Dissecting similarities in self-consistency: An analysis on impact of semantic consistency on language model reasoning

Anonymous ACL submission

Abstract

While large language models (LLMs) have rapidly improved performance on a broad number of tasks, they still fall often fall short on reasoning tasks. Wang et al. (2023) propose *self-consistency*, finding that sampling multiple rationales before taking a majority vote stably improves performance across a wide variety of closed-answer reasoning tasks. Standard self-consistency aggregates the numerical outputs of these rationales; our work instead incorporates the content of the rationales to identify consensus responses, re-weighting solutions based on patterns found in their vector embeddings of sequence outputs. By doing so we analyze and evaluate the implied effect of consistent reasoning paths over the traditional focus on numerical outputs, while improving accuracy on common benchmarks by weighting based on semantically consistent answers.

1 Introduction

High-level thoughts so far: there are some impressive results that are buried deep in the results section (outlier detection first and foremost; inverse weighing next). Take those and put them at the beginning. Some of the results (k-means, e.g.) are more so additional studies and should be put there to clarify the narrative. Goal here is to streamline things and make it very clear to a reviewer what the contributions of the paper are.

In recent years, the development of large language models has witnessed remarkable strides, with significant advancements in their accuracy and expressive capabilities. (Brown et al., 2020; Sarker, 2021; Naveed et al., 2023; Bubeck et al., 2023) Despite these achievements, models still perform suboptimally in domains such as mathematic, commonsense, and complex algorithmic reasoning. (Hendrycks et al., 2021)

We build on the framework of self-consistency, a technique that samples and ensembles multiple model responses to improve prediction quality (Mialon et al., 2023). Our paper introduces various methods that improve performance and accuracy by exploiting semantic contrast between generations. We propose multiple techniques that adds a separate filtering layer to discard irrelevant, inaccurate or degenerated responses. Furthermore we introduce the application of semantic vector embeddings in relationship to self-consistency to group consistent model outputs, aiding identification of alike responses to estimate an accurate representation about output sequences. Additionally weighting responses based of these semantic representations has shown an inclining effect on model performance in terms of accuracy. We also explore the impact of weighting responses based on these semantic representations. Figure 1 exemplary illustrates our filtering process after mapping embeddings to a two-dimensional space.

Overall, we show that self-consistency with semantic marginalization not only substantially improves accuracy on a range of benchmarks, but also can be used as a filtering mechanism to improve robustness towards nonsensical and degenerated responses. By addressing these issues we want to provide multiple methods that can be utilized as a framework towards improvement of performance and more textually aware and concise sequences in the majority responses.

Our contributions are as follows:

1. **Clustering Based on Embedding Vectors**: Our research successfully clustered results based on embedding vectors. This approach can be instrumental in identifying underlying patterns and structures in complex data sets.

2. **Weighted Results Analysis**: We introduced a novel approach to weigh results based on their mapped position relative to the overall mean of all data points. This method offers a

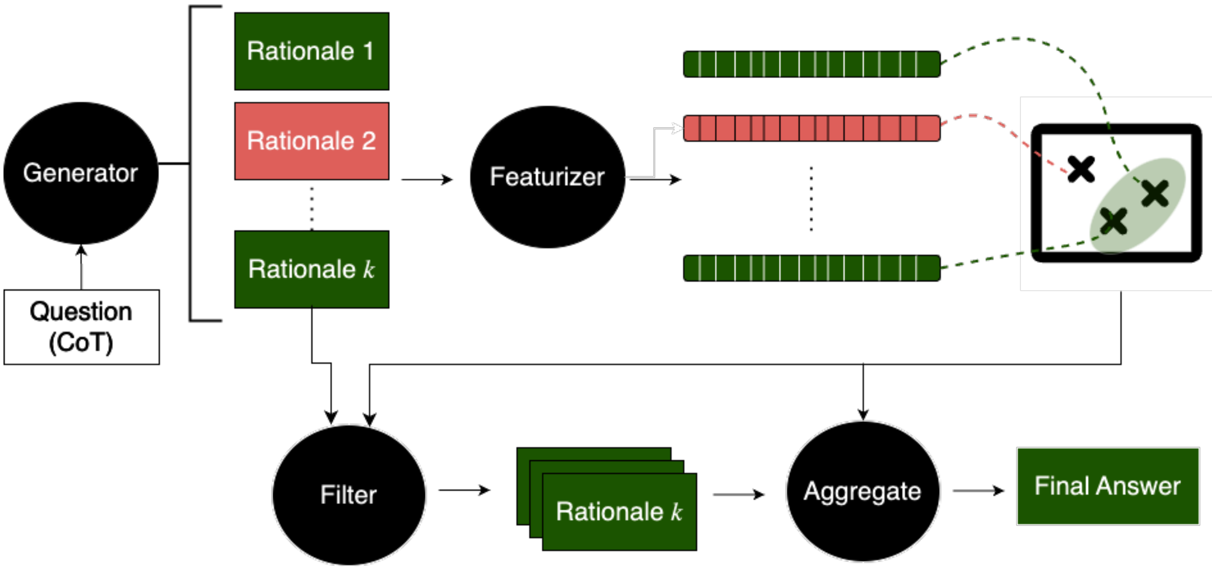

Figure 1: Default self-consistency comprises three steps: (1) Prompt a model with chain-of-thought reasoning; (2) Generate n sampled sequences, and (3) Marginalize results based on the most occurring numerical output. Our proposed method samples results and marginalizes not only based on consistency in the output but also on the consistency of the employed reasoning path. Our assumption is that Language Models often apply the correct reasoning but lack the ability to conduct the needed mathematical operations correctly. We utilize this concept to let reasoning paths improve the confidence in similar reasoning responses.

refined data analysis technique that could be beneficial in large-scale data studies.

3. **Anomaly Handling via Marginalization**: We developed a method to marginalize out anomalous points based on mapped embedding vectors, enhancing the robustness and reliability of data-driven models, particularly in scenarios with noisy or outlier data.

4. **Sequence Similarity Evaluation using Cosine Similarity**: We evaluated the similarity between subsequent responses using cosine similarity, providing a quantitative measure of the effectiveness of response generation algorithms in maintaining thematic consistency.

2 Methodology

2.1 Semantic marginalization techniques

We analyse a range of mechanisms for weighting and categorization.

