

SAY WHAT?

IDENTIFYING THE IMPACT OF PROMPT TECHNIQUE ON AI GENERATION OF SYSTEMS ENGINEERING ARTIFACTS

Erin Smith Crabb, Matthew Jones, and Cedric Bernard

Leidos

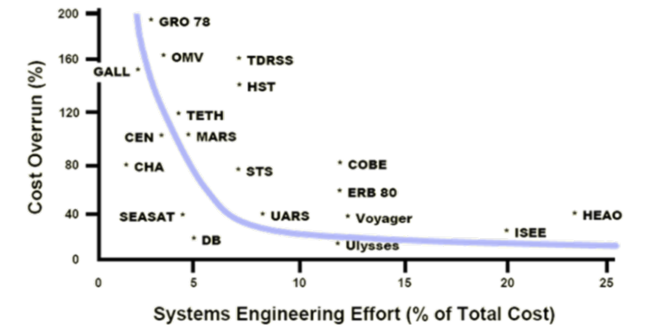
OUTLINE

- The Problem
- Toward Generative AI as Systems Engineering Assistants
- What is Prompt Engineering?
 - The Effect of Temperature
 - Model Selection
- The Experiment
- Findings
 - Sample Problem
 - Study Results
 - LLM Best Practices Summary

THE PROBLEM

- The demand for systems engineering exceeds the available practitioners. Current systems engineering activities are labor and tools intensive, and frequently require specialized domain knowledge.
- Pain Points:
 1. Customers expect faster delivery cycles (model-based delivery)
 2. Text (Stakeholder Needs and Requirements) must be transformed into models that represent the necessary parts, actions, and integrations of a product or system
 1. Input and output spaces are large (lots of artifacts), and it is hard for humans to remember every detail, especially when faced with unfamiliar domains

Cost Overrun as a Function of Systems Engineering Effort



Moody, et al, *Metrics and Case Studies for Evaluating Engineering Designs*

TOWARD LLMS AS SYSTEMS ENGINEERING ASSISTANTS

Results of Smith Crabb and Jones (2024) - experiment on using LLM for Systems Engineering

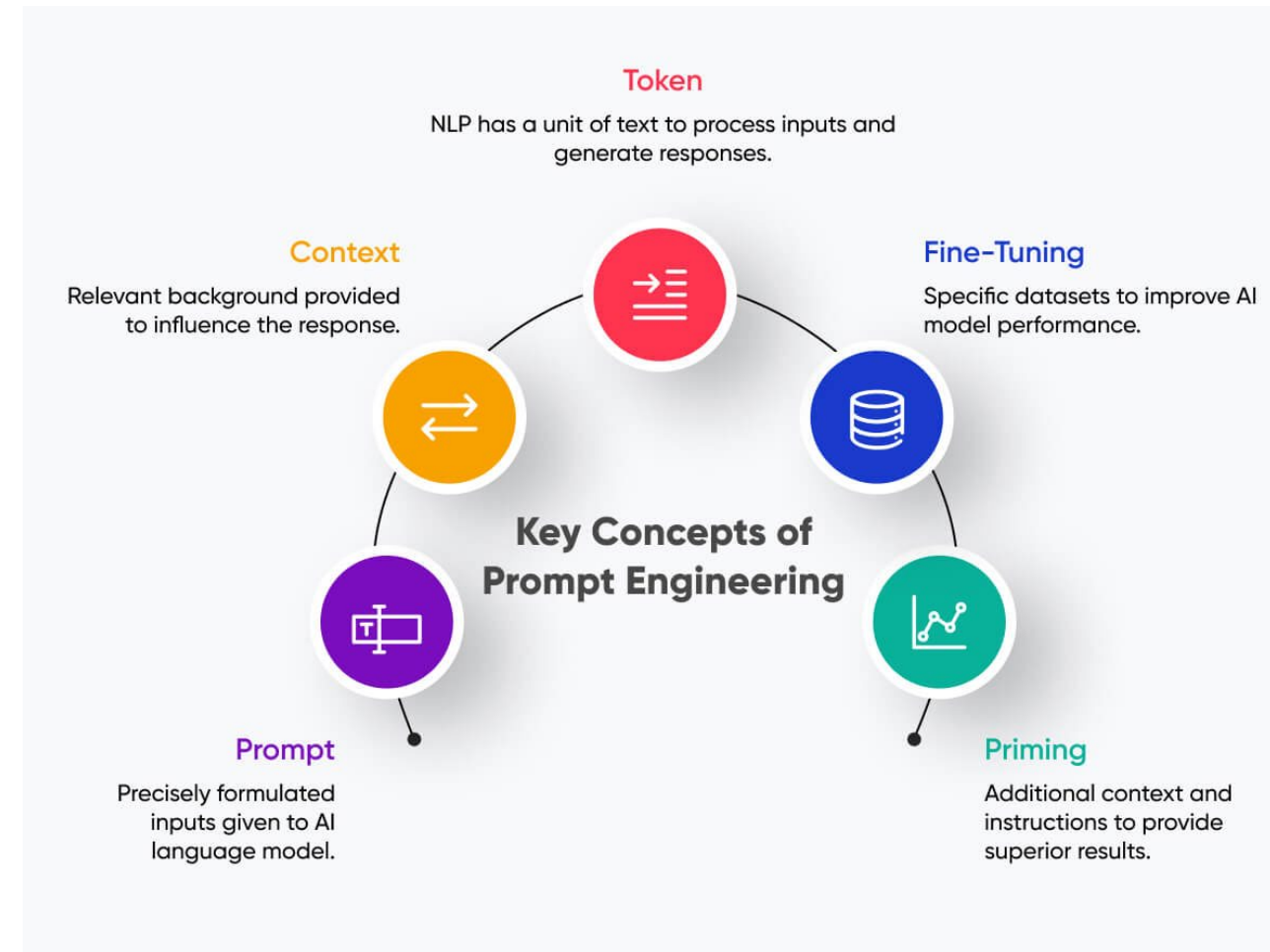
With access to an LLM, both SMEs (5+ years experience) and associates (<2 years experience) produced higher quality SE artifacts in less time.

