Investigating Uncertainty Quantification in Large Language Models an Inplementation of : https://arxiv.org/abs/2305.19187

Dhairya Karna, Prashant Kumar.

1 Abstract

The realm of uncertainty quantification has gained substantial research attention within diverse machine learning domains, including the field of natural language processing (NLP). However, previous NLP studies have predominantly addressed uncertainty quantification challenges using the framework of classification or regression methodologies. For instance, a recent study approached the task of selective question-answering as a multiple-choice problem, treating it as a classification task. This approach, while allowing the application of well-established uncertainty quantification measures from the classification and regression contexts, overlooked the unique challenges associated with natural language generation (NLG).

In light of this, our current endeavor revolves around the implementation of a paper titled "Generating with Confidence: Uncertainty Quantification for Black-box Large Language Models." This paper dwells deeper into the topics of uncertainty quantification and explains the shortcoming of the models which treat uncertainty quantification in NLG as a classification task, by addressing the challenges of uncertainty in the context of NLG. The paper aims to bridge the gap between existing methodologies and the intricacies involved in text generation tasks.

2 Introduction:

Let us first attempt to comprehend some of the fundamentals of what an LLM is and what NLG is, which are critical in understanding the paper's implementation specifics.

LLM stands for Language Models, and it refers to large-scale language models in this context. Consider conversing with your smartphone or entering an inquiry into a search engine and obtaining a response that appears to be from a human. This is frequently powered by a large-scale language model. These models are trained on vast volumes of text data, allowing them to create human-like text depending on input. For example, if you ask your virtual assistant, "What's the weather like today?" an LLM examines the data and provides a correct response.

Natural Language Generation, or NLG is the process of creating human-like language using computers. Consider your weekly weather update email blog or update post on a website which contains generated account of a sports game or a political campaign or a Uchicago commencement speech. In many situations, these texts are generated using NLG systems. They evaluate data, such as temperature trends or gaming statistics, and then transform that data into a logical, human-readable story. An NLG system, for example, may use the score, player data, and important moments from a cricket game to generate an interesting recap that you can read the next morning, without the involvement of a human writer.

Now that we know what or LLMs are, we can approach what we are aiming to do in our work in greater depth. Imagine a fascinating situation where we could connect the dots between two exciting areas: understanding how to trust models (like those predicting things or helping generate language) and figuring out how uncertain or confident these models are. Traditionally, experts focused on how much these models trust their own answers based on the input they get and what they predict. But we're here to bridge the gap between these fields. We're diving into the details of how these models rank different examples and giving them a way to show how unsure or sure they are about their responses. This is like giving these models a confidence check to see how well they know their stuff when they create sentences.

Our aim is to look deeper into the nuances of the paper called ""Generating with Confidence: Uncertainty Quantification for Black-box Large Language Models." This paper dives into the challenge of understanding how reliable the responses from these models are, especially when we don't have much information about how certain they are. Think of it as trying to see if you can trust the answers your computer gives you. As the way these models work becomes a bit of a mystery due to technology changes, the paper is digging into this puzzle for models that are a bit secretive or a black box. We're looking at how much uncertainty comes from what they're given and how much comes from what they create, and coming up with different ways to check how confident they are. These ways come in handy when we want to choose the best sentences they create and double-check the ones that might not be so sure. The experiments we conducted based on the aforemention research work have shown a simple yet smart trick—the average spread of meaning—can tell us a lot about how good their sentences are. So, we' implemented these ideas from the paper with our own logic to better understand and use these models.

3 Related Work

In order to comprehend the research paper "Generating with Confidence: Uncertainty Quantification for Black-box Large Language Models," we studied the LLM domain, looking at how various scholars have attempted to quantify and manage uncertainty in the outputs of these models. This summary encapsulates numerous works that were studied to varying degrees in the process of comprehending this paper.

People have investigated Large Language Models (LLMs) for their ability to grasp their own knowledge boundaries and quantify uncertainty, which is critical for preventing incorrect information.[17] focuses on "known-unknown" issues with considerable ambiguity due to a lack of unambiguous solutions. To investigate this topic, they compiled a dataset of Known-Unknown Questions (KUQ) and devised a classification approach to identify the sources of ambiguity. In an Open-Ended QA setting, they then examined the LLMs' capacity to discriminate between known and unknown questions and categorize them, as well as evaluate the quality of their replies. To measure the models' accuracy in conveying uncertainty between known and unknown issues, a unique semantic evaluation approach was created.

Current researchers are investigating several methods for determining how certain or uncertain a language generation model (LGM) is about its responses. A number of researchers are fine-tuning models to better understand the uncertainty, while others are investigating how distinct cues are connected to one another and determining the uncertainty from there[4,5,6]. [5] found that a GPT-3 model could tell us how confident it is in its own responses in simple, ordinary language, without the necessity for sophisticated mathematical computations (logits). For example, it may state that it is "90% confident" or has "high confidence" in an answer. These confidence levels turned out to be very close to the real probability. What's more, this confidence stays steady even with changes in the questions. This was a first, and it's a big deal! Researchers used new tests and comparisons to prove this, and they found that GPT-3's consistency in confidence seems rooted in deep underlying patterns. It's another step towards understanding how these models think, and getting them to communicate like humans.