1. ***Generate candidate responses:*** Given a query of few-shot examples, we generate n samples based on chain of thought prompting. (Wei et al., 2022)

2. ***Embed reasoning paths:*** Here, we deviate from the typical sentence-wise approach used in BERT models. Instead, we take the entire sequence, including the generated responses, and use fine-tuned variants of BERT-models to embed the answer in semantic space.

3. ***Filter and marginalize:*** We use various algorithms to filter and marginalize out results based on its featurized embedding vector.

2.1.1 Inverse-distance weighting

In a set of examples, it is common to observe that general answers exhibit similar operational patterns and behaviors. This observation underpins the application of inverse distance weighting, a technique where each vector in the set is assigned a weight based on its distance from a reference point or query. The essence of this approach lies in the principle that vectors closer to the query are more likely to be relevant and thus are given greater weight in the decision-making or reasoning process.

We calculate the weights for each data point and normalize the weights so that they sum to 1. The process is shown below. To quantify these distances and subsequent weights, we adapt a radial basis function.

$$\text{centroid} = \frac{1}{N}\sum_{i=1}^{N} \text{data\_embedding}[i]$$

$$\text{distances}[i] = \|\text{data\_embeddings}[i] - \text{centroid}\|$$

$$\text{weights}[i] = \frac{1}{\text{distances}[i]}$$

Optional normalization step:

$$\text{weights}[i] = \frac{\text{weights}[i]}{\sum_{i=1}^{N} \text{weights}[i]}$$

In these formulations:

- centroid symbolizes the geometric center of all data points.
- distances$[i]$ denotes the distance of the i-th data point from the centroid.
- weights$[i]$ indicates the normalized weight of the i-th data point, derived from its distance to the centroid.
- N is the total number of data points in the dataset.
- data_embedding$[i]$ represents the vector representation of the i-th data point.
- $\|\cdot\|$ signifies an arbitrary distance function, including but not limited to Euclidean and Manhattan distance.

Our results are evaluated with Euclidean distance. Additionally, we use Manhattan (L1)-distance as an alternative approach to Euclidean distance to measure the closeness of relevant data points, which is more robust to outliers.

2.1.2 Identification of Anomalous Data Points

Our research involved a thorough examination of different techniques for detecting outliers, specifically focusing on methods such as k-nearest neighbors (KNN), isolation forest (ISF), and One-class support vector machines (OCSVM) (Liu et al., 2008; Manevitz and Yousef, 2002; Cover and Hart, 1967).

For the KNN method, the distance $D(x, y)$ between two points x and y in an n-dimensional space is calculated using the formula:

$$D(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{1}$$

This formula helps in determining the closeness of data points in the feature space.

In the ISF approach, the anomaly score $s(x, n)$ of a data point x is determined based on the path length $h(x)$ within the isolation tree, using the formula:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{2}$$

Here, $E(h(x))$ represents the average path length and $c(n)$ is a normalization factor.

Lastly, for the OCSVM technique, the objective is to find the parameters ω, b, and ζ_i that minimize:

$$\min_{\omega, b, \zeta} \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{n} \zeta_i \tag{3}$$

Subject to constraints: ω and b define the hyperplane, ζ_i are slack variables allowing for anomalies, and C balances margin maximization with classification error minimization, preventing disregarding data points.

The objective of this analysis was to effectively isolate data points that significantly diverge from the norm. This is particularly relevant in identifying instances of flawed reasoning, degenerated outputs, or hallucinations within the model response.

2.2 Sequence comparison

To get a direct comparison of effectiveness between evaluating the embedding position in correlation to its other datapoints and evaluating wise we used cosine similarity to evaluate direct similarities between sequences.

Therefore we take $n_1, n_2, n_3, \ldots, n_i$ which represents distinct elements in our set N, where each element n corresponds to a featurized embedding in the vector space.

Then we determine the cosine similarity between all vectors (Here n_a and n_b) given by the formula:

$$\text{cosine\_similarity}(n_a, n_b) = \frac{n_a \cdot n_b}{\|n_a\|_2 \|n_b\|_2}$$

For a given rationale n_e, we evaluate the cosine similarity between n_e and each n_i in the set N.

$$\forall n_i \in N, \text{ calculate cosine\_similarity}(n_e, n_i)$$

Then, we aggregate the weights (or scores) of all these cosine similarity results for n_e. By summing:

$$S_{n_e} = \sum_i \text{cosine\_similarity}(n_e, n_i)$$

where S_{n_e} represents the aggregated score for n_e.

This process is then repeated for each element n_j in the set N, resulting in a series of aggregated scores $S_{n_1}, S_{n_2}, S_{n_3}, \ldots, S_{n_i}$.

These scores are then summed based on their answer decision. This system effects that the highest consensual response gets chosen as the solution.

2.3 Abstract Consistency

One of our findings utilizes the premise that exposing the model to a spectrum of different temperatures facilitates the model of more diverse decision-making processes. This process could be harnessed to improve the introduced semantic marginalization methods. Our configuration is explained in more detail in Appendix D.

3 Experimental Setup

We conduct multiple experiments with varying setups in form of different benchmarks tested on each model to cover a broad range of possible outputs. Detailed information on the configurations used for out models can be found in Appendix C.

3.1 Dimensionality reduction

We test dimensionality reduction with PCA and t-SNE to see performance and preservation of the distribution on different algorithms. (Pearson, 1901; Hotelling, 1933; Jolliffe, 2002) A detailed overview is referenced in Section 5.5.

Additionally use the t-SNE for the visualization of high-dimensional vector spaces, the configuration is explained in Appendix J. (van der Maaten and Hinton, 2008)

3.2 Datasets

3.2.1 Arithmetic reasoning

We evaluate arithmetic reasoning on AQuA-RAT and SVAMP. (Ling et al., 2017; Patel et al., 2021) We also use GSM8K (Cobbe et al., 2021) for some ablations to evaluate performance on lower-difficulty problems.

3.2.2 Code synthesis

To test our hypothesis on code generation we use HumanEval introduced by Chen et al. (2021) in connection with OpenAI.

3.3 Language Models

Our models are divided into *generators*, which provide the reasoning/result sequences of of which we build the solutions and *featurizers*, which convert the output sequences into suitable vector representations.