LLM-generated artifacts serve as a basis for producing complete artifacts.

Sample Group	Time Taken	Included Diagrams	Qualitative Ranking	Quantitative Ranking
Associate Control	4h35	2	2	2 (77.84%)
Associate Accelerated	5h00	2	3	3 (71.37%)
Associate Open	5h30	2	1	1 (82.35%)
SME Control	2h25	5	3	3 (68.54%)
SME Accelerated	1h30	2	2	2 (70.59%)
SME Open	2h00	10	1	1 (80.88%)
ChatGPT 3.5	0h05	2	Last	Last (38.24%)

WHAT IS PROMPT ENGINEERING

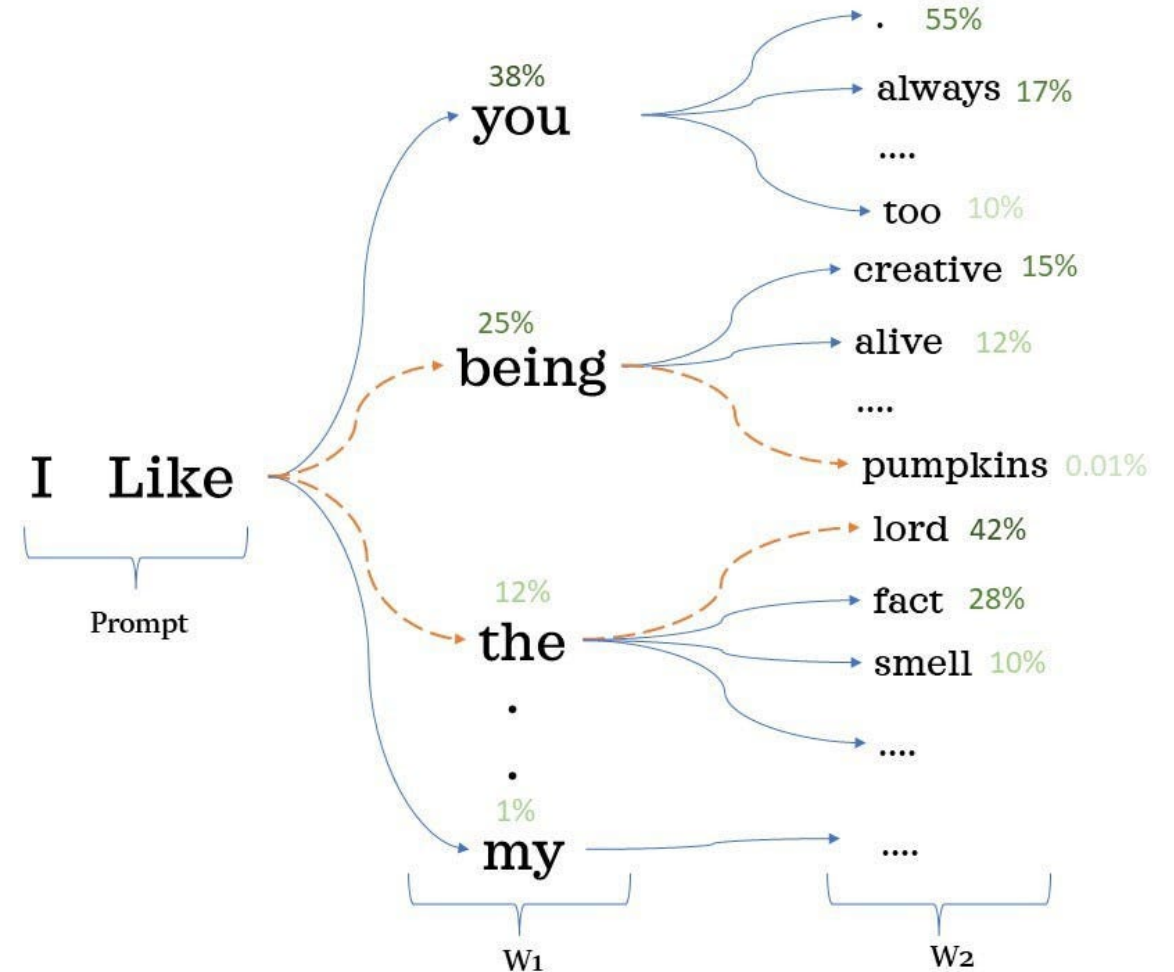
- Strategies for working with model inputs to get best outputs
- Configuration of prompt and model hyper-parameters
 - Model temperature
 - Prompting style
- Results vary on random seed
 - Looking for the best 'average' approach
- Results vary on selected model
 - Different models can react to changes differently



