Improved Methods for AI Agent Pruning

Anonymous Authors[1]

Abstract

Pruning techniques have evolved significantly since their inception in 2014, with applications ranging from enhancing search engines to optimizing large models by 2024. Our paper focuses on introducing a novel concept of agent pruning that targets their core working spaces, namely perception, planning, reasoning, and action space. Given the vast amount of potential plans that an AI agent can generate, our method selectively eliminates redundant or suboptimal plans fully or partially, enabling the whole decision-making cycle to be faster and more stable. By identifying and pruning elements that do not contribute significantly to the high-level application-specific goal, we streamline the whole process without sacrificing the agent's rational ability to interact with the physical world. With the global presence of billions of AI agents in the future, optimizing their working spaces through pruning is essential. Thus, we propose 101 agent pruning, aiming and hoping to enhance agent performance, minimize resource consumption, and unlock new capabilities for them.

1. Introduction

For over a decade, pruning has stood as a remarkable optimization technique with a rich history (Suel et al., 2016) (Dean, 2014). Dating back to around 2014, when search engines like Google were soaring in popularity, pruning played a pivotal role. Data centers were grappling with the ever expanding volume of data, and pruning became the key to maintaining high quality service while slashing costs. By selectively removing redundant data, search engines could operate more efficiently, reducing storage requirements and speeding up query responses. A plethora of academic papers have explored the ins and outs of inverted index pruning, from simple heuristic based methods to complex machine learned algorithms. Incidentally, the technique of tiering (Rodriguez, 2021) also emerged in this context, further refining the search engine optimization domain.

Fast forward to around 2024, the landscape of pruning had evolved significantly. With the advent of popular app such as chatbots based mostly on language models (Goodfellow et al., 2016) (OpenAI, 2023), pruning techniques were ingeniously adapted to transfer the knowledge of these large, resource hungry models into smaller, more manageable counterparts. This technique not only made these models more accessible for deployment on a wider range of devices but also improved their computational efficiency. By carefully pruning away less important connections and neurons from deep neural network (Sun et al., 2024) (Frantar & Alistarh, 2023), the smaller models could still capture the essence of the larger ones, achieving comparable performance with far fewer computation resources (Hinton et al., 2015).

As technology continues to evolve, we are approaching a future where agents are not just confined to virtual environments but are embodied in physical ones (Russell & Norvig, 2020), equipped with sensory inputs and capable of complex tasks. These agents, with their brain operating on advanced models (He, 2024), will face a significant challenge: managing vast, multidimensional spaces. Specifically, these agents will need to generate, and navigate perception, planning, reasoning, and action spaces that can become overwhelmingly large, consisting of billions of potential states and actions. To formalize this challenge, we propose mathematical models and related pruning technique for each space and state clearly the research and engineering problems we face for future improvement. For example, the planning space is designed to include the sequences of actions required to achieve goals, while the reasoning space enables agents to decide the most efficient or relevant path based on context, and the action space defines the specific actions an agent can take and perform. By representing these spaces as seperated but interconnected graphs, we might be able to apply pruning techniques to smooth the whole decision making process. For instance, we can prune the reasoning space by eliminating nodes that do NOT contribute much to the agent's central goal, thus we can create shortcuts that lead to more efficient problem solving capabilities. Likewise, pruning the action

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

space reduces unnecessary or redundant actions, enabling the agent to act faster, and potentially more effectively and efficiently. Under our assumptions and proposed spaces, an AI agent can perform simple or complex tasks either in sequential or in parallel manner, transitioning smoothly from one state to another, and achieving their objectives with minimized resource consumption and enhanced adaptability in general. In order to make our readers easy to understand, we intentionally give one concrete example on how to construct an intelligent agent for fashion, the task is mostly on capturing the moment of beauties for a fashion clothing house, such as mix-and-match of the top and bottom, beautiful and smiling faces among our customers etc (W et al., 2025).

