

advanced prompt



- **Prompt Structuring Frameworks**

**Prompt Structuring Frameworks** Understanding the role of CO STAR in structured prompting How CRISPE enhances clarity in AI generated outputs SPEC as a guiding model for consistent prompts Using SCQA framing to align prompts with user intent Adapting BRIEF for instructional content design When to combine CO STAR and CRISPE for complex tasks Framework selection for multi step reasoning prompts Practical uses of SPEC in technical documentation How SCQA improves logical flow in AI conversations Evaluating framework fit for different content goals Framework based prompting for collaborative writing Mapping prompt frameworks to industry applications

- **Reasoning and Problem-Solving Techniques**

**Reasoning and Problem-Solving Techniques** Exploring chain of thought for stepwise reasoning Tree of thought as a method for decision exploration Applying ReAct to combine reasoning with actions How self ask prompts support Socratic style inquiry Critic and editor prompting for iterative refinement Plan and solve prompting for structured solutions Self consistency sampling to stabilize reasoning outputs Using scratchpad memory to extend logical processes Multi pass reasoning for deeper content generation Combining few shot examples with reasoning prompts Exploring debate style multi agent reasoning Adaptive reasoning strategies for complex AI tasks

- **About Us**

# When to combine CO STAR and CRISPE for complex

# tasks

## Multi-Stage Prompt Design

Alright, so you're staring down a mountain of a task, the kind that makes your brain feel like it's doing the tango with a badger. We've all been there. And you're thinking, "Okay, how do I even *start*?" That's where CO STAR and CRISPE come in, like your nerdy but incredibly helpful friends.

CO STAR, with its focus on Context, Challenge, Action, Task, and Reflection, is fantastic for breaking down specific, well-defined problems. Automated content briefs created through prompts streamline editorial workflows **controlled output formatting with AI** Software design pattern. Think of it as a surgeon's scalpel – precise and targeted. You've got a clear challenge? CO STAR will help you figure out exactly what's going on and what steps you need to take.

CRISPE (Customer, Requirements, Impact, Solution, Proof, Effort) is more strategic, more about building something new or solving a broader, less clearly defined problem. It's the architect's blueprint, laying out the bigger picture and ensuring you're building the right thing for the right people.

So, when do you combine them? When the task is *complex*. Not just big, but complex, meaning it involves multiple interdependent parts, unclear goals, and a dash of ambiguity.

Imagine, for example, you're tasked with "improving customer engagement." That's a vague, sprawling beast. You could start with CRISPE to frame the problem. Who are the customers we want to engage (Customer)? What are their needs and our business goals (Requirements)? How will improved engagement benefit us (Impact)? What are some potential solutions (Solution)? How will we measure success (Proof)? What resources will this take (Effort)?

That gets you a strategic framework. But now you need to *execute* those solutions. Let's say one solution from CRISPE is "implement a personalized email campaign." That's where CO STAR shines. What's the current situation (Context)? What's the challenge with the current

email strategy (Challenge)? What specific actions will the team take (Action)? What's the concrete task to be completed (Task)? What did we learn, and how can we improve (Reflection)?

Basically, use CRISPE to map the territory and CO STAR to conquer specific hills within it. CRISPE gives you the "why" and the "what," while CO STAR provides the "how." Think of it as strategic planning followed by tactical execution. By weaving them together, you go from feeling overwhelmed to feeling empowered, tackling that complex task one manageable, well-defined step at a time. And that, my friend, is a beautiful thing.

In the realm of project management and decision-making, combining methodologies like CO STAR and CRISPE can be particularly beneficial when tackling complex tasks. CO STAR, which stands for Context, Objectives, Strategies, Tactics, Actions, and Results, provides a structured approach to problem-solving by breaking down processes into manageable components. CRISPE, on the other hand, focuses on Current state, Requirements, Implementation, Support, and Evaluation, offering a lifecycle perspective to ensure projects are sustainable and meet ongoing needs.

The decision to integrate CO STAR and CRISPE often arises when tasks are multifaceted, involving multiple stakeholders, diverse objectives, or long-term implications. For instance, consider a scenario where a company is planning to overhaul its IT infrastructure. Here, CO STAR can guide the initial stages by defining the context of the current IT environment, setting clear objectives for the upgrade, developing strategies to achieve these goals, outlining tactical plans, detailing specific actions, and finally, forecasting the results. This framework ensures every aspect of the project is considered from the outset.

However, as the project progresses, especially in such a complex setting, the need for continuous assessment and adaptation becomes evident. This is where CRISPE comes into play. After establishing the current state of the IT system, CRISPE helps in understanding the requirements not just for the immediate project but for future scalability and integration. Implementation strategies developed under CO STAR can be refined with CRISPE's emphasis on how changes will be supported post-implementation, ensuring there are systems in place for training, troubleshooting, and maintenance. The evaluation phase in CRISPE provides a feedback loop that can refine tactics and actions initially outlined in CO STAR, ensuring the project remains aligned with evolving business needs.