<https://www.weblineindia.com/blog/prompt-engineering-in-software-development/>

EFFECT OF MODEL TEMPERATURE

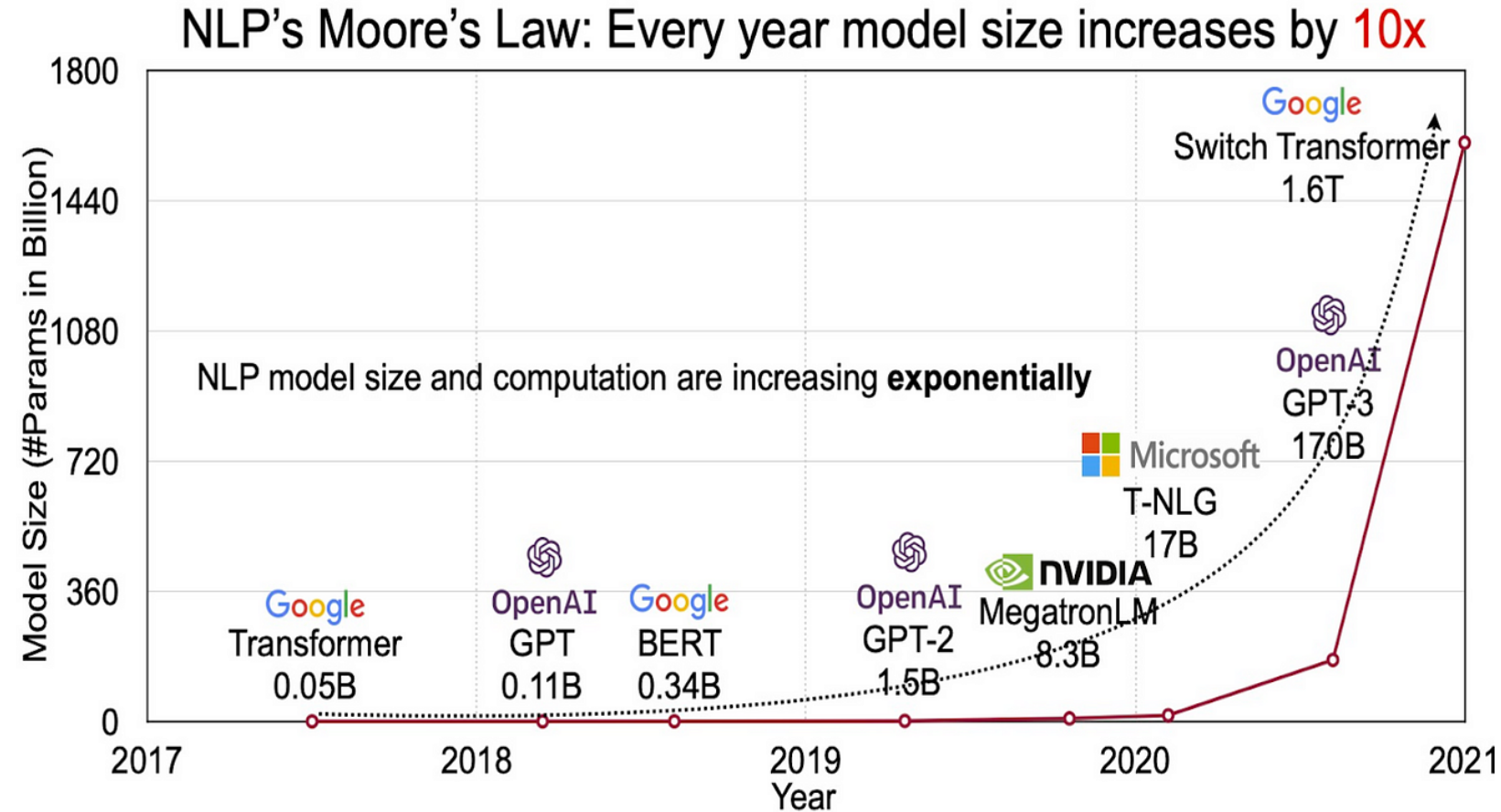
- Temperature is proportional to model creativity
 - Temperature = 0 means the most probable next word is always selected (no randomness)
 - Temperature > 0 scales up the chance of other next words being selected
- Since next word is dependent on previous word, changing one word can alter the entire response!