2. The Proposed Conceptual Model

Our proposed conceptual model for AI agent is structured around two fundamental components: the environment and the agent itself. The environment within which the agent operates is in a constant state of flux. This dynamic nature necessitates that the agent adapt continuously, leveraging its perceptual capabilities and taking appropriate actions. The agent, on its part, is composed of four key spaces: the perception space, the planning space, the reasoning space, and the action space. These four spaces construct the 'see and act' loop in the context of AI agents in the physical world.

2.1. Define the Perception Space

The perception space acts as the gateway through which the agent receives information from the environment. A multitude of signals, such as visual signals captured by cameras, speech signals picked up by microphones, and potentially other sensory inputs, stream into the agent's "mind". These signals are processed to build an understanding of the environment's state, including the presence of objects, the actions of other entities, and the overall context. For instance, visual signals can help the agent identify the layout of a room, the position of obstacles, or the appearance of specific items. Speech signals might convey instructions from a human operator or provide information about ongoing events. The more comprehensive the perception, the more detailed the agent's understanding of the world becomes.

2.2. Define the Planning Space

Once the agent has gathered information via the perception space, it moves on to the planning space. Here, the agent formulates strategies to interact with the environment. Plans can range from high level objectives, like completing a complex manufacturing task or navigating to a distant location. These plans are designed to achieve the agent's goals while taking into account the information gleaned from the perception space. However, due to limited computational power, as well as security, safety, and privacy concerns, the number of plans generated needs to be carefully managed. An overly expansive planning space could lead to resource exhaustion, potential security vulnerabilities, and a lack of focus on the most important task.

2.3. Define the Reasoning Space

In the reasoning space, the agent evaluates the plans formulated in the planning space. It uses logical and data-driven reasoning to rank and select the most appropriate plans. This involves considering factors such as the likelihood of success, the resources required, and the potential impact on the environment and other entities. For example, a plan that requires excessive computational resources or poses a risk to safety might be downgraded in the reasoning process. Similar to the planning space, the reasoning space also needs to be pruned to ensure efficient decision making. Unnecessary or sub-optimal lines of reasoning can waste valuable resources and time.

2.4. Define the Action Space

The action space is where the agent translates its plans into real world interactions with the environment. The actions can be guided by general and specific domain policies. For example, in a service oriented robotic task, there might be general policies regarding how to interact with humans in a polite and helpful manner, along with specific policies for handling particular tasks. For example, the actions might carry out the underlying principles of peace, love, and rationality in the brain of an AI agent. 'Peace' implies non-disruptive and harmonious interaction. 'Love' represents a careful and considerate approach, aiming to provide positive experiences. 'Rationality' ensures that actions are based on sound decision making processes, considering both the agent's goals and the well being of all involved entities. Given the physical and computational constraints, as well as the need for safe and private interactions, the action space also requires pruning to ensure that only the most necessary and appropriate actions are executed.

3. The Proposed Mathematical Model

This section presents in detail the mathematical models that form the backbone of the AI agent, spanning the perception, planning, reasoning, and action aspects. These models are designed to facilitate effective interaction between the system and its environment, enabling it to achieve its intended objectives.

3.1. Model the Perception Space

The perception space, denoted as $\mathcal{S}$, serves as the gateway through which the AI agent gathers information about the

external world via various sensory inputs.

Visual Signals. Images are mathematically represented as matrices $I \in \mathbb{R}^{m \times n \times c}$, where m and n denote the height and width of the image respectively, and c stands for the number of color channels (e.g., $c = 3$ for the commonly used RGB format). Convolutional Neural Networks (CNNs) play a pivotal role in processing these visual data. A CNN layer l with k filters of size $s \times s$ is formulated as follows:

$$y_{ij}^{l} = \text{activation}\left(\sum_{u=0}^{s-1}\sum_{v=0}^{s-1}\sum_{c=0}^{c_{l-1}} w_{uvc}^{lk} x_{ij}^{l-1} + b^{lk}\right)$$

Here, x_{ij}^{l-1} represents the input feature map at layer $l-1$, y_{ij}^{l} is the output feature map at layer l. The elements w_{uvc}^{lk} are the weights of the k^{th} filter, and b^{lk} is the bias term. Commonly employed activation functions like ReLU (Rectified Linear Unit), sigmoid, and tanh introduce non-linearity to the network, enabling it to capture complex visual patterns. Pooling layers, such as max pooling and average pooling, are strategically used to reduce the dimensionality of the feature maps while retaining the most salient information.