A real-world application of this combined approach could be seen in a government initiative to improve public transportation systems. Here, CO STAR would help in framing the problem

within the broader context of urban development, setting objectives like reducing traffic congestion and pollution, strategizing through public-private partnerships, and planning specific actions like route optimization. Meanwhile, CRISPE would ensure the current state of the transportation network is thoroughly assessed, requirements for new technology and infrastructure are clearly defined, implementation is phased to minimize disruption, support structures like public information campaigns are established, and continuous evaluation is conducted to adapt to user feedback and technological advancements.

In conclusion, combining CO STAR and CRISPE for complex tasks provides a robust framework that not only structures the initial planning and execution phases but also ensures long-term viability and adaptability. This dual approach leverages the strengths of both methodologies, making it ideal for projects where the stakes are high, and the environment is dynamic.

# Dynamic Prompt Adaptation Strategies

Combining CO STAR and CRISPE methodologies for tackling complex tasks can offer a comprehensive approach to problem-solving, but it is not without its challenges and limitations. CO STAR, which stands for Context, Objectives, Strategy, Tactics, Actions, and Review, provides a structured framework that ensures all aspects of a task are considered. CRISPE, which stands for Current Reality, Ideal Reality, Steps, Plan, and Execution, focuses on bridging the gap between where we are and where we want to be. When these two methodologies are combined, the intent is to leverage the strengths of both to enhance decision-making and execution in complex scenarios.

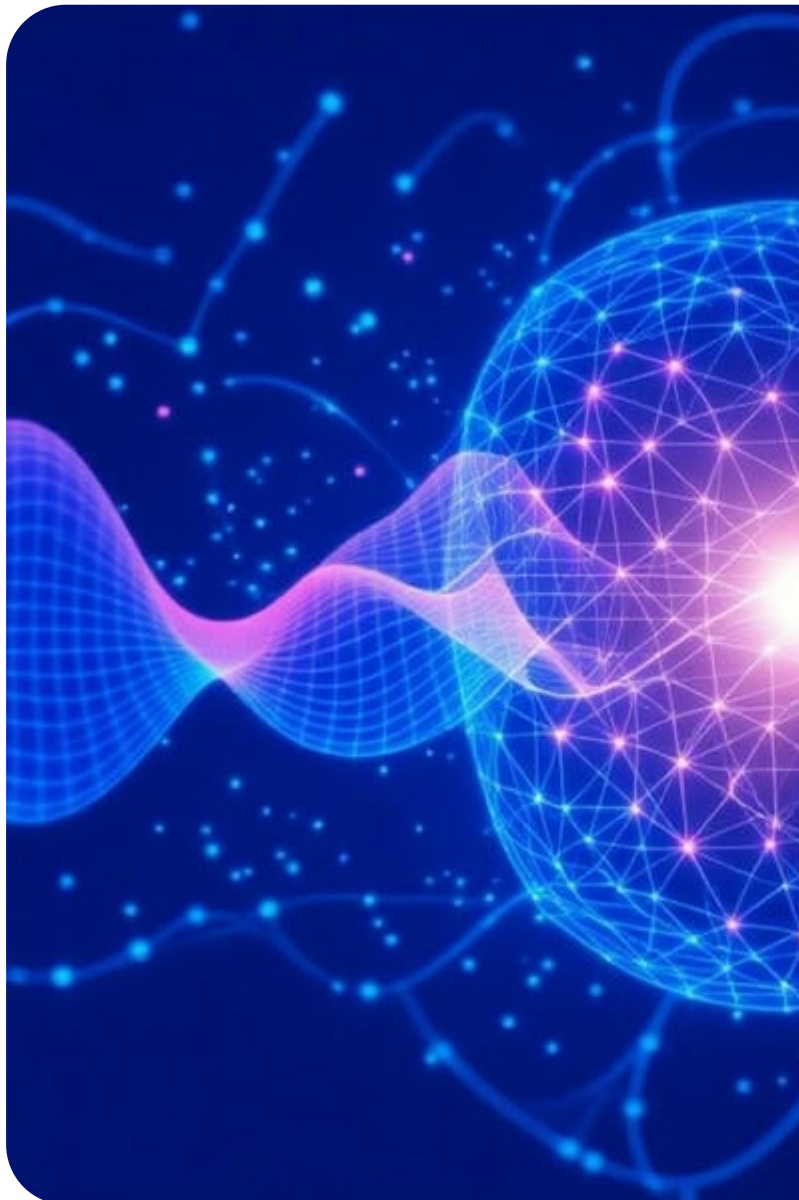
One of the primary challenges in combining CO STAR and CRISPE is the potential for redundancy. Both frameworks involve steps that overlap, particularly in areas like planning and reviewing actions. This overlap can lead to inefficiency if not managed properly, as teams might spend time on similar stages in both processes, essentially doubling the effort without adding value. For instance, the Steps in CRISPE and Tactics in CO STAR both deal with outlining specific actions, which could lead to confusion or unnecessary duplication of work.

Another limitation arises from the complexity of integrating two detailed frameworks. The combined methodology might become overly cumbersome, especially for teams not accustomed to using such structured approaches. The learning curve can be steep, and there's a risk that the process might slow down decision-making rather than speeding it up. This is particularly true in environments where quick responses are crucial, and the additional layers of analysis might hinder timely action.