<https://masteringllm.medium.com/demystifying-the-temperature-parameter-a-visual-guide-to-understanding-its-role-in-large-language-d9e8ea4b9956>

MODEL SELECTION

- LLMs typically come in different sizes!
 - 5-10B parameter 'sweet spot' for computation time and available context
 - Models often available in 70B+ sizes as well
- Model architecture
 - Fast developing field, models from recent years are significantly more effective
 - Model architectures respond differently to parameter changes
 - Ex. Mixtral is less affected by temperature changes



<https://medium.com/@harishdatalab/unveiling-the-power-of-large-language-models-llms-e235c4eba8a9>

THE EXPERIMENT

- Used Mixtral 8x7B to generate 100 dot notation examples of each:

Items	Artifacts	Temperatures
Vacuum	Requirements lists	0.2
Air Purifier	Use case diagram	0.6
Toaster	State machine diagram	0.95

- Using each of four prompt techniques:
 - Zeroshot
 - Oneshot
 - Fewshot
 - Chain of thought
- These were then compared using a variety of NLP metrics to gold-standard artifacts developed by a SME systems engineer

WHY DOT NOTATION

- Models need context to produce effective outputs
 - Needs to exist in the model's training data
 - Needs to be a consistent and agreed upon format
- SysML v1 is not be consistent enough
- SysML v2 may be too new to be included in a given model's training data



Next Generation Systems Modeling Language

Formal release in Q1 of 2024

SAMPLE PROBLEM

- Prompt: "Generate a state machine for a toaster"
- Desired output is a commented dot diagram defining actions, states, transitions, initial and final states
- Model outputs a generic description and ascii state graph

```
State 1: Off
* Input: Power button is pressed
* Output: Toaster turns on, transitions to "Preheating" state

State 2: Preheating
* Input: Toast lever is inserted
* Output: Toaster begins toasting, transitions to "Toasting" state
* Input: Power button is pressed
* Output: Toaster turns off, transitions to "Off" state

State 3: Toasting
* Input: Toast lever is raised
* Output: Toaster stops toasting, transitions to "Cooling" state
* Input: Power button is pressed
* Output: Toaster turns off, transitions to "Off" state

State 4: Cooling
* Input: Power button is pressed
* Output: Toaster turns off, transitions to "Off" state

This state machine assumes that the toaster has a power button, a toast lever, and some way of detecting when it is in the process of toasting. The specific details of the inputs and outputs will depend on the design of the toaster.

Here is a diagram of the state machine:

Off
|
V
Power button pressed
|
V
Preheating
|
V
Toast lever inserted
|
V
Toasting
|
V
Toast lever raised
|
V
Cooling
|
V
Power button pressed
|
V
Off

I hope this helps! Let me know if you have any questions.
// Define the final state
end [shape=doublecircle];
EjectBread -> end;
}
```

Model Output

```
digraph ToasterStateMachine {
    // Define the actions
    ToastBread [shape=rectangle];
    PowerOn [shape=rectangle];
    PowerOff [shape=rectangle];
    EjectBread [shape=rectangle];
    AdjustTemperature [shape=rectangle];
    ActivateSafetyMechanism [shape=rectangle];
    DeactivateSafetyMechanism [shape=rectangle];
    CleanCrumbTray [shape=rectangle];

    // Define the states
    Ready [shape=circle];
    Toasting [shape=circle];
    PowerOffState [shape=circle];
    SafetyMode [shape=circle];
    CrumbTrayCleaning [shape=circle];

    // Define transitions
    Ready -> PowerOn [label="UserInput"];
    PowerOn -> Toasting [label="UserInput"];
    Toasting -> EjectBread [label="TimerElapsed"];
    Toasting -> AdjustTemperature [label="UserInput"];
    PowerOn -> PowerOffState [label="UserInput"];
    PowerOffState -> PowerOn [label="UserInput"];
    PowerOffState -> SafetyMode [label="ActivateSafetyMechanism"];
    SafetyMode -> PowerOffState [label="DeactivateSafetyMechanism"];
    SafetyMode -> CrumbTrayCleaning [label="CleanCrumbTray"];
    CrumbTrayCleaning -> SafetyMode [label="Done"];

    // Define initial state
    start [shape=point];
    start -> Ready;

    // Define the final state
    end [shape=doublecircle];
    EjectBread -> end;
}
```

Desired Output

SAMPLE PROBLEM - SPECIFY OUTPUT

- “Given your knowledge of Model-Based Systems Engineering, generate the dot notation representation of a state machine diagram for a toaster. Please include at least 5 states”
- Dot notation format. Includes 5 states and defines actions and transitions

```
digraph Toaster {
  rankdir=LR;
  node [shape=circle, style=filled, fillcolor=gray95];
  Unpowered [label="Unpowered"];
  Preheating [label="Preheating"];
  Toasting [label="Toasting"];
  Ready [label="Ready"];
  Error [label="Error", shape=doublecircle];

  Unpowered -> Preheating [label="Power on"];
  Preheating -> Toasting [label="Achieve toasting temperature"];
  Toasting -> Ready [label="Toasting complete"];
  Toasting -> Error [label="Overheat"];
  Ready -> Toasting [label="Toast button pressed"];
  Ready -> Unpowered [label="Power off"];
  Error -> Unpowered [label="Cool down"];
}
...
```