For example, consider an AI agent task with navigating within a warehouse environment. The visual signals captured by the robot's onboard cameras are processed using CNNs. The initial convolutional layers extract low-level features like edges and textures. As the data passes through subsequent layers, more complex features related to objects, such as the shape of pallets or the presence of obstacles, are identified. Max pooling, for instance, helps in downsampling the feature maps after convolutional layers, effectively extracting the dominant features of objects. This enables the robot to quickly and accurately distinguish between different elements in its surroundings. The final output of the CNN can then be utilized for a variety of downstream tasks, including precise object detection for identifying specific items to interact with, semantic segmentation to understand the spatial layout of the environment, or generating a comprehensive scene feature vector that can inform navigation and manipulation actions.

Audio Signals. Audio waveforms are modeled as time series $s(t)$. To analyze these signals in the frequency domain, we apply the Fourier transform, resulting in $S(f) = \mathcal{F}\{s(t)\}$. Hidden Markov Models (HMMs) are then employed for general purpose speech recognition. An HMM is defined by several key components:

- States: A set of states $Q = \{q_1, q_2, \ldots, q_N\}$ that represent the different hidden states in the speech generation process. These states can be thought of as corresponding to different phonetic units or parts of words.
- Transition Probabilities: $A = \{a_{ij}\}$, where $a_{ij} = P(q_{t+1} = q_j | q_t = q_i)$ indicates the probability of transitioning from state q_i at time t to state q_j at time $t+1$. This captures the sequential nature of speech production.
- Emission Probabilities: $B = \{b_j(k)\}$, where $b_j(k) = P(O_t = v_k | q_t = q_j)$ represents the probability of observing a particular symbol v_k in the observation sequence when the system is in state q_j. These emission probabilities are often modeled using Gaussian Mixture Models (GMMs) to account for the inherent variability in speech sounds.
- Initial State Distribution: $\pi = \{\pi_i\}$, where $\pi_i = P(q_1 = q_i)$ gives the probability of starting in state q_i.

The Viterbi algorithm is utilized to find the most likely sequence of hidden states given an observed speech sequence. This is crucial as it helps in determining the most probable sequence of words that the user intended to say. Meanwhile, the Baum - Welch algorithm is commonly employed for training the HMM parameters, adjusting them based on a given set of speech data to optimize recognition accuracy.

In a practical scenario where a user issues voice commands to our AI agent, the speech signal is first transformed using the Fourier transform. Subsequently, relevant features, such as Mel-Frequency Cepstral Coefficients (MFCCs), are extracted from the frequency domain representation. These MFCCs serve as the observations for the HMM. By applying the Viterbi algorithm, the system can then determine the most likely sequence of words, thereby enabling it to understand and respond appropriately to the user's commands.

Multimodal Signals.From the start, the AI agent is designed to integrate information from multiple sensory modalities to form a more holistic and accurate understanding of the environment. We employ different strategies for this multimodal integration:

- **Early Fusion**. This approach involves concatenating raw features obtained from different modalities before feeding them into a learning model. For example, we might combine the feature vectors extracted from the processing of visual signals and speech signals at an early stage. A neural network can then be trained on this combined representation to learn the relationships between the different modalities and make predictions based on the integrated information.
- **Late Fusion**. In contrast, late fusion combines decisions or predictions made independently by individual modalities at a later stage. For instance, the system might first make separate predictions regarding the presence of an object using visual signals and the intent of a user's command from speech signals. Then,

a separate mechanism, such as a voting scheme or a weighted combination based on confidence scores, is used to combine these individual predictions to arrive at a final decision.