3.3.1 Generators

- **GPT-3.5:** For our evaluation we use the closed-source GPT-3.5 model architecture which is a transformer based large-scale language created by OpenAI.(Brown et al., 2020)

- **Llama 2:** Llama 2 is a collection of open-weight Transformer models that perform well on a multitude of common benchmarks. We evaluate the 7-billion parameter variant. (Touvron et al., 2023)

- **Mistral 7B:** Mistral 7B is a strong front to back transformer model. (Jiang et al., 2023) It outperforms larger-parameter models in processing large contextual information. We are using version 0.1 of the model.[1]

3.3.2 Featurizers

All of our featurizers are based on the BERT-architecture. (Devlin et al., 2019) This enables us to use different fine-tuned models to produce more concise embedding-vectors based on the given task.

- **roBERTa:** roBERTa (Liu et al., 2019) is an "robustly" fine-tuned 125M parameter model derived from the original BERT architecture, featuring careful optimization to outperform its predecessor on several natural language processing benchmarks.

- **sciBERT:** sciBERT is a 110M parameter BERT-model fine-tuned on a multi-domain corpus of roughly 1.14M scientific publications, making it particularly adept at understanding more complex terminology and structure in academic contexts. (Beltagy et al., 2019)

- **MathBERT:** MathBERT is a 100M token BERT-model that is fine-tuned on mathematical language based on up to an college level math curriculum, books and math arXiv-paper-abstracts.(Shen et al., 2023)

- **codeBERT:** codeBERT is a 125M parameter fine-tuned BERT model for coding assignments with a more pronounced understanding of code. (Feng et al., 2020)

[1] Our employed model does not utilize instruction tuning.

4 Results

4.1 Weighting results

4.1.1 Arithmetic reasoning

The results presented in Table 1 demonstrate notable improvements in accuracy when inverse distance weighting is applied, particularly in scenarios with higher variance in overall numerical outputs. The weighting models based on the inverse of the distance outputs have shown to improve overall self-consistency by an average margin of **3.75%** for AQuA-rat and **0.9%** for SVAMP.

The use of Euclidean distance has yielded higher average results but also greater variance in accuracy compared to Manhattan distance. This suggests that penalizing more deviating results can be beneficial for models with stronger performance. We observe the same correlational increase in performance in higher parameter models as already percieved in *self-consistency* and chain-of-thought prompting.

4.1.2 Weighted Code Synthesis

As evidenced in Table 2, employing inverse distance weighting enhances the quality of code synthesis. This method consistently selects the sample with the greatest weighting, aligning it closer to the aggregate mean. Importantly, this approach demonstrates a preference for clean and concise code. This increases the likelihood of a sample being nearer to the mean, especially when the majority of code samples exhibit qualities of clarity and brevity.

Table 2: Model Performance Overview on HumanEval at pass@1

Model	Dataset	accuracy (%)	
		Avg. default	Inverse Distance Weighting
Mistral	HumanEval	18.7	23.8 (**+5.1**)

4.2 Self-consistency with outlier detection

Outlier detection proves crucial for enhancing the overall quality of the results. This technique effectively marginalizes points that detract from the model's self-consistency and filters out irrelevant responses. This refinement in output quality is evident even when the quantity of samples is reduced, suggesting that the effectiveness of anomaly detection techniques is not solely dependent on sample size. [2] Results show meaningful increases in performance over the default. Anomaly detection[3], while showing a frailty across different results with deviations up to 1% of the baseline, becomes a pivotal method when considering the dual benefit of outlier detection.

By selectively sampling out these outlier points, not only is the relevance of the responses maintained, but the model's self-consistency is ensured in a reduced sample space. This suggests that using outlier detection techniques can lead to a cleaner analysis and a more comprehensive distribution of relevant results, aiding in understanding the actual deviation of reasoning paths that are significant to the results.

4.3 Direct comparison of Sequences

To get a direct comparison of effectiveness between evaluating the embedding position in correlation to its other datapoints and evaluating sequence wise we used cosine similarity to evaluate direct similarities between sequences. (Gatto et al., 2023)

Model	AQuA-rat	SVAMP
LLAMA 2	25.0 (**+0.2**)	46.9 (**+0.4**)
MISTRAL	29.8 (**+3.6**)	70.2 (**+1.7**)
GPT3.5	65.4 (**+6.0**)	80.3 (**+0.5**)

Table 4: Showcasing cosine similarity (weighted) compared to all rationales

These results show that when sequences get weighted based on maintained consistency between all responses, we exhibit results that are more prone to errors and reveal higher accuracy that got lost in default self-consistency.

4.4 Abstract Consistency

While default self-consistency samples of one static temperature models often present results that are either deterministic or overly random, our employed mechanism allows the model to find a "sweet-spot" that lies high emphasis on wide-ranging but sensical reasoning paths. To leverage this, we sample from a wide distribution of different reasoning paths, from a variety of **5** different temperatures

[2]The obtained results exhibited slight deviations between the different configurations. An extensive review across different sets of configurations and parameters can be found under Appendix H.1 to H.3.

[3]To provide a more stable assessment, we average the results across all variations of different parameters.

Model	**Method**	**AQuA-rat**	**SVAMP**
	SC baseline	24.8	46.5
Llama 2 7B	Inverse distance	24.6 (**-0.2**)	47.4 (**+0.9**)
	L1 inverse distance	23.9 (**-0.9**)	46.7 (**+0.2**)
	SC baseline	25.6	68.5
Mistral 7B	Inverse distance	29.0 (**+3.4**)	69.8 (**+0.3**)
	L1 inverse distance	28.6 (**+3.0**)	69.8 (**+1.3**)
	SC baseline	59.4	79.8
GPT 3.5	Inverse distance	68 (**+8.6**)	81.0 (**+1.2**)
	L1 inverse distance	68 (**+8.6**)	80 (**+0.2**)

Table 1: Comparison of Inverse distance weighting on different distance metrics and models

Model	Method	AQuA-rat		SVAMP	
		Best	Average	Best	Average
	SC baseline	24.8	24.8	46.5	46.5
LLAMA 2	Isolation Forest	**28.45**	**26.04**	45.94	45.60
	K-nearest-neighbors	25.40	25.37	45.85	45.71
	Oneclass SVM	**26.70**	24.25	44.94	43.30
	SC baseline	25.6	25.6	68.5	68.5
Mistral	Isolation Forest	**26.61**	25.97	68.84	68.34
	K-nearest-neighbors	25.91	25.66	68.84	68.52
	Oneclass SVM	**28.45**	26.08	67.23	65.33
	SC baseline	59.4	59.4	79.8	79.8
GPT3.5	Isolation Forest	**65.27**	**63.73**	**84.65**	**84.28**
	K-nearest-neighbors	**62.81**	60.04	**84.64**	**84.42**
	Oneclass SVM	59.55	59.26	**85.23**	**84.54**

Table 3: Outlier detection performance on SVAMP and AQuA-rat. Performance increase over baseline of $n > 1\%$ featured in bold.

per generation. These findings show that ***Abstract Consistency*** not only provides a wider range of outputs with a more diverse spectrum of answers, but also performs above average compared to default ***self-consistency***.