However the work in our paper [8] on uncertainty quantification is derived from [7] which introduces the concept of "semantic entropy" to quantify uncertainty in generated answers by considering equivalence relationships. This approach does not involve training, but it does rely on access to token-level numerical output from the Language Model (LLM), which may not always be accessible or available in certain contexts. Therefore we have gone ahead and aim to use NLI(Natural Language Inference) classifier to evaluate the similarity of responces as in [8]. We would be explaining more on these methods in upcoming sections.

Scavenging more articles and reading some more literature we found that Language Models (LMs) have impressive capabilities in solving new tasks with limited examples or textual instructions, despite struggling with basic functionalities such as arithmetic or factual lookup, where smaller models outperform them. In [16], the authors introduce Toolformer, a self-supervised model trained to leverage external tools through simple APIs. Toolformer learns to decide which APIs to use, when to call them, what arguments to provide, and how to incorporate the results for future token prediction. This approach requires only a small number of demonstrations for each API. The paper incorporates various tools like a calculator, Q&A system, search engine, translation system, and calendar. Toolformer significantly enhances zero-shot performance across various downstream tasks, often matching or surpassing larger models, while maintaining its core language modeling capabilities.

Finally although we failed to employ prompt engineering, some of our preliminary literature study helped us comprehend the concept of Prompt Engineering. Prompt engineering is a common method for interacting with Large Language Models (LLMs) and has been widely used across various NLP tasks. This approach requires careful design, as LLMs interpret prompts differently than humans. Traditional methods using gradient-based optimization become impractical with larger models, and recent work has focused on optimizing instructions within the natural language space itself. Unlike earlier techniques relying on specialized models, this new approach shows that the entire process can be conducted using a single LLM.

4 **Preliminaries Uncertainty Vs Confidence:**

These are some concepts introduced by the paper “Generating with Confidence: Uncertainty Quantification for Black-box Large Language Models." and we try to explain and elaborate them for better understanding.

4.1 Uncertainty

refers to the dispersion or variability present in the posterior distribution of a predicted outcome Y conditioned on the input X=x. Specifically p(Y|X=x).The randomness or entropy in this prediction is termed as predictive entropy, which is formally defined as:

$$-\int p(y|x)\log(p(y|x))dy$$

Drawing from this, the authors define Uncertainty for a given input x and a random sequence of tokens s as:

$$U(x) = H(s|x) = -\sum_s p(s|x) * \log(p(s|x))$$

4.2 Confidence

On the other hand, **Confidence** is often misconstrued as the opposite of

Uncertainty. However, they serve distinct purposes. Confidence scores are a measure of the predicted probability of an outcome, considering both the input and the output. It can be simplified as

$$p(Y=y|X=x)$$

The joint probability, representing the confidence score, is given by:

$$C(x, s) = \widehat{p}(s|x) = \prod_i \widehat{p}(s_i|s_{<i}, x)$$

However, a challenge arises in computing confidence, as it necessitates access to embeddings and the dimensionality of the output space, which is often restricted in black-box models.

To elucidate the distinction between Uncertainty and Confidence, consider the question posed to a Language Model (LLM): “Where was the 2020 Olympics held?” The varied responses, ranging from "Tokyo, Japan" to "Paris, France", highlight the model's uncertainty. However, confidence estimation requires knowledge of the correct answer and then determining which set of responses aligns with that correct answer.

4.3 Entailment and NLI

Imagine two pieces of text having a conversation. One, the "premise," makes a statement, and the other, the "hypothesis," tries to deduce something from it. This is the essence of Natural Language Inference (NLI). It's like the brainwork behind our chatbots and dialogue systems, helping them grasp what we're hinting at and respond thoughtfully. Think of NLI as categorizing these deductions into three 'moods': Positive entailment, where the hypothesis nods in agreement, saying, "Yes, that proves your point!"; Negative entailment, where it shakes its head, countering, "No, that contradicts what you said!"; and Neutral entailment, where it shrugs, suggesting there's no clear connection between the two statements.

4.4 Rouge Scores

Imagine you're trying to tell a friend about the plot of a movie you just watched, but you only have a few seconds. You quickly summarize it, and then your friend, who's seen the movie, rates how close your quick recap was to the actual story. That's what ROUGE scores do for machine-generated summaries. Think of them as a "match score" between the computer's summary and the original text. They look at how many words or phrases (n-grams) match up. The scores dance between 0 and 1, with a score closer to 1 meaning the machine did a top-notch job. For instance, if the original sentence was "The cat is sleeping on the mat," and the machine said, "The cat is on the carpet," the ROUGE score would be 0.76. Not perfect, but pretty close!

5 Infrastructure

Deciding on the right infrastructure is pivotal for NLP projects; a more robust setup allows training of larger models, potentially boosting accuracy. However, this enhancement often comes with increased costs and resource demands.