This diagram includes the following states:

- * Unpowered: The toaster is plugged in but turned off.
- * Preheating: The toaster is turning on and heating up to the toasting temperature.
- * Toasting: The toaster is toasting the bread.
- * Ready: The toaster has finished toasting and is ready for the next slice of bread.
- * Error: The toaster has overheated and needs to cool down before it can be used again.

The transitions between the states are labeled with the events that trigger the transitions. For example, the transition from the Unpowered state to the Preheating state is triggered by the event "Power on".

SAMPLE PROBLEM - ONESHOT

Request



```
Given your knowledge of Model-Based Systems Engineering, generate the dot notation representation of a state machine diagram for a toaster. Please include at least 5 states
```

Clearly labelled example



```
Example:
```

Example state machine in dot notation



```
digraph ElectronicPhotoFrameStateMachine {
    rankdir=LR;
    node [shape = ellipse];

    // States
    idle [label = "idle"];
    ChangingSettings [label = "ChangingSettings"];
    DisplayingImage [label = "DisplayingImage"];
    AdjustingInterval [label = "AdjustingInterval"];

    // Transitions
    idle -> ChangingSettings [label = "SettingChange"];
    idle -> DisplayingImage [label = "ImageNavigate"];
    idle -> AdjustingInterval [label = "AdjustInterval"];

    ChangingSettings -> idle [label = "ChangeSettings"];
    DisplayingImage -> idle [label = "after 24h"];
    AdjustingInterval -> idle [label = "AdjustSlideshowInterval"];
}
```

Prompt includes an example of desired output

SAMPLE PROBLEM - ONESHOT

- State diagram includes comments and same format as example
- Still includes extra text after the diagram!

```
digraph ToasterStateMachine {
  rankdir=LR;
  node [shape = ellipse];

  // States
  Unpowered [label = "Unpowered"];
  Idle [label = "Idle"];
  Heating [label = "Heating"];
  Toasting [label = "Toasting"];
  Done [label = "Done"];

  // Transitions
  Unpowered -> Idle [label = "Power On"];
  Idle -> Heating [label = "Start Toast"];
  Heating -> Toasting [label = "Heating Complete"];
  Toasting -> Done [label = "Toasting Complete"];
  Done -> Idle [label = "Toast Done"];
  Idle -> Unpowered [label = "Power Off"];
}
...
```

This state machine diagram includes the following states:

- * Unpowered: The toaster is unpowered and not ready for use.
- * Idle: The toaster is powered on and ready for use, but not currently toasting.
- * Heating: The toaster is heating up the bread.
- * Toasting: The toaster is actively toasting the bread.
- * Done: The toaster has completed toasting the bread.

The state machine includes transitions for powering the toaster on and off, starting a toasting cycle, and completing the toasting cycle.