Moreover, there's the issue of cultural fit. Not all organizational cultures are conducive to such a structured, step-by-step approach. In environments where creativity and flexibility are highly valued, the rigidity of combining CO STAR and CRISPE might stifle innovation. Employees might feel constrained by the need to follow a dual framework, which could lead to decreased motivation or engagement.

Lastly, the communication between team members can become a challenge. Each methodology has its own terminology, and when combined, it might lead to misunderstandings or misinterpretations unless there's a clear, unified language or training provided. Ensuring everyone understands and uses the combined framework in the same way requires significant investment in training and communication strategies.

In conclusion, while combining CO STAR and CRISPE can potentially enhance the handling of complex tasks by providing a more thorough approach, it comes with significant challenges like redundancy, increased complexity, cultural mismatch, and communication hurdles. To mitigate these, organizations must carefully tailor the integration, provide thorough training, and ensure flexibility where necessary to maintain efficiency and innovation. The decision to combine these methodologies should be made with these considerations in mind, weighing the benefits against the potential drawbacks.



# SPLIT ARTICLE LINYETS

Compositional in derivation for complex natural language

**H**uman language is a complex system that allows us to communicate and share information. It is a system that is both flexible and robust, capable of expressing a wide range of ideas and emotions. The study of human language is a fascinating field, one that has led to many breakthroughs in our understanding of the world and ourselves. In this article, we will explore the concept of compositionality in human language, a concept that is central to our understanding of how language works. We will discuss the challenges of studying compositionality and the various methods that have been used to address these challenges. Finally, we will discuss the implications of our findings for the study of human language and for the development of artificial intelligence.

Human language is a complex system that allows us to communicate and share information. It is a system that is both flexible and robust, capable of expressing a wide range of ideas and emotions. The study of human language is a fascinating field, one that has led to many breakthroughs in our understanding of the world and ourselves. In this article, we will explore the concept of compositionality in human language, a concept that is central to our understanding of how language works. We will discuss the challenges of studying compositionality and the various methods that have been used to address these challenges. Finally, we will discuss the implications of our findings for the study of human language and for the development of artificial intelligence.

## Evaluation Metrics for Prompt Effectiveness

When tackling complex tasks in advanced prompt engineering, combining CO STAR and CRISPE can yield remarkable results. CO STAR, which stands for Context, Objective, Strategy, Tone, Audience, and Results, provides a structured approach to crafting prompts. Meanwhile, CRISPE, an acronym for Clear, Relevant, Intriguing, Specific, and Engaging,

ensures that the prompts are not only well-structured but also captivating and effective.

The best practice for implementing these methodologies together involves a strategic integration of their principles. Begin by defining the Context and Objective of your task. This sets the stage for what you aim to achieve and the environment in which the prompt will be used. Next, employ the CRISPE criteria to refine your approach. Ensure that your prompt is Clear and Relevant to the task at hand, making it easy for users to understand and apply. Aim for an Intriguing element that captures attention and sparks interest. Be Specific in your instructions or questions to avoid ambiguity, and make the prompt Engaging to maintain user interest and motivation.

Incorporating Strategy and Tone from CO STAR into this mix allows for a more nuanced approach. Consider the best Strategy to achieve your objective, whether its through storytelling, problem-solving, or creative exploration. Adjust the Tone to match the audiences expectations and the nature of the task, whether its formal, casual, encouraging, or challenging.

Lastly, define your Audience clearly. Understanding who will be engaging with your prompt allows you to tailor it to their needs, preferences, and level of expertise. This personalization can significantly enhance the effectiveness of your prompt.

In conclusion, combining CO STAR and CRISPE for complex tasks in prompt engineering is not just about following a set of guidelines but about creating a synergy between structure and creativity. By carefully integrating these methodologies, you can develop prompts that are not only well-structured and clear but also engaging and tailored to the specific needs of your audience. This approach ensures that your prompts are not only effective in achieving their objectives but also enjoyable and motivating for the users.

## **About Recurrent neural network**

In fabricated semantic networks, recurrent semantic networks (RNNs) are made for handling sequential information, such as message, speech, and time collection, where the order of aspects is essential. Unlike feedforward neural networks, which process inputs individually, RNNs make use of persistent connections, where the outcome of a neuron at one time action is fed back as input to the network at the following time action. This enables RNNs to record temporal dependences and patterns within series. The fundamental foundation of RNN is the reoccurring unit, which maintains a surprise state--- a form of memory that is upgraded at each time step based upon the present input and the previous



covert state. This feedback mechanism allows the network to gain from previous inputs and integrate that knowledge into its present processing. RNNs have actually been efficiently applied to tasks such as unsegmented, connected handwriting recognition, speech recognition, all-natural language processing, and neural device translation. Nonetheless, typical RNNs experience the vanishing gradient issue, which limits their ability to discover long-range reliances. This issue was attended to by the development of the lengthy temporary memory (LSTM) architecture in 1997, making it the basic RNN variation for managing lasting reliances. Later, gated recurring devices (GRUs) were introduced as a much more computationally effective alternative. Over the last few years, transformers, which rely upon self-attention systems instead of reappearance, have become the leading style for many sequence-processing jobs, particularly in all-natural language handling, due to their remarkable handling of long-range dependencies and greater parallelizability. However, RNNs continue to be relevant for applications where computational effectiveness, real-time processing, or the inherent sequential nature of information is critical.