Method	**Accuracy (%)**
Self-Consistency	46.50
Abstract consistency MV	46.53
Abstract consistency (weighted)	**48.54**

Table 5: Weighted self-consistency with varying levels of abstraction improves performance over default.

It is to note that higher temperature showed a degree of randomness that can lead to higher degeneration. However this limiting factor can be mitigated when applied with inverse temperature weighting and improve performance of up to **2.5%**. The effect of different temperature sets can be found in Appendix I

5 Additional studies

5.1 Finetuned featurizers

The process of converting rationales into semantic embedding vectors was applied to multiple featurizer-models at different forms of fine-tuning to measure the ability of models to effectively convert sequences into fitting embedding vectors.

BERT-Model	**avg distance (↓)**	
RoBERTa	48.697	
MathBERT	45.892	**(-2.8)**
SciBERT	45.281	**(-3.4)**

Table 6: Featurizers finetuned on similar distributions tend to pack answers more tightly together

The results revealed elevated results for SciB-

ERT and MathBERT when compared to RoBERTa. This is likely due to RoBERTa's general robust training where in contrast, both MathBERT and SciBERT exhibit stronger performance[4]. We conjecture that this is due to their training data being more representative of the reasoning tasks that we evaluate on here (Sun et al., 2020). This observation suggests that improper or "unfitting" fine-tuning reduces overall data point density, resulting in a loss of information within the produced vectors, and consequently hindering subsequent marginalization techniques (Merchant et al., 2020).

5.2 Comparison and effects

Meta-Reasoning over multiple chains of thoughts While meta reasoning has proven effective on tasks that have qualitative evident information, its ability to stay consistent between arithmetic operations and its subsequent reasoning path witnesses the same limiations as default self-consistency and chain of thought. (Yoran et al., 2023)

5.3 Evaluation on clusters

The implementation of k-means clustering[5] showed that regardless of the fact that reasoning can be improved by detailed mappings, clustering didn't attribute to enhance the quality of the semantic evaluation. Additionally we reason this to be attributed to two limiting factors:

1. Lower amount of samples used for evaluation
2. To broad marginalization and consideration as outlying points

We systematically experimenting with a spectrum of values for the parameter k, with a significant emphasis on k=2 to ensure that the clusters would still provide a sufficient amount of associated rationales with each cluster to utilize the effect of self-consistency.

Our objective was to ensure that these more substantial clusters provide a robust framework for the influence of self-consistency. It is probable that higher amounts of samples enables not only better and more accurate clustering but enables higher values of k to show higher performance.

[4]Tested on arithmetic samples only, due to their greater variability and problem-solving scope compared to the more logic-bound and less varied nature of coding tasks.

[5]Averaged over 10 random states to ensure an representative example. Please refer to Appendix G.2 for the unaveraged values.

Table 7: Performance using k-means for outlier detection, with $k = 2$

Model	AQuA-rat	SVAMP
LLAMA 2	24.16	42.47
Mistral	24.83	62.52
GPT-3.5	65.52	78.67

Table 8: Averaged over 10 runs, clustering has shown volatility based on initial cluster placement. The unaveraged runs are referenced in Appendix G.2

This method implies that the predictions associated with the majority cluster are the ones for which the model exhibits the greatest overall confidence. A detailed accessment of the found results can be accessed in Appendix G.1.

5.4 Result augmentation

To enhance the quality of our embeddings and ensure they are not clustered solely based on output results, we implemented a process of result augmentation. This involved removing end results before generating embedding vectors, which were then used to form clusters. Our findings demonstrate that this approach shows the influence of inconclusive answers without results and proves that even incorrect outputs can still be used in different methods to enhance overall output quality and mechanisms that make use of semantic evaluation.

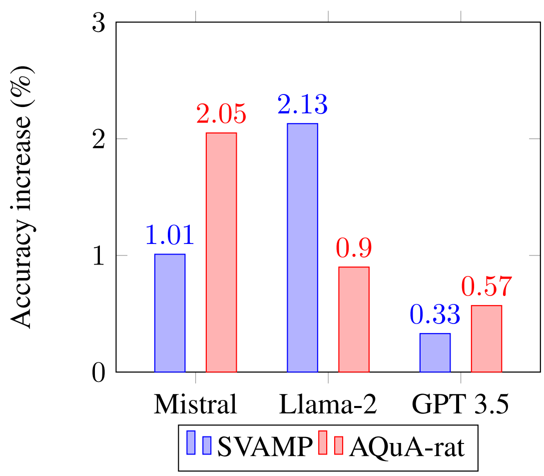

Figure 2: Accuracy representation with and without incorporating results from None numerical solutions.

5.5 Robustness to dimensionality reduction

Inverse distance has shown high variance over different dimensionality reduction techniques which impacts accuracy on a margin that overall decreases performance.

In high-dimensional spaces, both Euclidean and Manhattan distances demonstrate effective performance, making them viable for visualization purposes. However, they are less suitable for weighting data points when benchmarking performance.

Model	Dataset	PCA	t-SNE
LLAMA 2	AQuA-rat	22.98	**25.0**
LLAMA 2	SVAMP	43.04	42.84
MISTRAL	AQuA-rat	**26.21**	**25.81**
MISTRAL	SVAMP	66.77	63.76
GPT3.5	AQuA-rat	**66.23**	**63.37**
GPT3.5	SVAMP	**80.15**	79.16

Table 9: Dimensionality reduced results that improve quality over default are featured in Bold.