SAMPLE PROBLEM - ONESHOT W/ SPECIFICS

- “Include only the dot notation diagram, with no introductory or concluding text, and do not include any irrelevant information from the example.”

```
digraph ToasterStateMachine {
  rankdir=LR;
  node [shape = ellipse];

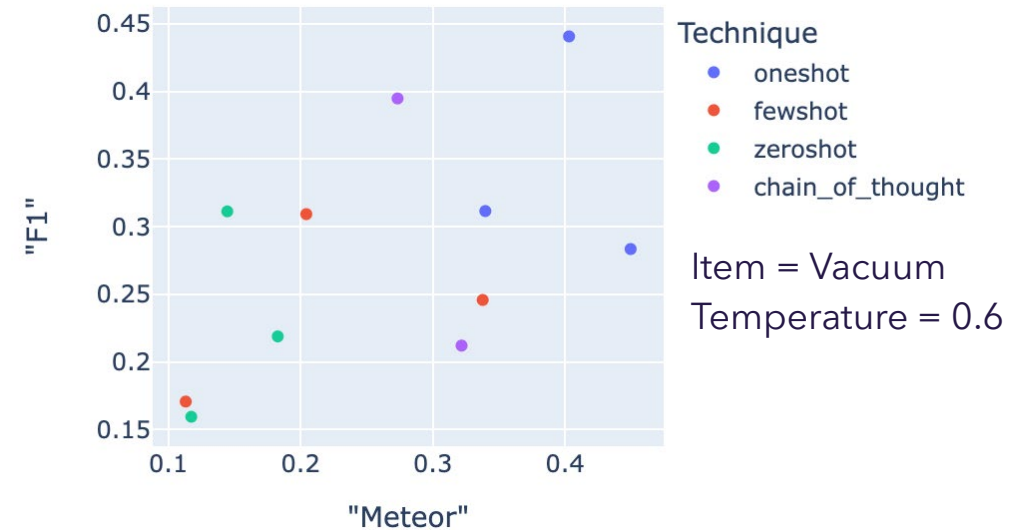
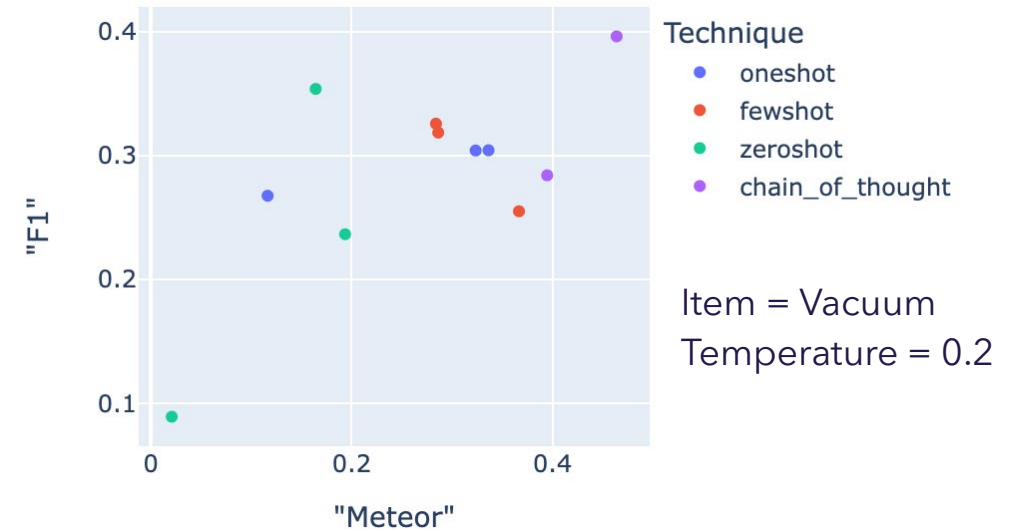
  // States
  Off [label = "Off"];
  Preheating [label = "Preheating"];
  Toasting [label = "Toasting"];
  Ready [label = "Ready"];

  // Transitions
  Off -> Preheating [label = "Power On"];
  Preheating -> Toasting [label = "Bread Inserted"];
  Toasting -> Ready [label = "Toasting Complete"];
  Ready -> Toasting [label = "Toast Button Pressed"];
  Ready -> Off [label = "Power Off"];
  Toasting -> Off [label = "Toasting Aborted"];
}
```

Clean and simple output. No additional text, dot diagram format.

STUDY RESULTS

- On average, **zeroshot prompts scored the lowest** on all metrics - all other techniques showed statistically significant improvements on it
- Increasing examples from **oneshot to fewshot** showed a **statistically significant change** in performance across most metrics - not always for the positive!
- **Chain of thought prompting performed exceptionally well generating state machine diagrams**, especially at lower temperatures