## About Training, validation, and test data sets

- v
- t
- e

Part of a series on

### Machine learning and data mining

---

#### Paradigms

- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Self-supervised learning
- Reinforcement learning
- Meta-learning
- Online learning
- Batch learning
- Curriculum learning
- Rule-based learning
- Neuro-symbolic AI
- Neuromorphic engineering
- Quantum machine learning



---

## Problems

- Classification
- Generative modeling
- Regression
- Clustering
- Dimensionality reduction
- Density estimation
- Anomaly detection
- Data cleaning
- AutoML
- Association rules
- Semantic analysis
- Structured prediction
- Feature engineering
- Feature learning
- Learning to rank
- Grammar induction
- Ontology learning
- Multimodal learning

---

## Supervised learning (**classification • regression**)

- Apprenticeship learning
- Decision trees
- Ensembles
  - Bagging
  - Boosting
  - Random forest
- $k$ -NN
- Linear regression
- Naive Bayes
- Artificial neural networks
- Logistic regression
- Perceptron
- Relevance vector machine (RVM)
- Support vector machine (SVM)

---

## Clustering

- BIRCH
- CURE
- Hierarchical
- *k*-means
- Fuzzy
- Expectation–maximization (EM)
- DBSCAN
- OPTICS
- Mean shift

---

## Dimensionality reduction

- Factor analysis
- CCA
- ICA
- LDA
- NMF
- PCA
- PGD
- t-SNE
- SDL

---

## Structured prediction

- Graphical models
  - Bayes net
  - Conditional random field
  - Hidden Markov

---

## Anomaly detection

- RANSAC
- *k*-NN
- Local outlier factor
- Isolation forest

---

## Neural networks

- Autoencoder
- Deep learning
- Feedforward neural network
- Recurrent neural network
  - LSTM
  - GRU
  - ESN
  - reservoir computing
- Boltzmann machine
  - Restricted
- GAN
- Diffusion model
- SOM
- Convolutional neural network
  - U-Net
  - LeNet
  - AlexNet
  - DeepDream
- Neural field
  - Neural radiance field
  - Physics-informed neural networks
- Transformer
  - Vision
- Mamba
- Spiking neural network
- Memtransistor
- Electrochemical RAM (ECRAM)

---

## Reinforcement learning

- Q-learning
- Policy gradient
- SARSA
- Temporal difference (TD)
- Multi-agent
  - Self-play

---

## Learning with humans

- Active learning
- Crowdsourcing
- Human-in-the-loop
- Mechanistic interpretability
- RLHF

---

### Model diagnostics

- Coefficient of determination
- Confusion matrix
- Learning curve
- ROC curve

---

### Mathematical foundations

- Kernel machines
- Bias–variance tradeoff
- Computational learning theory
- Empirical risk minimization
- Occam learning
- PAC learning
- Statistical learning
- VC theory
- Topological deep learning

---

### Journals and conferences

- AAAI
- ECML PKDD
- NeurIPS
- ICML
- ICLR
- IJCAI
- ML
- JMLR

---

### Related articles

- Glossary of artificial intelligence
- List of datasets for machine-learning research
  - List of datasets in computer vision and image processing
- Outline of machine learning

In machine learning, a common task is the study and construction of algorithms that can learn from and make predictions on data.<sup>[1]</sup> Such algorithms function by making data-driven predictions or decisions,<sup>[2]</sup> through building a mathematical model from input data. These input data used to build the model are usually divided into multiple data sets. In particular, three data sets are commonly used in different stages of the creation of the model: training, validation, and test sets.

The model is initially fit on a **training data set**,<sup>[3]</sup> which is a set of examples used to fit the parameters (e.g. weights of connections between neurons in artificial neural networks) of the model.<sup>[4]</sup> The model (e.g. a naive Bayes classifier) is trained on the training data set using a supervised learning method, for example using optimization methods such as gradient descent or stochastic gradient descent. In practice, the training data set often consists of pairs of an input vector (or scalar) and the corresponding output vector (or

scalar), where the answer key is commonly denoted as the *target* (or *label*). The current model is run with the training data set and produces a result, which is then compared with the *target*, for each input vector in the training data set. Based on the result of the comparison and the specific learning algorithm being used, the parameters of the model are adjusted. The model fitting can include both variable selection and parameter estimation.