The optimal infrastructure for NLP projects involves a trade-off between cost, performance, and scalability. **Local setups** provide direct control and data security, but they may lack the processing capability needed for bigger models. **Colab** is a free, user-friendly environment with GPU support; nevertheless, its session-based structure might be restricting for long-term activities. **AWS services**, on the other hand, provide huge scalability and a range of integrated tools, but they may be more expensive and have a steeper learning curve. Each option has advantages and disadvantages, and the selection should be made in light of the project's scope, budget, and technical needs.
Diving deeper into some of these choices as follows:

5.1 Local Infrastructure:

Pros:

- **Control and Customisation**: Local setups allows us more control over our project and customisations can be done without much external cost.
- **Data Security**: Local setups also gives us more control over the data we produce.
- **Ease of use**: There is no initial learning barrier

Cons:

- **Lack of resources**: Local setups will not be scalable as the requirements of the NLP project grow.
- **Lack of advanced tools and plugins:** There are various utility tools that comes with various cloud solutions which may not be compatible with the local infrastructure.

5.2 Google Colab

Pros:

- **Gpu with minimal initial cost**:Colab offers access to GPUs at a minimal initial cost of 10$.
- **No setup required**:There is no initial setup required and quick to begin with as no external learning is required.
- **Integration with Google drive**: Handling data becomes easier because it is integrated with Google drive enabling storing both results and training data with ease.

Cons:

- **Session based limitations**:Freequent session timeout even in the paid version limits the max training time of any model, and larger models cannot be trained.
- **Cap on the max compute on high end GPUs:** While GPUs are available, some of the high end GPUs might not be sufficient for extremely large models or datasets over multiple iterations.
- **Data Transfer**: Large datasets need to be uploaded frequently if not stored on Google Drive, which can be time-consuming.

5.3 AWS

Pros:

- **Scalability**:AWS provides a diverse set of instance types appropriate for a variety of NLP tasks, ranging from small-scale processing to massive model training.
- **Integrated Toolset:** With services like SageMaker, AWS provides tools specifically designed for ML and NLP tasks**.**

- **Integration with Google drive**: Handling data becomes easier because it is integrated with Google drive enabling storing both results and training data with ease.

Cons:.

- **Learning Curve:**For newbies, AWS's wide collection of services might be overwhenlming.

After careful consideration, we decided on a hybrid approach to our NLP project's infrastructure. For computationally intensive tasks, such as training our models, we leveraged Google Colab, capitalizing on its relatively cheap GPU access and robust processing capabilities. However, for lighter tasks like generating sequences, we opted for our local setup. This division was primarily influenced by Colab's limitation of 100 hours of GPU compute per month. By using Colab for the heavy lifting and our local environment for more routine operations, we ensured optimal resource allocation without compromising on efficiency or incurring unnecessary costs.

6 Dataset

In our endeavor to quantify uncertainty and confidence in Natural Language Processing tasks, we leveraged two prominent datasets: TriviaQA and CoQA.

6.1 TriviaQA

TriviaQA is a reading comprehension dataset containing over 650K question-answer pairs obtained from trivia and quiz-league websites. The dataset is unique in its design, as each question is associated with multiple evidence documents, which can be a combination of web pages and structured texts. This multi-evidence structure makes it especially suitable for our task, as it allows for a nuanced understanding of model confidence across different sources.

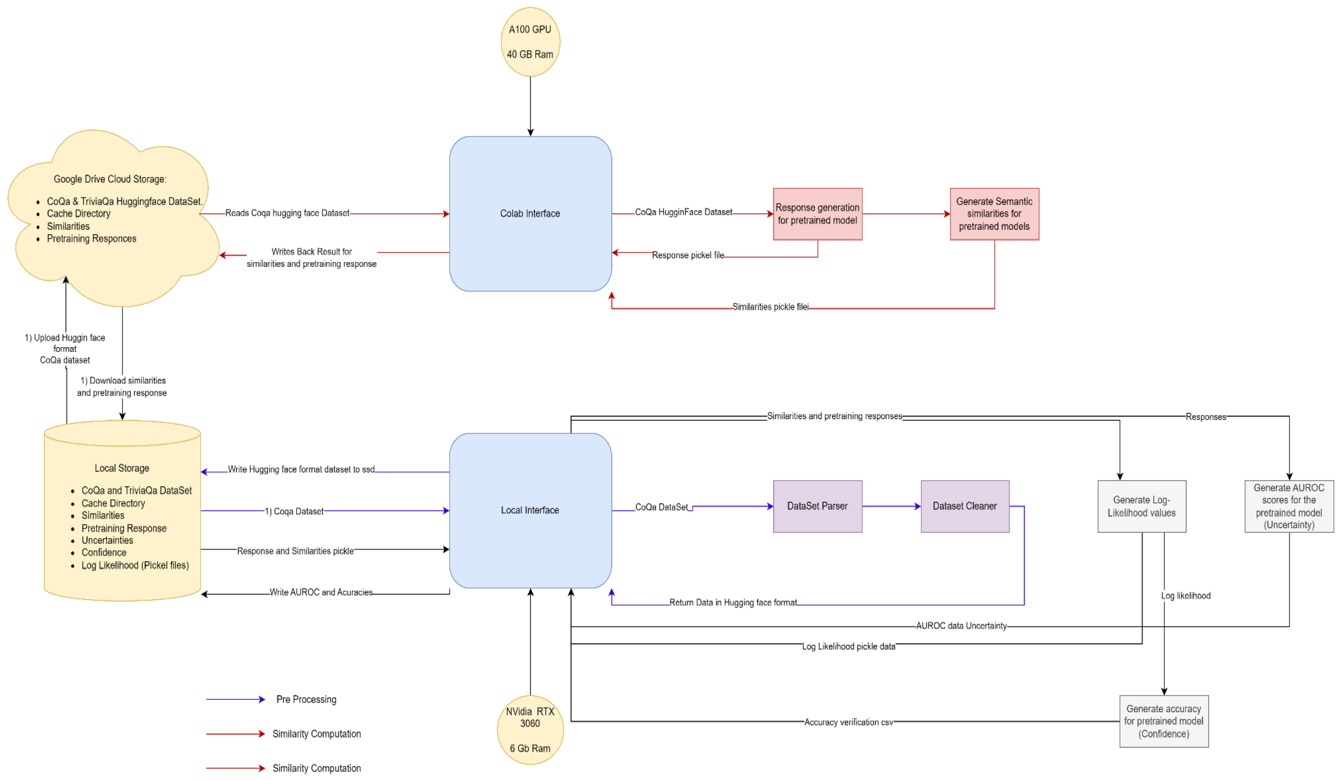
Example:
Question: Who won the Nobel Peace Prize in 2009?
Answer: Barack Obama
Evidence: Web page snippet from a news article detailing the award ceremony.

