**Prompts or Instruction:**

* **Definitely useful for zero or few shot for large enough LM**
* **We don’t know to what degree the following conditions are responsible for such improvement:**

1. **length(more computational units used for each question)**

**Makes sense especially when using a decoder only, because of the limited units of computation; the total units of computation is always limited by the length of the input and output.**

**Try to few shots with different length of null token, random generalized tokens, chosen prompts with different length. Then do the same thing on the encoder decoder model(first check if the model chosen is performing as well as the chosen LM), to see if the behaviors are the same.**

**Simplified text**

1. **distribution(related to the topic so narrows down the possible generations for the final answer) having useful word distribution in few shot might include the condition of having more more length or computation units used**

**Some (paper 5) tests the effect of prompt of a sequence of null tokens, which provides some more computational units, or the length of prompt, but without such “distribution”, the result is not as good as the regular, misleading, or irrelevant prompts.**

**Try input scrambled prompts to compare the performance with few shot with good prompts or no prompts**

**Try give a prompt word list like this:**

**Input: “A likes soccer, B doesn’t. Messi is a soccer player. Who has a higher chance of loving Messi, A or B?”**

**Prompt word list: [“sports”, “soccer”, “Messi”, “interest”]**

1. **Effects on label input mapping The main question here to ask is how to test such effects directly or without being affected by more length or helpful word distribution**

**Ask: How to observe these effects other than just looking at the result? Need a more direct way to observe these change, such as having a function that maps certain layer of parameters to certain functions)**

**Observe the increase of probability of generating the correct and incorrect target word when we give instructions. Then observe the increase of probability of generating the correct and incorrect target word when we give reversed instructions.**

**For example, for a binary classification task, in zero shot setting, without any instruction yet, if the prob of generating the correct target, like “yes”, after only knowing the input, is X, and the prob of generating the wrong target word, like “no”, is Y. Check what happens when the instruction “Is this sentence positive?” and “Is this sentence negative” are provided, how does the number X and Y change, and how do they change as the number of shots increase, which measures the influence of the true meaning of the prompt on such tasks.**

1. **Other**

* **Relationship with instruction tuning** 
  + **It’s believed that instruction tuning can make the model more robust to things like misleading or irrelevant prompts (acc stays the same when few shot with misleading, irrelevant or good prompts). But why and how? Such “robust behavior” is worth exploring.**
  + **If the instruction tuned model are less sensitive to the prompt, then it should perform bad on zero shot (very unlikely, since such models are designed to follow instruction)**
  + **What happens as the number of shots increases, how and when does the gap between such prompts close when using an instruction tuned model? (In paper 5 the gap between results using these three types of prompts decreases as the number of shots increases. So when zero shot, the performance of a good prompt is better than the other two.) We can repeat the experiment mentioned in Effects on label input mapping.**
  + **How does instruction tuning affect the result of the experiments above? Can repeat those experiments on instruction tuned models to see the difference with the result of the regular LM**
* **If effects on label input mapping are true, how to choose the best prompts?**
  + **Formal language or natural language**

**Explanation:**

* **E-P**
* **P-E**

**Try few shots with task 1 with reversed label, make sure the reversed result is high enough; then try it with another task (easy task is enough) see if the result is also reversed. Also try this with instructions to see if it will be more robust to such “reverse” change.**

**Chain of thought:**

* **Definitely useful for zero or few shot**
* **We don’t know how:**