5.6 Correlation of Sequence Length on Model Performance

We observe a correlation[6] indicating statistical significance, supporting the robustness of the observed trend between the average sequence length generated by our models and the improvement in accuracy when employed with inverse distance weighting.

We attribute this to the increased importance of exemplar selection across longer chains of thought that can be more prone to outliers over the course of the reasoning process.

Dataset	Model	Avg. Seq. Length	Avg. Accuracy Increase (%)
AQUA-rat	GPT3.5	102.40	8.6
AQUA-rat	Mistral	53.24	3.2
SVAMP	MISTRAL	52.92	0.8
SVAMP	LLAMA 2	52.29	0.5
SVAMP	GPT3.5	49.71	1,3
AQUA-rat	LLAMA 2	49.58	-1.55

Table 10: Comparison of Sequence Length and Accuracy Increase

6 Related Work

-

Reasoning has been identified as an ubiquitous issue, across many domains in Large Language Models (Creswell et al., 2022). After Rae et al. (2021) highlighted the challenges in reasoning across various domains in Large Language Models, subsequent research has increasingly focused on enhancing these models reasoning capabilities.

[6] ρ = *0.83*, p-value 0.042

One general method applied in many of those studies, is **few-shot learning** which shown positive results in guiding a model into a more contextually aware and accurate direction. By training with a small but highly fitting set of examples, these models demonstrate an enhanced ability to infer and apply knowledge. (Brown et al., 2020)
Furthermore **fine-tuning** has shown positive results on specialized data in a broad amount of areas. Research by Radford and Narasimhan (2018) shows that targeted fine-tuning can notably enhance the model's performance in certain areas.
One other significant advancement in the area that has synergized with few shot has been the development of the '**chain of thought**' prompting, which guides LLM's to mimic human-like step-by-step reasoning processes. (Wei et al., 2022) We also draw information from Saparov and He (2023) which discusses chain-of-thought on a fundamental level. In the context of our research, we extend the concept of **self-consistency**, as originally proposed by Wang et al. (2023).

7 Conclusion and discussion

This study demonstrates that a model's reasoning path can be a relevant attribute when evaluating responses. We overview straightforward yet effective methods to improve self-consistency by utilizing the coherency and consistency of reasoning sequences, while maintaining sequence production. Furthermore, manipulating output sequences serves not just to improve accuracy but data quality and robustness. Marginalizing outliers specifically shows promise for increasing reliability and integrity of evaluation sequences. Additionally, sampling from different temperatures improves over static sampling. Future work may use these techniques to increase commonsense reasoning performance or apply the reasoning path methods and marginalization for other intrinsic evaluations.

8 Limitations

Our study proposes the application of semantic vector representations to group and weigh model outputs, which is designed to facilitate the identification of consensus responses (Wang et al., 2023). Semantic vectors must capture the subtle variations in meaning and context, which is particularly hard in abstract reasoning tasks without a sufficient amount of context making prompting techniques to enhance the models output structure and size an important factor as visualized in Table 10. The process of clustering based on semantic vectors can be challenging due to the nuanced and abstract nature of reasoning processes. This limitation underscores the need for advanced featurization models and explicit choice of a fitting fine-tuned model (Merchant et al., 2020). Like showcased in Table 6, multiple models should be considered for semantic analysis, to ensure that the model outputs are grouped in a way that truly reflects their underlying meaning and relevance.

9 Reproducibility Statement

Our experiments include a variety of models with different sizes: Microsoft Phi1.5B is publicly available at https://huggingface.co/microsoft/phi-1_5/tree/main and can be used under the Microsoft Research License.
GPT-3 has an API that is open for public use https://openai.com/blog/openai-api.
Mistral 7B is available for unrestricted use under the Apache 2.0 license, while its model architecture and setup are open source https://github.com/mistralai/mistral-src.
Llama 2 is a model with restricted access, made available by Meta. You can gain access to it by requesting permission through the provided Meta license. You can find more information about it at https://ai.meta.com/llama/.
All of our BERT models are built upon the BERT-base model developed by google-research, which is accessible under the Apache 2.0 license including MathBERT and sciBert. RoBERTa and codeBERT can be used under the MIT license.

Our Datasets as well as used configuration for our language Models, are accessible throughout this paper and in the Appendix to aid the reproducibilty of our experiments.
A majority of our experiments were done using huggingface to access datasets, models and general data. Some of the used algorithms were implemented with scikit-learn (Pedregosa et al., 2011) and the sklearn api (Buitinck et al., 2013).

9.1 GPU usage

approx. Hours	GPU	Model	Memory
200 h	NVIDIA	T4	15GB
50 h	NVIDIA	V100	16GB
50 h	NVIDIA	A100	40GB

10 Ethical Considerations & Risks

Language Models can produce factual incorrect information and might induce biases based on user prompts.
The employed featurizers, based on BERT models, have been trained exclusively on English language corpora, making them unsuitable and inconsistent when utilized with texts in other languages, potentially altering results negativly.
Mistral 7B does not include content moderation. We encourage anyone to use produced results and capabilities of Language Models in a responsible manner.

References

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, John A. Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuan-Fang Li, Scott M. Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *ArXiv*, abs/2303.12712.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen

Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

T. Cover and P. Hart. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

Antonia Creswell, Murray Shanahan, and Irina Higgins. 2022. Selection-inference: Exploiting large language models for interpretable logical reasoning. *ArXiv*, abs/2205.09712.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. Codebert: A pre-trained model for programming and natural languages.

Joseph Gatto, Omar Sharif, Parker Seegmiller, Philip Bohlman, and Sarah Masud Preum. 2023. Text encoders lack knowledge: Leveraging generative llms for domain-specific semantic textual similarity.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset.

Harold Hotelling. 1933. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6 7):417–441 498–520.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

I. T. Jolliffe. 2002. *Principal Component Analysis*, 2 edition. Springer Series in Statistics. Springer-Verlag New York, New York.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Annual Meeting of the Association for Computational Linguistics*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation : Learning to solve and explain algebraic word problems.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Larry M. Manevitz and Malik Yousef. 2002. One-class svms for document classification. *J. Mach. Learn. Res.*, 2:139–154.

Amil Merchant, Elahe Rahimtoroghi, Ellie Pavlick, and Ian Tenney. 2020. What happens to bert embeddings during fine-tuning?

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. 2023. Augmented language models: a survey.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2023. A comprehensive overview of large language models.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems?