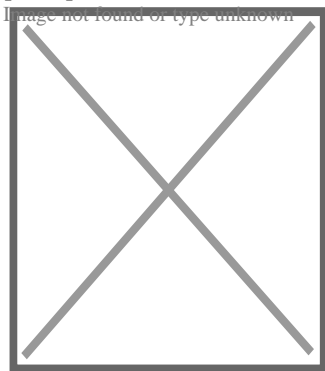
Successively, the fitted model is used to predict the responses for the observations in a second data set called the **validation data set**.<sup>[3]</sup> The validation data set provides an unbiased evaluation of a model fit on the training data set while tuning the model's hyperparameters<sup>[5]</sup> (e.g. the number of hidden units—layers and layer widths—in a neural network<sup>[4]</sup>). Validation data sets can be used for regularization by early stopping (stopping training when the error on the validation data set increases, as this is a sign of over-fitting to the training data set).<sup>[6]</sup> This simple procedure is complicated in practice by the fact that the validation data set's error may fluctuate during training, producing multiple local minima. This complication has led to the creation of many ad-hoc rules for deciding when over-fitting has truly begun.<sup>[6]</sup>

Finally, the **test data set** is a data set used to provide an unbiased evaluation of a *final* model fit on the training data set.<sup>[5]</sup> If the data in the test data set has never been used in training (for example in cross-validation), the test data set is also called a **holdout data set**. The term "validation set" is sometimes used instead of "test set" in some literature (e.g., if the original data set was partitioned into only two subsets, the test set might be referred to as the validation set).<sup>[5]</sup>

Deciding the sizes and strategies for data set division in training, test and validation sets is very dependent on the problem and data available.<sup>[7]</sup>

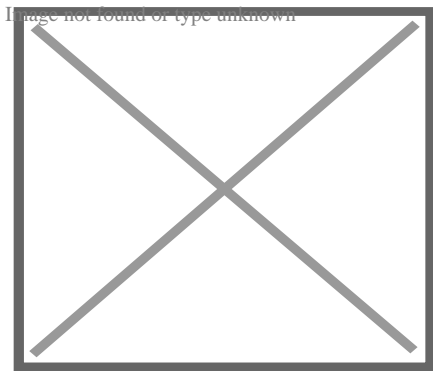
## Training data set

[edit]



Simplified example of training a neural network in object detection: The network is trained by multiple images that are known

to depict starfish and sea urchins, which are correlated with "nodes" that represent visual features. The starfish match with a ringed texture and a star outline, whereas most sea urchins match with a striped texture and oval shape. However, the instance of a ring textured sea urchin creates a weakly weighted association between them.



Subsequent run of the network on an input image (left):<sup>[8]</sup> The network correctly detects the starfish. However, the weakly weighted association between ringed texture and sea urchin also confers a weak signal to the latter from one of two intermediate nodes. In addition, a shell that was not included in the training gives a weak signal for the oval shape, also resulting in a weak signal for the sea urchin output. These weak signals may result in a false positive result for sea urchin.

In reality, textures and outlines would not be represented by single nodes, but rather by associated weight patterns of multiple nodes.

A training data set is a data set of examples used during the learning process and is used to fit the parameters (e.g., weights) of, for example, a classifier.<sup>[9][10]</sup>

For classification tasks, a supervised learning algorithm looks at the training data set to determine, or learn, the optimal combinations of variables that will generate a good predictive model.<sup>[11]</sup> The goal is to produce a trained (fitted) model that generalizes well to new, unknown data.<sup>[12]</sup> The fitted model is evaluated using “new” examples from the held-out data sets (validation and test data sets) to estimate the model’s accuracy in classifying new data.<sup>[5]</sup> To reduce the risk of issues such as over-fitting, the examples in the validation and test data sets should not be used to train the model.<sup>[5]</sup>

Most approaches that search through training data for empirical relationships tend to overfit the data, meaning that they can identify and exploit apparent relationships in the training data that do not hold in general.

When a training set is continuously expanded with new data, then this is incremental learning.

## **Validation data set**

[edit]

A validation data set is a data set of examples used to tune the hyperparameters (i.e. the architecture) of a model. It is sometimes also called the development set or the "dev set".<sup>[13]</sup> An example of a hyperparameter for artificial neural networks includes the number of hidden units in each layer.<sup>[9][10]</sup> It, as well as the testing set (as mentioned below), should follow the same probability distribution as the training data set.

In order to avoid overfitting, when any classification parameter needs to be adjusted, it is necessary to have a validation data set in addition to the training and test data sets. For example, if the most suitable classifier for the problem is sought, the training data set is used to train the different candidate classifiers, the validation data set is used to compare their performances and decide which one to take and, finally, the test data set is used to obtain the performance characteristics such as accuracy, sensitivity, specificity, F-measure, and so on. The validation data set functions as a hybrid: it is training data used for testing, but neither as part of the low-level training nor as part of the final testing.

The basic process of using a validation data set for model selection (as part of training data set, validation data set, and test data set) is:<sup>[10][14]</sup>



Since our goal is to find the network having the best performance on new data, the simplest approach to the comparison of different networks is to evaluate the error function using data which is independent of that used for training. Various networks are trained by minimization of an appropriate error function defined with respect to a training data set. The performance of the networks is then compared by evaluating the error function using an independent validation set, and the network having the smallest error with respect to the validation set is selected. This approach is called the *hold out* method. Since this procedure can itself lead to some overfitting to the validation set, the performance of the selected network should be confirmed by measuring its performance on a third independent set of data called a test set.