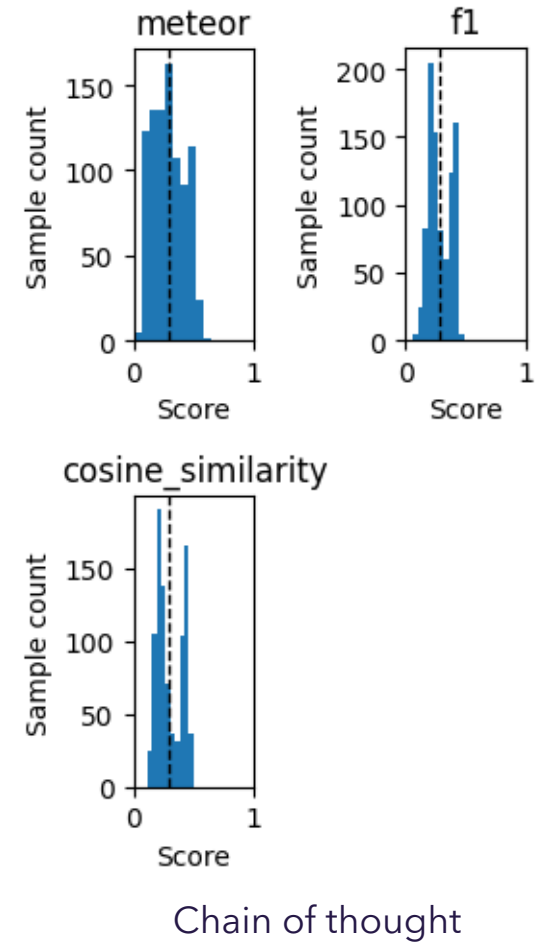
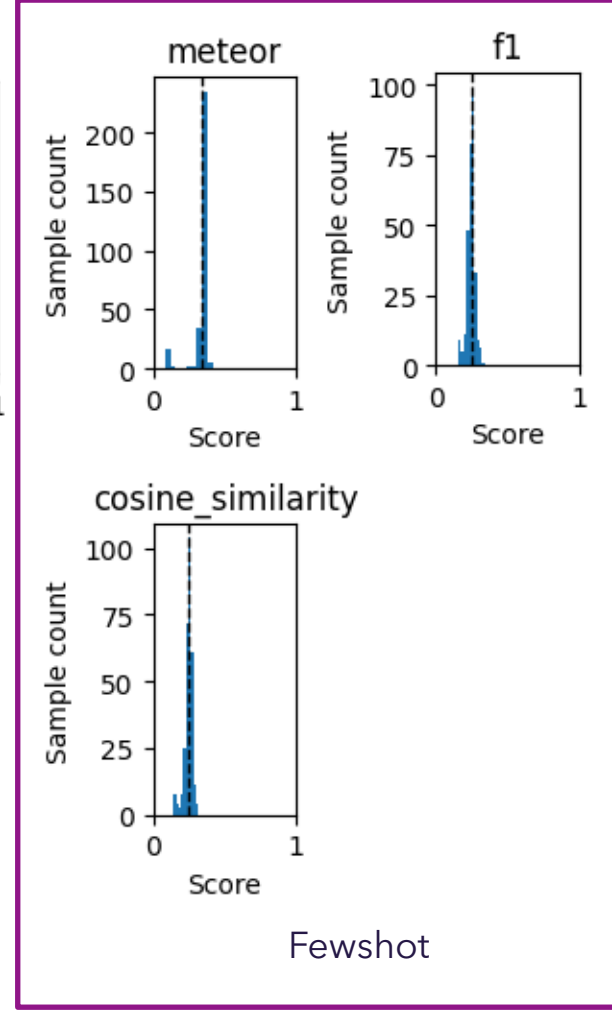
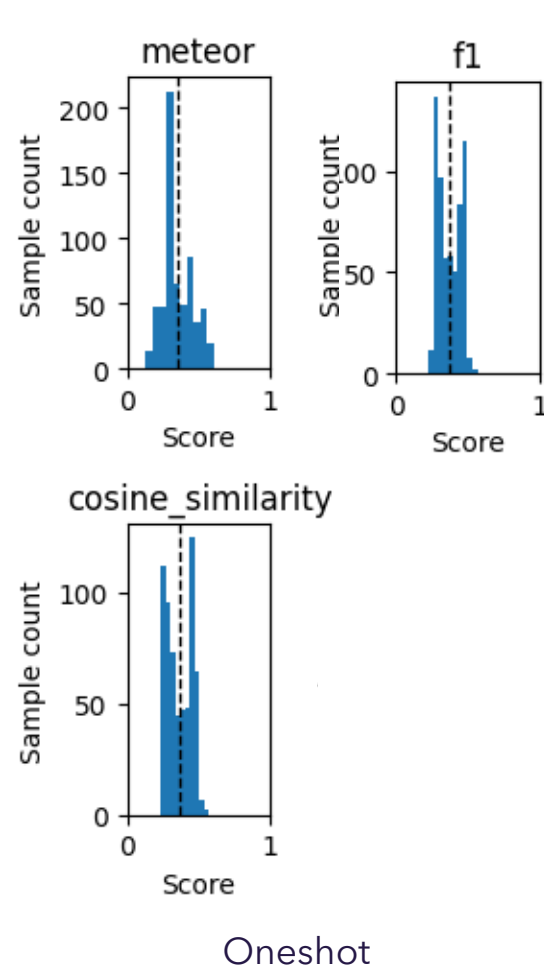
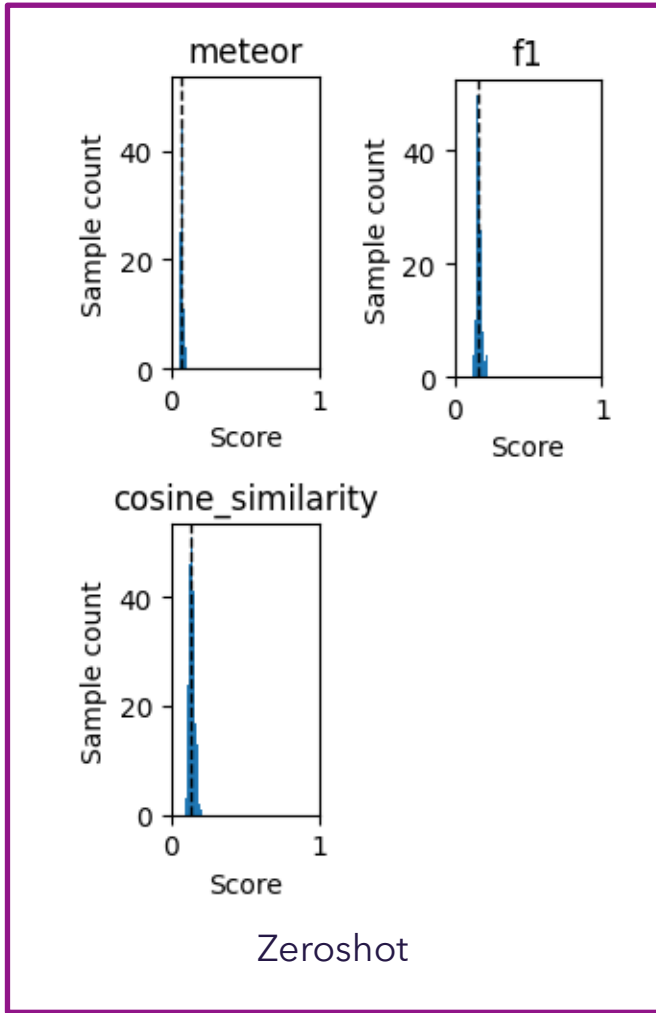