Karl Pearson. 1901. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine, Series 6*, 2(11):559–572.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John F. J. Mellor, Irina Higgins, Antonia Creswell, Nathan McAleese, Amy Wu, Erich Elsen, Siddhant M. Jayakumar, Elena Buchatskaya, David

Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, L. Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, N. K. Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Tobias Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew G. Johnson, Blake A. Hechtman, Laura Weidinger, Iason Gabriel, William S. Isaac, Edward Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem W. Ayoub, Jeff Stanway, L. L. Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. 2021. Scaling language models: Methods, analysis & insights from training gopher. *ArXiv*, abs/2112.11446.

Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought.

Iqbal H Sarker. 2021. Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. *SN Comput Sci*, 2(6):420.

Jia Tracy Shen, Michiharu Yamashita, Ethan Prihar, Neil Heffernan, Xintao Wu, Ben Graff, and Dongwon Lee. 2023. Mathbert: A pre-trained language model for general nlp tasks in mathematics education.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2020. How to fine-tune bert for text classification?

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903.

Ori Yoran, Tomer Wolfson, Ben Bogin, Uri Katz, Daniel Deutch, and Jonathan Berant. 2023. Answering questions by meta-reasoning over multiple chains of thought.

11 Appendices

A Perplexity of generated Sequences

Table 11 illustrates that there is no apparent correlation between the performance of the models and their respective perplexity scores. A notable trend is the consistently *better* performance on the SVAMP dataset compared to AQuA-rat, likely attributable to the simpler nature of SVAMP's questions. Furthermore, the Mistral model exhibits a slightly superior performance, which can be ascribed to its higher accuracy across both datasets. This suggests that the confidence in the sequences remains robust, regardless of the model choice and accuracy.

Model	Dataset	Perplexity
SVAMP	Mistral	**0.1422**
SVAMP	LLAMA 2	0.1483
AQUA-rat	Mistral	0.1841
AQUA-rat	LLAMA 2	0.1861

Table 11: Perplexity Scores across different Models, "best" result is featured in bold.
Not evaluated on GPT-3.5 due to limited possibilities on the OpenAI public API.

B N-Gram Rationale Comparison

B.1 Rouge-N as a performance measure

Contrary to GPT-3.5's performance in terms of accuracy, it under performs in comparison when taking ROUGE metrics into account. As expected it excels in generating accurate, contextually relevant responses but expressed responses more detailed in a more comprehensive fashion, leading to lower ROUGE scores due to the strictly accurate less extensive rationale annotated in the dataset. (Lin, 2004)
The other Models like LLAMA 2 7B and Mistral 7B produce higher scores. This might be related to factors like style of writing and higher text length

which although it leads to more comprehensive embeddings lowers it's score when compared with a metric like *Rouge-N* as visible in Table 10

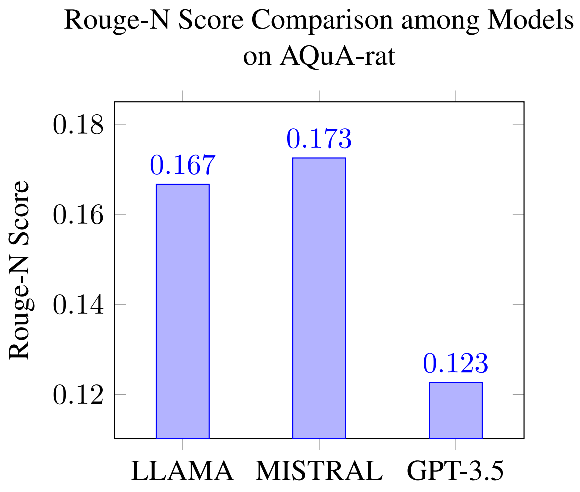

Figure 3: The ROUGE-N score was applied solely to the AQuA-rat dataset, as datasets like SVAMP provide numerical answers instead of sequential/textual rationales.

B.2 N-Gram weighting

N-Grams are often used for context understanding, aiding tasks like sentiment analysis and language modeling In our study, we used N-Grams to weigh their impact on results, testing different 'n' values to see how they affect accuracy outcomes.

Table 12: Weighting results based on N-Gram overlap with n = 2

Model	AQUARAT	SVAMP
LLAMA 2	15.5	32.8
MISTRAL	16.7	47.1
GPT3.5	25.3	63.9

The low accuracy and poor results, coupled with a degree of randomness in the result distribution, indicate challenges in effectively correlating text using N-Grams. We experimented with different values of 'n' for N-Grams, aiming to improve performance, but encountered limitations. As depicted in the table, the effectiveness of N-Grams varied, suggesting that the pure similar wording in rationales cant be utilized in an effective way to improve or even stably perform similar to the baseline. Higher values of "n" consecutively worsened results.

C Model configurations

Configurations may deviate slightly on GPT3.5 due to usage via the public API.

- top-k: 50
- top-p: 50
- sampling: true
- max-new-tokens: see Appendix C.1
- temperature: see Appendix D.1

C.1 max-new-tokens

We used a default of 150 max-new-tokens across all models on SVAMP, due to the complexity and length of sequences on AQuA-rat we chose 200 max-new-tokens. Due to the length of Code tasks we set the max generation of new tokens to 400 on humaneval.

D Abstract consistency

D.1 Temperature sets

We tested our theory of abstraction on a variety of temperature sets and found that ***set 1*** exhibits the best balance between diversity and correctness in our examples. Therefore, it outperforms the other proposed sets.

Set 1 (t)	Set 2 (t)	Set 3 (t)
0.9	0.7	0.5
0.8	0.6	0.4
0.7	0.5	0.3
0.6	0.4	0.2
0.5	0.3	0.1

Table 13: Each Temperature is tested on 1/5 of the samples per generation, to ensure an even distribution.

All other experiments have been conducted on a static *temperature* of **0.8** to aid reproducibility and comparability between results and effects of the employed mechanisms.

D.2 Weighing abstract consistency

We propose several methods for weighing sequences from different temperatures. Additionally, we employ a weighing system based on the inverse of the applied temperature. Furthermore, we conducted tests using weighted squared inverse weighting on a small subset. However, these tests did not yield substantially elevated results and performed on a similar margin.