- **Attention Mechanisms**. Attention mechanisms are employed to dynamically assign weights to different modalities based on their relevance in a given context. For example, in a noisy environment where speech signals might be less reliable due to background noise, the attention mechanism could give more weight to visual cues to enhance the overall perception of the situation. This allows the system to adaptively focus on the most informative modality or combination of modalities in different circumstances.

3.2. Model the Planning Space

The planning space of the AI agent is structured using Hierarchical Task Networks (HTNs). Here, let T be the set of all tasks that the system is capable of performing. Each task $t \in T$ can be decomposed into a hierarchical structure of sub-tasks. Primitive tasks are those that can be directly executed by the system, while compound tasks require further decomposition into smaller, more manageable sub-tasks. This decomposition can be represented as a tree-like structure. For instance, a high-level task T_{high} might be decomposed as follows:

$$T_{high} = \{T_{mid1}, T_{mid2}, \ldots, T_{midm}\}$$

where each T_{midi} can be further broken down into lower-level tasks until only primitive tasks remain. This hierarchical decomposition provides a systematic way to break down complex tasks into simpler, actionable steps. To prioritize tasks within this framework, we define a cost function:

$$C(t) = \sum_i w_i f_i(t)$$

In this function, $f_i(t)$ represents various factors that contribute to the cost of a task. These factors include, but are not limited to, the resource requirements $R(t)$ (such as computational resources needed to execute the task, energy consumption, or memory usage), the estimated time $T(t)$ required to complete the task, the associated risk (e.g., the probability of failure or the potential negative impact on the system or the surrounding environment), and the importance of the task in achieving the overall system goals. The weights w_i are assigned to each factor to reflect their relative importance in the specific application context.

For example, in a manufacturing setting where the AI agent is mostly a robotic arm, a high-level task could be "assemble a product." This high-level task would be decomposed into mid-level tasks like "pick up component A," "position component A correctly," and so on. Each of these mid-level tasks would further break down into low-level tasks involving specific movements of the robotic arm joints. The cost function would then be used to prioritize these tasks based on factors such as the time it takes to perform each movement, the energy consumed during the operation, and the importance of getting each step right to ensure the successful assembly of the target product. HTN planning algorithms are employed to recursively decompose tasks based on the defined hierarchy and cost function until only primitive tasks are left. These algorithms also take into account various constraints, such as the availability of resources and the order in which tasks can be executed, to find a valid and efficient plan that can be implemented by the system.

3.3. Model the Reasoning Space

The reasoning space of an AI agent is modeled using mostly Bayesian networks, which are represented as directed acyclic graphs (DAGs) $G = (V, E)$. In this context, the nodes V represent random variables that are relevant to the system's decision-making process. These can include variables such as the task success probability, resource availability, environmental conditions, and other factors that influence the system's actions and outcomes. The edges E denote the causal relationships between these variables, indicating how changes in one variable can affect others. The joint probability distribution over all the variables in the Bayesian network is given by:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | pa(X_i))$$

where $pa(X_i)$ are the parents of node X_i, signifying the variables that directly influence X_i.

To perform probabilistic reasoning within the Bayesian network, we utilize inference algorithms. The variable elimination algorithm simplifies the calculation of marginal probabilities by systematically eliminating variables one by one, reducing the computational complexity of the analysis. Markov chain Monte Carlo (MCMC) methods, on the other hand, are useful for sampling from complex probability distributions when exact calculations are infeasible. These methods allow us to approximate the probabilities of different states and events within the network.

Moreover, we explore methods for learning the structure of the Bayesian network from data. This involves techniques such as score-based methods, for example, the Bayesian Information Criterion (BIC), which evaluates different possible network structures based on a combination of how well they fit the data and their complexity. Another approach is constraint-based methods, which analyze the dependencies

between variables in a given dataset to infer the appropriate causal relationships and construct a DAG that accurately represents the underlying structure of the system's domain.

