Knowledge Distillation. *IEEE Trans. on Image Process*. **2022**, *31*, 3359–3370. https://doi.org/10.1109/TIP.2022.3170728.
48. Yu, Y.; Kim, N. Heterogeneous Knowledge Distillation Using Conceptual Learning. *IEEE Access* **2024**, *12*, 52803–52814. https://doi.org/10.1109/ACCESS.2024.3387459.
49. Rashid, A.; Lioutas, V.; Ghaddar, A.; Rezagholizadeh, M. Towards Zero-Shot Knowledge Distillation for Natural Language Processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*; Association for Computational Linguistics: Online and Punta Cana, Dominican Republic, 2021; pp 6551–6561. https://doi.org/10.18653/v1/2021.emnlp-main.526.
50. Xu, R.; Luo, F.; Wang, C.; Chang, B.; Huang, J.; Huang, S.; Huang, F. From Dense to Sparse: Contrastive Pruning for Better Pre-Trained Language Model Compression. arXiv December 14, 2021. https://doi.org/10.48550/arXiv.2112.07198.
51. Luo, J.-H.; Wu, J.; Lin, W. ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression. arXiv July 20, 2017. https://doi.org/10.48550/arXiv.1707.06342.

Author

Nathan Lam is a Junior at Stuyvesant High School in New York, New York, at the time of writing. He is passionate about all things engineering, and he aspires to integrate Artificial Intelligence and Machine Learning into any project where it may improve it.

DOI: 10.36838/IJHSR85.76