6.2 CoQA

The Conversational Question Answering (CoQA) dataset is designed for the task of conversational question answering over a range of domains. It contains 127K questions with answers, where each conversation consists of a series of interconnected questions and answers. The conversational nature of CoQA provides a dynamic context, making it a valuable resource for studying uncertainty, as the context can significantly influence the model's confidence.

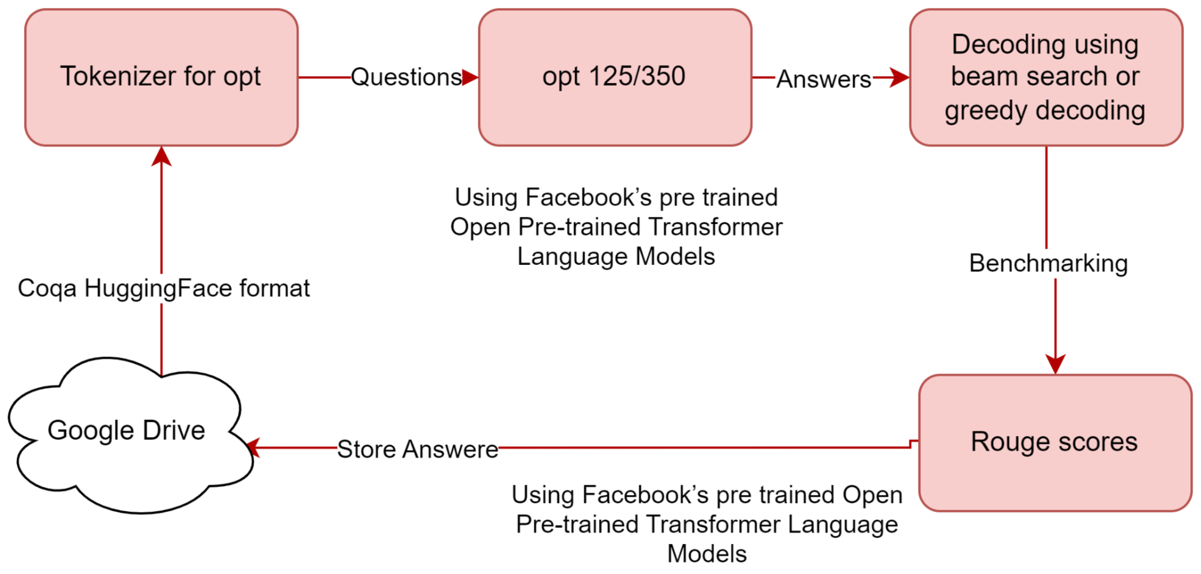
Example:
Context: Anne is a software developer who recently moved to Seattle.
Question: Where did Anne move recently?
Answer: Seattle
Follow-up Question: What does she do for a living?
Answer: She is a software developer.