STUDY RESULTS

- **Lower temperatures showed greater consistency in responses**, as expected; but **some high temperature outlier responses were exceptionally good**
- The different items showed various levels of performance, with **air purifier-related artifacts scoring the lowest** across the board

Technique	Artifact	Temp	Kurtosis	F1	Meteor	Cosine
oneshot	use_case	0.2		0.369143918	0.293072	0.372074393
oneshot	requirements	0.2	-0.73158474	0.290857541	0.354798667	0.275947174
oneshot	state_machine	0.2		0.274185993	0.210992333	0.287923633
fewshot	use_case	0.2		0.391408067	0.325124333	0.397865385
fewshot	requirements	0.2	-0.850779239	0.230091893	0.327473667	0.226605655
fewshot	state_machine	0.2		0.288718201	0.219895333	0.318087461
zeroshot	use_case	0.2		0.119387414	0.061083667	0.121788099
zeroshot	requirements	0.2	0.868117281	0.170851392	0.137735333	0.147352473
zeroshot	state_machine	0.2		0.220176315	0.095013667	0.267983655
chain_of_thought	use_case	0.2		0.240296283	0.369808667	0.240860218
chain_of_thought	state_machine	0.2		0.320953817	0.319710667	0.346213628
oneshot	use_case	0.6		0.375903122	0.300788333	0.376191737
oneshot	requirements	0.6	-0.713658294	0.30459578	0.364439667	0.296312734
oneshot	state_machine	0.6		0.276786735	0.218113667	0.289051862
fewshot	use_case	0.6		0.37277812	0.301582333	0.379443911
fewshot	requirements	0.6	-0.812604439	0.229053622	0.323439333	0.225480878
fewshot	state_machine	0.6		0.285167738	0.234163	0.311726623
zeroshot	use_case	0.6		0.122259589	0.062566667	0.123794282
zeroshot	requirements	0.6	0.915166888	0.160097389	0.132023	0.137657489
zeroshot	state_machine	0.6		0.219719229	0.099518667	0.253344211
chain_of_thought	use_case	0.6		0.2482945	0.369354333	0.247636004
chain_of_thought	state_machine	0.6		0.325561821	0.301529667	0.351808283
oneshot	use_case	0.95		0.370284815	0.281657	0.370800922
oneshot	requirements	0.95	-0.701285373	0.305806407	0.365524333	0.29738403
oneshot	state_machine	0.95		0.268990594	0.214437	0.281501877
fewshot	use_case	0.95		0.299958613	0.274384333	0.308688883
fewshot	requirements	0.95	-0.708501774	0.227690861	0.310910333	0.223757929
fewshot	state_machine	0.95		0.282834105	0.231339	0.309062578
zeroshot	use_case	0.95		0.11807642	0.060495667	0.119399416
zeroshot	requirements	0.95	0.881815769	0.15971978	0.135619333	0.135684557
zeroshot	state_machine	0.95		0.219826855	0.10605	0.245366882
chain_of_thought	use_case	0.95		0.256782306	0.371891333	0.256996871
chain_of_thought	state_machine	0.95		0.321536171	0.305087667	0.346781374

STUDY RESULTS - TOASTERS



STUDY RESULTS - SUMMARY

- Zeroshot responses scored the lowest, but were not usually zero, indicating some level of useful material
- Oneshot responses greatly improved on zeroshot, and had a wide spread of low to high scores
- Fewshot responses tended to score consistently middle-to-high, but with much less variance than seen in oneshot responses
- Chain of thought responses had a wide variance, which seems to be influenced especially by temperature

LLM PROMPTING SUMMARY

- Start with a simple and short prompt and iterate from there
- Include an example, if you have one - it greatly improves model performance, especially with medium to large models
- Put the instructions at the beginning or at the very end of the prompt
- Clearly separate instructions from the text they apply
- Be specific and descriptive about the task and the desired outcome - its format, length, style, language, etc
- Avoid ambiguous descriptions and instructions
- Favor instructions that say "what to do" instead of those that say "what not to do"
- Use advanced techniques like Few-shot prompting and Chain-of-thought
- Version and track the performance of your prompts
- Keep prompt chains reasonably brief