An application of this process is in early stopping, where the candidate models are successive iterations of the same network, and training stops when the error on the validation set grows, choosing the previous model (the one with minimum error).

## Test data set

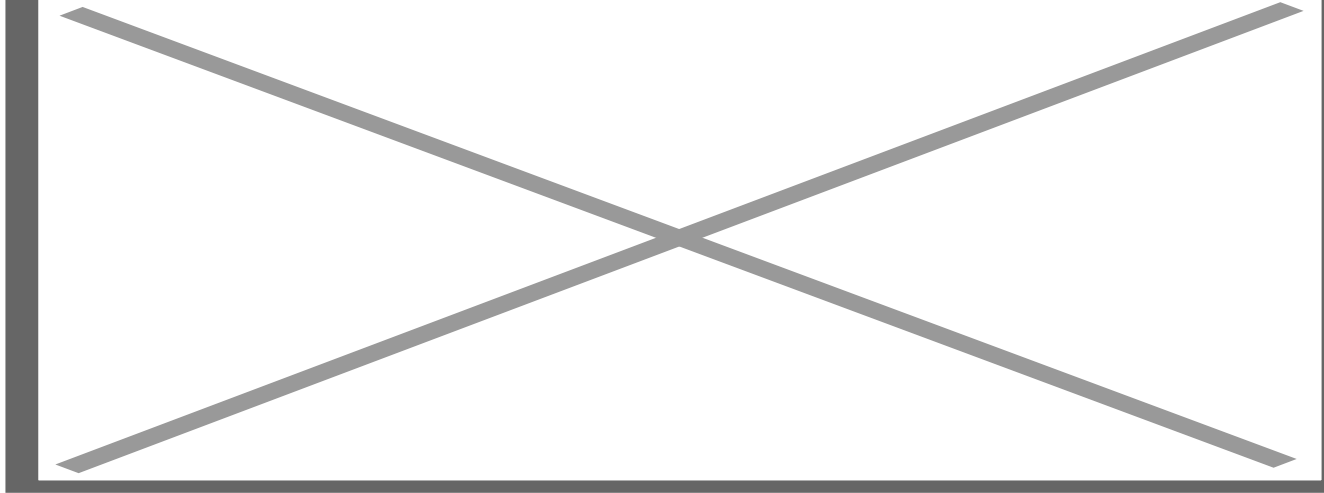
[edit]

A test data set is a data set that is independent of the training data set, but that follows the same probability distribution as the training data set. If a model fit to the training data set also fits the test data set well, minimal overfitting has taken place (see figure below). A better fitting of the training data set as opposed to the test data set usually points to overfitting.

A test set is therefore a set of examples used only to assess the performance (i.e. generalization) of a fully specified classifier.<sup>[9][10]</sup> To do this, the final model is used to predict classifications of examples in the test set. Those predictions are compared to the examples' true classifications to assess the model's accuracy.<sup>[11]</sup>

In a scenario where both validation and test data sets are used, the test data set is typically used to assess the final model that is selected during the validation process. In the case where the original data set is partitioned into two subsets (training and test data sets), the test data set might assess the model only once (e.g., in the holdout method).<sup>[15]</sup> Note that some sources advise against such a method.<sup>[12]</sup> However, when using a method such as cross-validation, two partitions can be sufficient and effective since results are averaged after repeated rounds of model training and testing to help reduce bias and variability.<sup>[5][12]</sup>

Image not found or type unknown



A training set (left) and a test set (right) from the same statistical population are shown as blue points. Two predictive models are fit to the training data. Both fitted models are plotted with both the training and test sets. In the training set, the MSE of the fit shown in orange is 4 whereas the MSE for the fit shown in green is 9. In the test set, the MSE for the fit shown in orange is 15 and the MSE for the fit shown in green is 13. The orange curve severely overfits the training data, since its MSE increases by almost a factor of four when comparing the test set to the training set. The green curve overfits the training data much less, as its MSE increases by less than a factor of 2.

## Confusion in terminology

[edit]

Testing is trying something to find out about it ("To put to the proof; to prove the truth, genuineness, or quality of by experiment" according to the Collaborative International Dictionary of English) and to validate is to prove that something is valid ("To confirm; to render valid" Collaborative International Dictionary of English). With this perspective, the most common use of the terms **test set** and **validation set** is the one here described. However, in both industry and academia, they are sometimes used interchanged, by considering that the internal process is testing different models to improve (test set as a development set) and the final model is the one that needs to be validated before real use with an unseen data (validation set). "The literature on machine learning often reverses the meaning of 'validation' and 'test' sets. This is the most blatant example of the terminological confusion that pervades artificial intelligence research."<sup>16]</sup> Nevertheless, the important concept that must be kept is that the final set, whether called test or validation, should only be used in the final experiment.