7 Modules


A100 GPU
40 GB Ram
Google Drive Cloud Storage:
CoQa & TriviaQa Huggingface DataSet.
Cache Directory
Similarities
Pretraining Responces
Reads Coqa hugging face Dataset
Writes Back Result for similarities and pretraining response
Colab Interface
CoQa HugginFace Dataset
Response generation for pretrained model
Generate Semantic similarities for pretrained models
Response pickel file
Similarities pickle filei
1) Upload Huggin face format CoQa dataset
1) Download similarities and pretraining response
Local Storage
CoQa and TriviaQa DataSet
Cache Directory
Similarities
Pretraining Response
Uncertainties
Confidence
Log Likelihood (Pickel files)
Write Hugging face format dataset to ssd
1) Coqa Dataset
Response and Similarities pickle
Write AUROC and Acuracies
Local Interface
CoQa DataSet
DataSet Parser
Dataset Cleaner
Return Data in Hugging face format
Similarities and pretraining responses
Responses
Generate Log-Likelihood values
Generate AUROC scores for the pretrained model (Uncertainty)
Log likelihood
AUROC data Uncertainty
Log Likelihood pickle data
Generate accuracy for pretrained model (Confidence)
Accuracy verification csv
NVidia RTX 3060
6 Gb Ram
Pre Processing
Similarity Computation
Similarity Computation

7.1 Pre-Processing and Generate Responses

This module is used for auto-generating and assessing responses using a pre-trained language model. The module processes datasets, either 'coqa' or 'trivia_qa', into a tokenized format suited for the model by leveraging Facebook's OPT (Open Pre-trained Transformer Language Models) from the Transformers library. Following that, responses to dataset prompts are created using different decoding methods such as beam search or greedy decoding and benchmarked against reference answers using metrics like as ROUGE and Exact Match. The results are cataloged for further study, including the produced replies and their related assessments.



Tokenizer for opt
Questions
opt 125/350
Answers
Decoding using beam search or greedy decoding
Using Facebook's pre trained Open Pre-trained Transformer Language Models
Coqa HuggingFace format
Benchmarking
Google Drive
Store Answere
Rouge scores
Using Facebook's pre trained Open Pre-trained Transformer Language Models

7.2 Generating Similarities

7.2.1 Theory

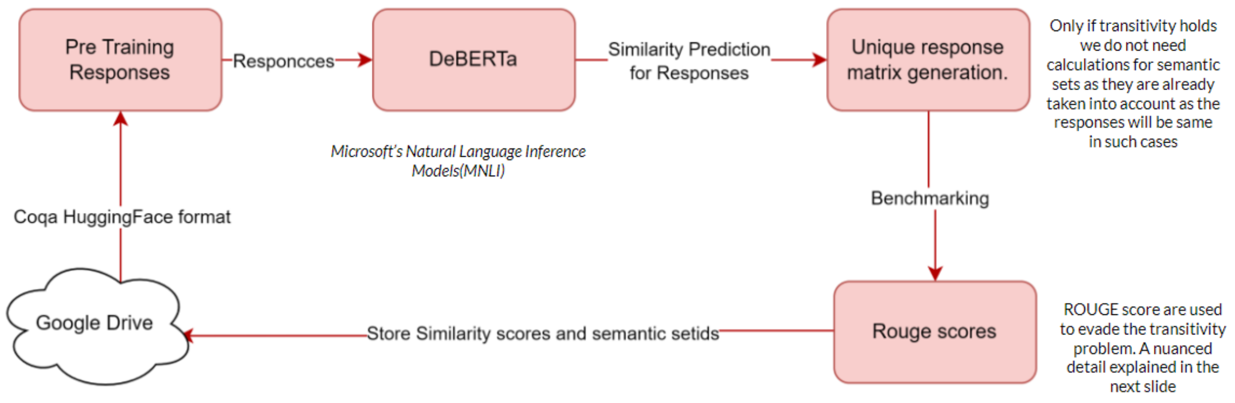
Measuring similarities is not straightforward. Jaccard(intersection/union) similarity is one of the most simplest metric employed to measure the similarity between response tokens from a LLM. Even though it is one of the most widely employed method to compute similarities in NLP it still does not take into account the sequence of words and crucial expressions likes negations.

An alternative to the rule based approach like jaccard similarity is to use NLI models which basically predicts probability scores between two segments. These predicted probabilities are for three classes, entailment, neutral and contradiction and these predicted probabilities can be used as similarity scores.

$a_{\text{NLI,entail}}(s_{j1}, s_{j2}) = p_{\text{entail}}(s_{j1}, s_{j2})$ $a_{\text{NLI, contra}}(s_{j1}, s_{j2}) = 1 - p_{\text{contra}}(s_{j1}, s_{j2})$

7.2.2 Implementation

The purpose of this module is to evaluate text replies produced by language models to evaluate their semantic and syntactic similarity. It evaluates the distinctness of produced texts using a pre-trained DeBERTa(Decoder Enhanced BERT with Disentangled Attention) Large Microsoft's Natural language inference(MNLI) model and measures syntactic similarity with the ROUGE metric. For convenience, the outcomes of pairwise semantic similarity predictions are stored into a CSV file.