While Bayesian networks form our primary approach for reasoning, we also consider alternative methods depending on the specific requirements of the application. In situations where there is a significant level of vagueness or imprecision in the information, fuzzy logic can be employed. Fuzzy logic allows for reasoning with degrees of truth rather than strict binary values, enabling the system to handle uncertainties more gracefully. For example, when assessing the similarity between two objects or the likelihood of an event occurring under ambiguous conditions, fuzzy logic can provide a more flexible and nuanced way of making decisions. Additionally, rule-based systems can be used to represent expert knowledge in a more explicit and interpretable manner. In a rule-based system, if certain conditions are met based on predefined rules, specific actions or decisions can be made. This can complement the probabilistic reasoning done by the Bayesian network, especially in cases where there is well-established domain knowledge that can be encoded as rules.

3.4. Model the Action Space

The action space of the AI agent is modeled using reinforcement learning, specifically Q-learning. Here, let S be the set of all possible states that the system can be in, A be the set of all available actions, and $R(s, a)$ be the reward function that assigns a numerical value representing the immediate reward obtained by taking action a in state s. The Q-value function, which estimates the expected cumulative reward starting from a given state and taking a particular action, is updated according to the following rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

Here, α is the learning rate that determines how quickly the Q-values are updated based on new experiences, γ is the discount factor that balances the importance of immediate rewards versus future rewards, and s' is the next state that the system transitions to after taking action a in state s.

For example, consider a robot learning to navigate through a maze. The different positions and orientations of the robot within the maze represent the states S, and the actions A could include moving forward, moving backward, turning left, or turning right. The reward function $R(s, a)$ might assign a positive reward for reaching the goal, a negative reward for hitting a wall, and a small negative reward for each step taken to encourage efficient navigation. Through Q-learning, the robot gradually learns the optimal actions to take in each state to maximize its cumulative reward and reach the goal in a timely manner.

In more complex environments where the action space is continuous or the state space is large and high-dimensional, we extend our approach to utilize Deep Q-Networks (DQNs) and potentially policy gradient methods like Deep Deterministic Policy Gradient (DDPG). DQNs use neural networks to approximate the Q-value function, enabling the system to handle complex visual and sensory inputs and make decisions in environments with a vast number of states and actions. DDPG, on the other hand, is designed for continuous action spaces, allowing for more precise control of actions such as the movement of a robotic arm with multiple degrees of freedom.

4. Our Proposed Pruning Algorithm

Our proposed pruning techniques revolve around the **wisdom graph**, an underlying data structure that maps the four essential AI agent spaces - perception, planning, reasoning, and action in a computationally efficient and high level manner which we call representation learning for widsom. In the perception space, the wisdom graph maps sensory inputs to graph nodes. For instance, visual signals from cameras are represented as feature vectors at nodes, which are then connected based on spatial and semantic relationships. Audio inputs, processed into relevant features like MFCCs, are also incorporated, with nodes reflecting different speech related states. This mapping allows for quick retrieval and processing of perceptual information during inference. Regarding the planning space, high level tasks are mapped to top level nodes in the wisdom graph. These tasks are hierarchically decomposed, with each sub-task becoming a lower level node. The edges between nodes represent task dependencies and sequences. During pruning, the graph's structure enables the identification of redundant or inefficient sub-tasks, optimizing the planning process for better computational performance. In the reasoning space, Bayesian network based representations in the wisdom graph map random variables, such as task success probabilities and resource availability, to nodes. The causal relationships between these variables are depicted as edges. This mapping supports probabilistic reasoning during inference, and pruning can be carried out by removing less influential nodes and edges, streamlining the decision making process. Finally, in the action space, the wisdom graph maps states, actions, and rewards. Each state is a node, and actions are represented as directed edges connecting states. Rewards associated with these state - action pairs are stored within the graph structure. This mapping is fundamental for Q-learning based algorithms. Pruning here involves eliminating actions that yield consistently low rewards, enhancing the agent's action selection efficiency during operation.