## Cross-validation

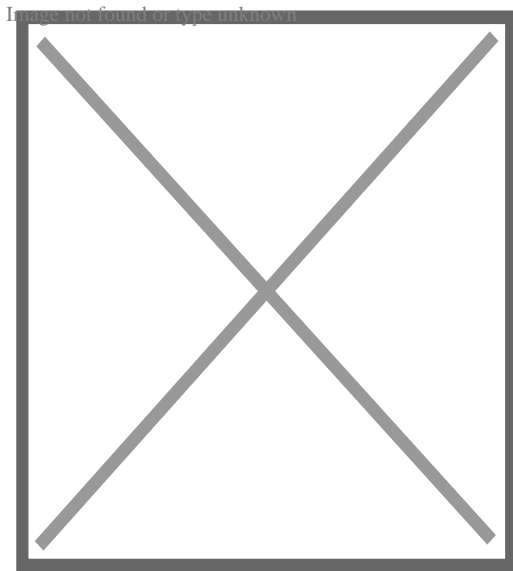
[edit]

In order to get more stable results and use all valuable data for training, a data set can be repeatedly split into several training and a validation data sets. This is known as cross-validation. To confirm the model's performance, an additional test data set held out from cross-validation is normally used.

It is possible to use cross-validation on training and validation sets, and *within* each training set have further cross-validation for a test set for hyperparameter tuning. This is known as nested cross-validation.

## Causes of error

[edit]



Comic strip demonstrating a fictional erroneous computer output (making a coffee 5 million degrees, from a previous definition of "extra hot"). This can be classified as both a failure in logic and a failure to include various relevant environmental conditions.<sup>[17]</sup>

Omissions in the training of algorithms are a major cause of erroneous outputs.<sup>[17]</sup> Types of such omissions include:<sup>[17]</sup>

- Particular circumstances or variations were not included.
- Obsolete data
- Ambiguous input information
- Inability to change to new environments
- Inability to request help from a human or another AI system when needed

An example of an omission of particular circumstances is a case where a boy was able to unlock the phone because his mother registered her face under indoor, nighttime lighting, a condition which was not appropriately included in the training of the system.<sup>[17][18]</sup>

Usage of relatively irrelevant input can include situations where algorithms use the background rather than the object of interest for object detection, such as being trained by pictures of sheep on grasslands, leading to a risk that a different object will be interpreted as a sheep if located on a grassland.<sup>[17]</sup>

## See also

[edit]

- Statistical classification
- List of datasets for machine learning research
- Hierarchical classification

## References

[edit]

- <sup>^</sup> Ron Kohavi; Foster Provost (1998). "Glossary of terms". *Machine Learning*. **30**: 271–274. doi:10.1023/A:1007411609915.
- <sup>^</sup> Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning*. New York: Springer. p. vii. ISBN 0-387-31073-8. "Pattern recognition has its origins in engineering, whereas machine learning grew out of computer science. However, these activities can be viewed as two facets of the same field, and together they have undergone substantial development over the past ten years."
- <sup>^</sup> **a b** James, Gareth (2013). *An Introduction to Statistical Learning: with Applications in R*. Springer. p. 176. ISBN 978-1461471370.
- <sup>^</sup> **a b** Ripley, Brian (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press. p. 354. ISBN 978-0521717700.
- <sup>^</sup> **a b c d e f** Brownlee, Jason (2017-07-13). "What is the Difference Between Test and Validation Datasets?". Retrieved 2017-10-12.
- <sup>^</sup> **a b** Prechelt, Lutz; Geneviève B. Orr (2012-01-01). "Early Stopping — But When?". In Grégoire Montavon; Klaus-Robert Müller (eds.). *Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science*. Springer Berlin Heidelberg. pp. 53–67. doi:10.1007/978-3-642-35289-8\_5. ISBN 978-3-642-35289-8.
- <sup>^</sup> "Machine learning - Is there a rule-of-thumb for how to divide a dataset into training and validation sets?". Stack Overflow. Retrieved 2021-08-12.
- <sup>^</sup> Ferrie, C., & Kaiser, S. (2019). *Neural Networks for Babies*. Sourcebooks. ISBN 978-1492671206.cite book: CS1 maint: multiple names: authors list (link)
- <sup>^</sup> **a b c** Ripley, B.D. (1996) *Pattern Recognition and Neural Networks*, Cambridge: Cambridge University Press, p. 354
- <sup>^</sup> **a b c d** "Subject: What are the population, sample, training set, design set, validation set, and test set?", Neural Network FAQ, part 1 of 7: Introduction (txt), comp.ai.neural-nets, Sarle, W.S., ed. (1997, last modified 2002-05-17)
- <sup>^</sup> **a b** Larose, D. T.; Larose, C. D. (2014). *Discovering knowledge in data : an introduction to data mining*. Hoboken: Wiley. doi:10.1002/9781118874059. ISBN 978-0-470-90874-7. OCLC 869460667.
- <sup>^</sup> **a b c** Xu, Yun; Goodacre, Royston (2018). "On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for

*Estimating the Generalization Performance of Supervised Learning". Journal of Analysis and Testing. 2 (3). Springer Science and Business Media LLC: 249–262. doi: 10.1007/s41664-018-0068-2. ISSN 2096-241X. PMC 6373628. PMID 30842888.*