Figure 4: Average

$$\sum_{i=1}^{n} \frac{1}{t_i} \quad (4)$$

Figure 5: Squared Average

$$\sum_{i=1}^{n} \left(\frac{1}{t_i}\right)^2 \quad (5)$$

E Used k-shot prompts

The used **8-Shot prompt** for mathematical reasoning follows the example provided in pg. 43 and use the on pg. 36 referenced set for AQuA on the AQuA-rat dataset with of the original Chain of thought paper.

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?
A: We start with 15 trees. Later we have 21 trees. The difference must be the number of trees they planted. So, they must have planted 21 - 15 = 6 trees. The answer is 6.
Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?
A: There are 3 cars in the parking lot already. 2 more arrive. Now there are 3 + 2 = 5 cars. The answer is 5.
Q: Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total?
A: Leah had 32 chocolates and Leah's sister had 42. That means there were originally 32 + 42 = 74 chocolates. 35 have been eaten. So in total they still have 74 - 35 = 39 chocolates. The answer is 39. Q: Jason had 20 lollipops. He gave Denny some lollipops. Now Jason has 12 lollipops. How many lollipops did Jason give to Denny?
A: Jason had 20 lollipops. Since he only has 12 now, he must have given the rest to Denny. The number of lollipops he has given to Denny must have been 20 - 12 = 8 lollipops. The answer is 8.
Q: Shawn has five toys. For Christmas, he got two toys each from his mom and dad. How many toys does he have now?
A: He has 5 toys. He got 2 from mom, so after that he has 5 + 2 = 7 toys. Then he got 2 more from dad, so in total he has 7 + 2 = 9 toys. The answer is 9.
Q: There were nine computers in the server room. Five more computers were installed each day, from monday to thursday. How many computers are now in the server room?
A: There are 4 days from monday to thursday. 5 computers were added each day. That means in total 4 * 5 = 20 computers were added. There were 9 computers in the beginning, so now there are 9 + 20 = 29 computers. The answer is 29.
Q: Michael had 58 golf balls. On tuesday, he lost 23 golf balls. On wednesday, he lost 2 more. How many golf balls did he have at the end of wednesday?
A: Michael initially had 58 balls. He lost 23 on Tuesday, so after that he has 58 - 23 = 35 balls. On Wednesday he lost 2 more so now he has 35 - 2 = 33 balls. The answer is 33.
Q: Olivia has $23. She bought five bagels for $3 each. How much money does she have left?
A: She bought 5 bagels for $3 each. This means she spent 5 * $3 = $15 on the bagels. She had $23 in beginning, so now she has $23 - $15 = $8. The answer is 8

Proposed 4-shot on AQuA(-rat):

Q: John found that the average of 15 numbers is 40. If 10 is added to each number then the mean of the numbers is?
Answer Choices: (a) 50 (b) 45 (c) 65 (d) 78 (e) 64
A: If 10 is added to each number, then the mean of the numbers also increases by 10. So the new mean would be
50. The answer is (a).
Q: If a / b = 3/4 and 8a + 5b = 22,then find the value of a.
Answer Choices: (a) 1/2 (b) 3/2 (c) 5/2 (d) 4/2 (e) 7/2
A: If a / b = 3/4, then b = 4a / 3. So 8a + 5(4a / 3) = 22. This simplifies to 8a + 20a / 3 = 22, which means 44a / 3 = 22. So a is equal to 3/2. The answer is (b).
Q: A person is traveling at 20 km/hr and reached his destiny in 2.5 hr then find the distance?
Answer Choices: (a) 53 km (b) 55 km (c) 52 km (d) 60 km (e) 50 km
A: The distance that the person traveled would have been 20 km/hr * 2.5 hrs = 50 km. The answer is (e).
Q: How many keystrokes are needed to type the numbers from 1 to 500?
Answer Choices: (a) 1156 (b) 1392 (c) 1480 (d) 1562 (e) 1788
A: There are 9 one-digit numbers from 1 to 9. There are 90 two-digit numbers from 10 to 99. There are 401 three-digit numbers from 100 to 500. 9 + 90(2) + 401(3) = 1392. The answer is (b).

Our generation on humaneval were conducted **0-shot** using just the raw prompt given by the dataset.

F Datasets

We use the configuration splits for testing as suggested by default. We employ a test split of 1000 samples on SVAMP and 1.3K for GSM8K. For AQuA-rat, our test includes the full set of 254 examples.

G K-means Clustering

Across our study we employed kmeans to cluster datapoints mapped by our featurizer model.

G.1 Clustering effects

Clustering has shown diminishing returns in terms of accuracy, however the herein provided evidence shows that clustering with k-means provides a notable advantages which even tho the accuracy was low can be used as a diagnostic tool and similarity measure

G.1.1 Silouhette score

We used the silhouette score to evaluate clustering effectiveness. This score measures how similar an object is to its own cluster compared to other clusters, ranging from -1 to 1.

Our obtained averaged silhouette score equals **0.41**, suggesting a moderate level of distinction between clusters. This range indicates that, on average, objects within a cluster are closer to each other than to objects in other clusters, but the separation is not highly distinct.

This finding suggests that clusters are indicating a clear structure in sentence and wording of results and due to Kmeans nature perform better on higher sample sizes.

G.1.2 Average correct datapoint proportion

Despite the fragility shown during evaluation on benchmarks, the k-means accurately categorizes the majority of correct answer within the preponderant cluster, not only based on cluster size. This implies that the method, even with limited data, captures essential patterns effectively.

High-performing models are more likely to adhere closely to the chosen method. This is because when most answers are correct, there's a lower chance of incorrect responses outweighing the correct ones, which could lead to inaccuracies.

Table 14: Proportion of correct responses in the majority cluster compared to total true responses.

Model	SVAMP	AQUA-rat
LLAMA 2	68.8	56.6
MISTRAL	66.2	46.2
GPT3.5	69.4	55.5

The shown results indicate a trend demonstrating that the selected cluster is more likely to feature the majority of correct responses, with an average of **60.5**%.

We witness the same strides towards higher sample sizes with the usage of k-means as already conveyed in the original self-consistency paper, here larger sample sizes might be able to capture the amount of correct answers in a more favorable manner due to their enabled potential for a higher number of clusters, capturing more nuanced and subtle variations rather than the broad range of responses.

G.1.3 Cluster density comparison

The primary cluster and the ostensibly weaker, later-disregarded cluster exhibit comparable performance in terms of the average distance of the data points to its subsequent cluster centroid.