Building upon the wisdom graph based representation of AI agent spaces, **our improved pruning methods** offer enhanced efficiency and performance. By leveraging the graph's structure, we can conduct targeted pruning operations that optimize the agent's decision making process. For the perception space within the wisdom graph, we implement a form of hierarchical pruning. We first group related perceptual features into clusters at higher levels of the graph. Then, during pruning, we can remove entire clusters that are deemed less relevant based on their contribution to the overall task. For example, in a visual perception task for an autonomous vehicle, if certain visual features related to distant, non-critical objects are identified as having minimal impact on the vehicle's navigation decisions, the corresponding clusters in the wisdom graph can be pruned. This reduces the computational load during perception processing without sacrificing crucial information for navigation. In the planning space, we use a cost based pruning approach. Given the hierarchical task representation in the wisdom graph, we calculate a cost function for each sub-task node. This cost function takes into account factors such as resource requirements, estimated time to completion, and the risk associated with the task. Sub-tasks with high costs and low contribution to the overall goal are pruned. For instance, in a manufacturing assembly task, if a particular sub-task requires excessive resources and can be bypassed without affecting the final product quality, the wisdom graph allows us to easily identify and remove it from the plan, streamlining the production process. Regarding the reasoning space, we apply a relevance based pruning method. Since the wisdom graph represents causal relationships between variables as in a Bayesian network, we can measure the relevance of each node (variable) to the final decision making process. Nodes that have a negligible impact on the outcome, based on statistical analysis of their influence over other variables, are pruned. This not only speeds up the reasoning process but also improves the clarity of the decision making mechanism. Finally, in the action space, we use a reward based pruning strategy. As the wisdom graph stores state-action reward relationships, we can analyze the long term rewards associated with each action. Actions that consistently result in low rewards across different states are removed from the graph. In a robot navigation task, if a particular movement action always leads to a dead end or a low reward area, the wisdom graph enables us to prune this action, guiding the robot towards more productive paths and enhancing its overall performance.

Preliminary experimental results show that both the wisdom graph and the improved pruning technique are working good.

Algorithm 1 Mama Prune - the detection phase

Input: node x_i, size m
repeat
  Initialize $noPruneFlag = true.$
  for $i = 1$ **to** $m - 1$ **do**
    if score(x_i) <Threshold **then**
      $noPruneFlag = false$
    end if
  end for
until $noPruneFlag$ is $true$

5. Related Work

Computational communication, at its core, represents an interdisciplinary approach that merges natural science and social science. This integration is fundamental in understanding how information is disseminated, received, and utilized. It provides crucial insights into the processes of information communication, broadcasting, and how target audiences benefit from the received information. The evolution of computational communication has been extensive, spanning from the era of search engines to the rise of social media and now extending into the realm of content generation. In the context of search engines, computational communication techniques were employed to index and retrieve relevant information efficiently. This involved algorithms that ranked search results based on relevance, enabling users to access the information they needed. With the advent of social media, computational communication took on a new dimension. It focused on how information spreads virally across social networks, considering factors such as user behavior, network structure, and the influence of social connections. Platforms used algorithms to personalize content feeds, ensuring that users were exposed to information tailored to their interests. Today, in the age of generation, computational communication is involved in creating and distributing new forms of content, such as generated text, images, and videos. This not only requires understanding how to produce engaging content but also how to target the right audience effectively. When it comes to **AI agent pruning**, computational communication plays a vital role. In the perception space of an AI agent, signals are received from the environment. Computational communication concepts can be applied to optimize the way these signals are broadcasted to the planning space. For example, techniques from computational communication can be used to filter and prioritize the incoming signals, ensuring that only the most relevant information is passed on. In the planning space, computational communication can be seen in the pruning and broadcasting processes. Pruning, a key aspect of our research, can be informed by communication based strategies. Just as in information dissemination where redundant or less relevant information is filtered out, in the planning space, sub-optimal plans can be pruned. The remaining plans are then broadcasted for further processing in other spaces, such as the reasoning and action spaces. This cyclic process of broadcasting and pruning between different spaces of the AI agent is parallel to the concepts of information flow and optimization in computational communication. Overall, computational communication provides a rich theoretical and practical framework that can enhance our understanding and improvement of AI agent pruning techniques.