Pre Training Responses
Responcces
DeBERTa
Similarity Prediction for Responses
Unique response matrix generation.
Only if transitivity holds we do not need calculations for semantic sets as they are already taken into account as the responses will be same in such cases
Microsoft's Natural Language Inference Models(MNLI)
Coqa HuggingFace format
Benchmarking
Google Drive
Store Similarity scores and semantic setids
Rouge scores
ROUGE score are used to evade the transitivity problem. A nuanced detail explained in the next slide

The need for significant GPU RAM while running this module is a major factor. Particularly, at least 40 GB of GPU RAM is required for smaller language models like Facebook's opt-125m or opt-350m. However, a minimum of 80 GB of GPU RAM is required for bigger versions like the opt-30b. This module's implementation was accomplished using Google Colab Pro. For larger RAM requirements we recommend using the Google Colab Pro Plus Subscription or using a high performance GPU cluster

The setup includes various configurations such as setting seeds for reproducibility and leveraging the wandb library for experiment, weights and biases tracking. Tokenizers from the Transformers library are employed for processing, and semantic similarity evaluations are facilitated using the DeBERTa model. The outcomes of the module, encompassing syntactic similarities and semantic set IDs, are stored for further analysis and utilization.

7.3 Semantic Entropies, defining number of Semantic sets as uncertainty:

7.3.1 Theory

Uncertainty can be quantified using the number of semantic sets derived from similarities in the NLI model and the LLM's numerical outputs, termed as semantic entropy. Spectral clustering is employed by fixing the input x and treating each output as a node. While direct implementation is challenging in black-box LLMs, we can use the count of these semantic sets as an uncertainty measure, obtained via bi-directional entailment algorithms, denoted as U_{NumSet}. It's pivotal to note the underlying assumption: semantic similarities are considered transitive.

7.3.2 Implementation:

7.3.2.1 compute_neg_log_likelihoods

This module is specifically designed to compute negative log-likelihoods and embeddings for the primary and secondary likely generations within a sequence set. Utilizing a pre-trained model, specifically Facebook's opt-125m and opt-250m, the metrics are evaluated considering both prompt and target IDs.

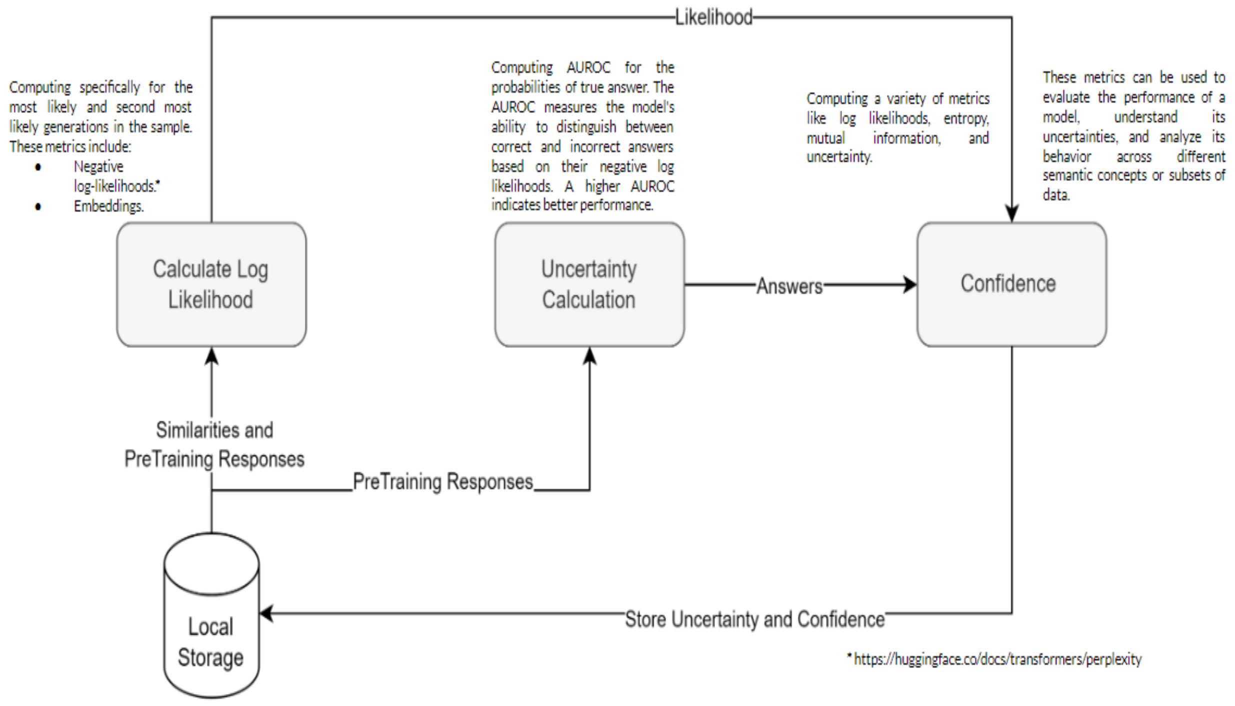
The main function, **compute_neg_log_likelihoods**, calculates various metrics, with particular emphasis on the most and second most likely generations. These metrics encompass average negative log-likelihoods, unconditioned negative log-likelihoods, pointwise mutual information, and unique sequence embeddings for both the most and second most likely generations. Imbuilt functions are used here and more on this can be found on the github link provided at the end of the article. For reproducibility, a seed value is set ensuring consistent results across multiple runs. After computation, results are stored for subsequent analysis, and GPU memory management is emphasized to optimize computational efficiency.