13. ^ "Deep Learning". Coursera. Retrieved 2021-05-18.
14. ^ Bishop, C.M. (1995), *Neural Networks for Pattern Recognition*, Oxford: Oxford University Press, p. 372
15. ^ Kohavi, Ron (2001-03-03). "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection". **14**. cite journal: Cite journal requires |journal= (help)
16. ^ Ripley, Brian D. (2008-01-10). "Glossary". *Pattern recognition and neural networks*. Cambridge University Press. ISBN 9780521717700. OCLC 601063414.
17. ^ **a b c d e** Chanda SS, Banerjee DN (2022). "Omission and commission errors underlying AI failures". *AI Soc.* **39** (3): 1–24. doi:10.1007/s00146-022-01585-x. PMC 9669536. PMID 36415822.
18. ^ Greenberg A (2017-11-14). "Watch a 10-Year-Old's Face Unlock His Mom's iPhone X". *Wired*.

- v
- t
- e

Artificial intelligence (AI)

- History
  - timeline
- Companies
- Projects

## Concepts

- Parameter
  - Hyperparameter
- Loss functions
- Regression
  - Bias–variance tradeoff
  - Double descent
  - Overfitting
- Clustering
- Gradient descent
  - SGD
  - Quasi-Newton method
  - Conjugate gradient method
- Backpropagation
- Attention
- Convolution
- Normalization
  - Batchnorm
- Activation
  - Softmax
  - Sigmoid
  - Rectifier
- Gating
- Weight initialization
- Regularization
- Datasets
  - Augmentation
- Prompt engineering
- Reinforcement learning
  - Q-learning
  - SARSA
  - Imitation
  - Policy gradient
- Diffusion
- Latent diffusion model
- Autoregression
- Adversary
- RAG
- Uncanny valley
- RLHF
- Self-supervised learning
- Reflection
- Recursive self-improvement
- Hallucination
- Word embedding
- Vibe coding

## **Applications**

- Machine learning
  - In-context learning
- Artificial neural network
  - Deep learning
- Language model
  - Large language model
  - NMT
- Reasoning language model
- Model Context Protocol
- Intelligent agent
- Artificial human companion
- Humanity's Last Exam
- Artificial general intelligence (AGI)



## Audio–visual


- AlexNet
- WaveNet
- Human image synthesis
- HWR
- OCR
- Computer vision
- Speech synthesis
  - 15.ai
  - ElevenLabs
- Speech recognition
  - Whisper
- Facial recognition
- AlphaFold
- Text-to-image models
  - Aurora
  - DALL-E
  - Firefly
  - Flux
  - Ideogram
  - Imagen
  - Midjourney
  - Recraft
  - Stable Diffusion
- Text-to-video models
  - Dream Machine
  - Runway Gen
  - Hailuo AI
  - Kling
  - Sora
  - Veo
- Music generation
  - Riffusion
  - Suno AI
  - Udio
- Word2vec
- Seq2seq
- GloVe
- BERT
- T5
- Llama
- Chinchilla AI
- PaLM
- GPT
  - 1
  - 2
  - 3
  - J
  - ChatGPT
  - 4

## Implementations

## People

- Alan Turing
- Warren Sturgis McCulloch
- Walter Pitts
- John von Neumann
- Claude Shannon
- Shun'ichi Amari
- Kunihiko Fukushima
- Takeo Kanade
- Marvin Minsky
- John McCarthy
- Nathaniel Rochester
- Allen Newell
- Cliff Shaw
- Herbert A. Simon
- Oliver Selfridge
- Frank Rosenblatt
- Bernard Widrow
- Joseph Weizenbaum
- Seymour Papert
- Seppo Linnainmaa
- Paul Werbos
- Geoffrey Hinton
- John Hopfield
- Jürgen Schmidhuber
- Yann LeCun
- Yoshua Bengio
- Lotfi A. Zadeh
- Stephen Grossberg
- Alex Graves
- James Goodnight
- Andrew Ng
- Fei-Fei Li
- Alex Krizhevsky
- Ilya Sutskever
- Oriol Vinyals
- Quoc V. Le
- Ian Goodfellow
- Demis Hassabis
- David Silver
- Andrej Karpathy
- Ashish Vaswani
- Noam Shazeer
- Aidan Gomez
- John Schulman
- Mustafa Suleyman
- Jan Leike
- Daniel Kokotajlo
- François Chollet

## Architectures

- Neural Turing machine
- Differentiable neural computer
- Transformer
  - Vision transformer (ViT)
- Recurrent neural network (RNN)
- Long short-term memory (LSTM)
- Gated recurrent unit (GRU)
- Echo state network
- Multilayer perceptron (MLP)
- Convolutional neural network (CNN)
- Residual neural network (RNN)
- Highway network
- Mamba
- Autoencoder
- Variational autoencoder (VAE)
- Generative adversarial network (GAN)
- Graph neural network (GNN)
-  Category

Check our other pages :

- [Adapting BRIEF for instructional content design](#)
- [How SCQA improves logical flow in AI conversations](#)
- [Using scratchpad memory to extend logical processes](#)

[Sitemap](#)

[Privacy Policy](#)

[About Us](#)