6. Conclusion

In conclusion, this research on AI agent pruning presents a significant advancement in the field. Our key contributions lie in three main aspects. First, we introduced a novel conceptual model that organizes an AI agent's operations around four crucial spaces: perception, planning, reasoning, and action. This model provides a clear and structured framework for understanding how agents interact with their environment.

Second, we successfully translated this conceptual model into mathematical models. By leveraging various mathematical techniques for each of the four spaces, we have enabled more precise and efficient implementation of AI agents. This allows for better management of the complex processes within the agent, from gathering information in the perception space to making decisions in the reasoning space and executing actions.

Third, we explored and practiced potential pruning techniques. These techniques have the potential to optimize the performance of AI agents by removing redundant or sub-optimal elements in the four spaces, enhancing decision making speed and stability while minimizing resource consumption.

However, our work is not without limitations. The current pruning techniques could be enhanced to embrace more dimensionalities. This would enable the agents to handle more complex and diverse data, leading to more accurate and adaptable decision making. For future work, we plan to delve deeper into the areas mentioned above. We aim to refine the mathematical models, improve the pruning techniques to handle higher dimensional data, and further explore the potential of our proposed conceptual model in different applications, ultimately driving the development of more advanced and efficient AI agent.

Accessibility

We make our work as accessible as possible for everyone including people with disabilities and sensory or neurological differences.

Software and Data

Software is accessible here *. The data is accessible here *.

Impact Statement

This paper presents work whose goal is to advance the field of AI Agent, Robotics, and Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Dean, J. How search works, 2014. URL https://www.google.com/intl/en/search/howsearchworks/how-search-works/.

Frantar, E. and Alistarh, D. Sparsegpt: Massive language models can be accurately pruned in one-shot. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10323–10337. PMLR, 2023. URL https://proceedings.mlr.press/v202/frantar23a.html.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep learning*, volume 1. MIT Press, 2016.

He, K. Deep generation models, 2024. URL https://mit-6s978.github.io/.

Hinton, G. E., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015. URL http://arxiv.org/abs/1503.02531.

OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023. doi: 10.48550/ARXIV.2303.08774. URL https://doi.org/10.48550/arXiv.2303.08774.

Rodriguez, J. *Optimizing Search Engine Efficiency with Static Index Pruning and Tiering*. PhD thesis, Tandon School of Engineering, New York University, 2021.

Russell, S. and Norvig, P. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020. ISBN 9780134610993. URL http://aima.cs.berkeley.edu/.

Suel, T., Rodriguez, J., and Jiang, W. Improved methods for static index pruning. In Joshi, J., Karypis, G., Liu, L., Hu, X., Ak, R., Xia, Y., Xu, W., Sato, A., Rachuri, S., Ungar, L. H., Yu, P. S., Govindaraju, R., and Suzumura, T. (eds.), *2016 IEEE International Conference on Big Data (IEEE BigData 2016), Washington DC, USA, December 5-8, 2016*, pp. 686–695. IEEE Computer Society, 2016. doi: 10.1109/BIGDATA.2016.7840661. URL https://doi.org/10.1109/BigData.2016.7840661.

Sun, M., Liu, Z., Bair, A., and Kolter, J. Z. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=PxoFut3dWW.

W, Y, L, W, and J. Ai agent sample work outcome, 2025. URL https://github.com/algmon/the-fashion-queen-agent.