7.3.2.2 generate_uncertainty

This module examines a selection of responses comprising questions and corresponding answers, with the goal to gauge the capability of a model in determining the accuracy of generated answers. Utilizing Facebook's pre-trained opt-125m and opt-125m model, each response is tokenized and its negative log likelihood is computed. To enhance the evaluation, the module crafts a few-shot prompt by iterating through a subset of sequences, integrating the most likely generated answers. Furthermore, the Area Under the Receiver Operating Characteristic (AUROC) is calculated for the probabilities of correct answers, offering a quantitative insight into the model's performance. The computed AUROC values are then efficiently stored for subsequent analysis. To ensure consistent results across varying runs, a fixed seed value is incorporated.

7.3.2.3 generate_confidence

The generate_confidence module uses the opt-125m and opt-350m model to measure the system's confidence in its responses. By calculating metrics like mutual_information and predictive_entropy, we gauge the system's confidence and understand its uncertainty in predictions. To ensure consistent results, a specific seed value is set. The findings are then stored for future analysis, providing a snapshot of the system's trustworthiness.


Likelihood
Computing specifically for the most likely and second most likely generations in the sample. These metrics include:
• Negative log-likelihoods.*
• Embeddings.
Computing AUROC for the probabilities of true answer. The AUROC measures the model's ability to distinguish between correct and incorrect answers based on their negative log likelihoods. A higher AUROC indicates better performance.
Computing a variety of metrics like log likelihoods, entropy, mutual information, and uncertainty.
These metrics can be used to evaluate the performance of a model, understand its uncertainties, and analyze its behavior across different semantic concepts or subsets of data.
Calculate Log Likelihood
Uncertainty Calculation
Answers
Confidence
Similarities and PreTraining Responses
PreTraining Responses
Local Storage
Store Uncertainty and Confidence
* https://huggingface.co/docs/transformers/perplexity

8 Graph Laplacian and its workaround

The issue of transitive nature while calculating similarities in reponses is the reason for differentiating between Uncertainity and Confidence.

- If a response A is similar to response B (eg. thier rouge_score (similarity metric) is high) and B is similar to response C, then we can not surely say that response A is similar to C

When can measure uncertainty of an LLM by calculating the probability of a sample response with the target's true answer which can also be better visualized using the AUROC scores for the correct answers for a given prompt.

However, since these similarities are not transitive in nature we can not surely say that the AUROC scores will be same for the generations of these responses because if we change the order of the responses then the answer changes.

The paper was using the concept of normalized Graph Laplcian to form the idea of the semantic embedding of a LLM. If we get the semantic embeddings for the LLM then we can be sure of the probability of most likely generation given the true response for the prompt. For this first we calculate the similarities for a pair of unique responses and its reverse.

```
# Create input pairs and encode them
input_pair = question_text + ' ' + unique_responses[i] + ' [SEP] ' +
unique_responses[j]
encoded_input = tokenizer.encode(input_pair, padding=True)
prediction = model(torch.tensor([encoded_input], device='cuda'))['logits']
predicted_label = torch.argmax(prediction, dim=1)

# Reverse the input pair and encode
reverse_input_pair = question_text + ' ' + unique_responses[j] + ' [SEP] ' +
unique_responses[i]
encoded_reverse_input = tokenizer.encode(reverse_input_pair, padding=True)
reverse_prediction = model(torch.tensor([encoded_reverse_input],
device='cuda'))['logits']
reverse_predicted_label = torch.argmax(reverse_prediction, dim=1)
```

According to the concept of Graph Laplacian we should form a weight matrix such that $W \rightarrow w_{j1,j2} = (a_{j1,j2} + a_{j2,j1})/2$. Then we use the matrix W to form a normalized graph laplacian L such that $L = I - D^{(-1/2)} W D^{(-1/2)}$
where, $D \rightarrow d_{j1,j2} = \Sigma_{j' \in [m]} w_{j1,j'}$ if $(j_1 = j_2)$ and 0 otherwise

This matrix D can help us generate the semantic embedding U_{NumSet} by using the formula: $U_{NumSet} = \Sigma^{m}_{k=1} \max(0, 1 - \lambda_k)$. Where λ_i are the eigenvalues for L.

The benefit of using this concept is that it plays a crucial role in embedding nodes of a graph into a lower dimensional space while preserving the local structure of the nodes. This can help us better understand the embeddings.