Table 15: Average Deviation for clusters

Method	Model	Chosen cluster	Disregarded cluster
SVAMP	LLAMA	2.037	2.567
SVAMP	MISTRAL	2.981	3.800
SVAMP	GPT	4.428	4.513
AQuA-rat	LLAMA	0.838	0.670
AQuA-rat	MISTRAL	0.871	0.598
AQuA-rat	GPT3.5	3.649	3.684

G.2 Clustering results

Due to k-means inherent randomness during initialization, we average its performance over 10 runs.

Table 16: Results of LLAMA 2

SVAMP			AQuA-rat		
Run Number	**random state**	**Accuracy (%)**	**Run Number**	**random state**	**Accuracy (%)**
1	10	42.31	1	10	25.47
2	20	42.40	2	20	24.53
3	30	42.25	3	30	22.38
4	40	41.99	4	40	24.51
5	50	41.94	5	50	26.76
6	60	42.80	6	60	23.81
7	70	43.07	7	70	25.12
8	80	42.70	8	80	24.02
9	90	42.35	9	90	22.58
10	100	42.89	10	100	22.42

Table 17: Results of Mistral 7B

SVAMP			AQuA-rat		
Run Number	**random state**	**Accuracy (%)**	**Run Number**	**random state**	**Accuracy (%)**
1	10	62.72	1	10	23.18
2	20	62.45	2	20	23.11
3	30	62.74	3	30	24.77
4	40	61.88	4	40	25.45
5	50	62.46	5	50	25.93
6	60	62.22	6	60	26.39
7	70	62.15	7	70	25.00
8	80	61.69	8	80	26.51
9	90	63.04	9	90	25.24
10	100	63.85	10	100	22.73

Table 18: Results of GPT3.5

SVAMP			AQuA-rat		
Run Number	**random state**	**Accuracy (%)**	**Run Number**	**random state**	**Accuracy (%)**
1	10	78.56	1	10	68.07
2	20	79.06	2	20	70.28
3	30	78.86	3	30	65.32
4	40	78.66	4	40	66.82
5	50	78.86	5	50	66.67
6	60	78.07	6	60	69.71
7	70	79.36	7	70	66.67
8	80	78.36	8	80	67.79
9	90	78.56	9	90	68.72
10	100	78.36	10	100	65.12

H Outlier detection across different parameters

H.1 k-nearest neighbor results

In the k-nearest neighbor (KNN) algorithm, parameters such as the number of neighbors (n_neighbors), the distance metric (metric), and the algorithm used for computing nearest neighbors were varied. The best-performing configuration in terms of accuracy was found with **n_neighbors set to 5**, using the **euclidean metric** using the **ball_tree algorithm** and a **threshold of 90%** that concluded to an averaged accuracy of **56.18%** with all Models and Datasets.

H.2 Isolation forest results

For the Isolation Forest, the grid search varied parameters including the number of estimators (n_estimators), the contamination factor, and the max samples size. The configuration yielding the highest accuracy utilized **n_estimators=200**, **contamination=auto**, and **max_samples=auto** with an performance of **58.56%** averaged across all Models and Datasets.

H.3 support vector machines results

In the case of Support Vector Machines (SVM), the kernel type (kernel), the regularization parameter (nu), and the gamma value were among the parameters adjusted. The most accurate results were achieved with a **linear kernel**, **nu set to 0.01**, and **gamma set to scale**. The average accuracy was **55.17%**

I Abstract consistency on different temperature sets

Higher temperature in generative models introduces a degree of randomness that can negatively impact performance by increasing degeneration in model outputs. However, this limiting factor can be partially mitigated through techniques such as inverse temperature weighting. When applied appropriately alongside temperature variation. The benefits of higher temperature are not monotonic - beyond an optimal level, continuing to increase temperature will again degrade performance. There exists a sweet spot where judiciously elevated temperature and re-weighting allows models to produce greater diversity without excessive degradation which we found to lay between t = *0.5* and t = *0.9*.

J t-SNE

To emphasize the separation and clustering since it provides more visually informative representations that can aid in data exploration and pattern recognition tasks superior to PCA We select a perplexity parameter of 2, grounded in the fact that local distributions yield a more informative representation than global distributions.
This is attributed to the increased density of points in close proximity, enhancing the detail captured in the mapping.

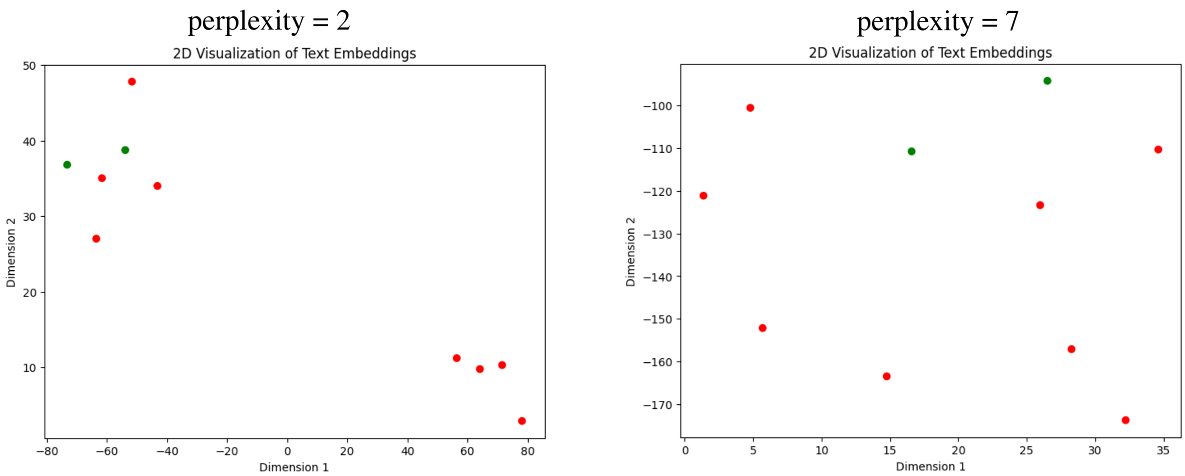

Figure 6: Based on a test on a subset of arithmetic reasoning examples, evaluated on 10, 15 and 20 generated outputs based on baseline self-consistency with the in Appendix E provided n-Shot prompts.