However, when we tried to implement this concept we were facing many computational, code execution and theortical implementation blocks. To overcome this gap we realized that we essentially need to use the semantic embeddings as a way to cross check if our predicted accuracy for the model is the same regardless of the order of the responses. So we can circumvent the need of implementing graph Laplacian to get U_{NumSet} and rather store all the different semantic sets generated and later on cross check the predictions with our responses by manually saving the most likely generation in U_{NumSet} and the true response with the accuracy predicted by our model.

```
# 6. Number of Semantic Sets
number_of_semantic_sets_auroc = sklearn.metrics.roc_auc_score(1 -
comprehensive_dataframe['correct'],

comprehensive_dataframe['number_of_semantic_sets'])
analysis_results['number_of_semantic_sets_auroc'] =
number_of_semantic_sets_auroc

# Compute average number of semantic sets for correct and incorrect predictions
analysis_results['number_of_semantic_sets_correct'] =
comprehensive_dataframe[comprehensive_dataframe['correct'] ==
1]['number_of_semantic_sets'].mean()
analysis_results['number_of_semantic_sets_incorrect'] =
comprehensive_dataframe[comprehensive_dataframe['correct'] ==
0]['number_of_semantic_sets'].mean()
```

Although this takes up more memory and it is not computationally efficient, this will help us get the Confidence score/ Accuracy of the model we require while circumventing the need to use regularization concept of Graph Laplacian. The results of our accuracy and the screenshot of the csv file for our embedding and prediction are in the results section.

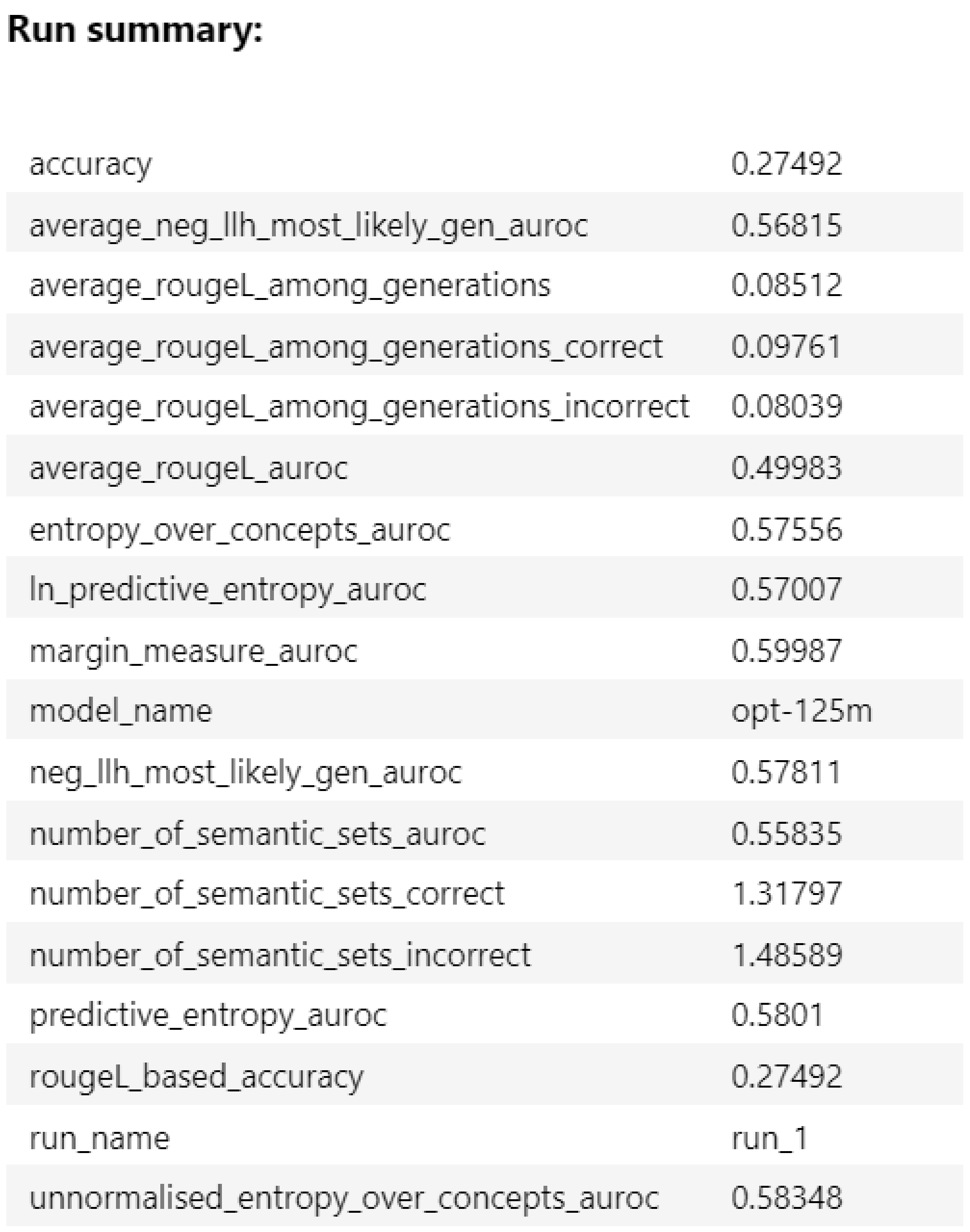
```
# Extract relevant columns from the comprehensive dataframe for accuracy
verification
accuracy_check_dataframe = comprehensive_dataframe[['most_likely_generation',
'answer', 'correct']]

# Save the accuracy verification data to a CSV file
accuracy_check_dataframe.to_csv('accuracy_verification.csv')
```

9 Results:

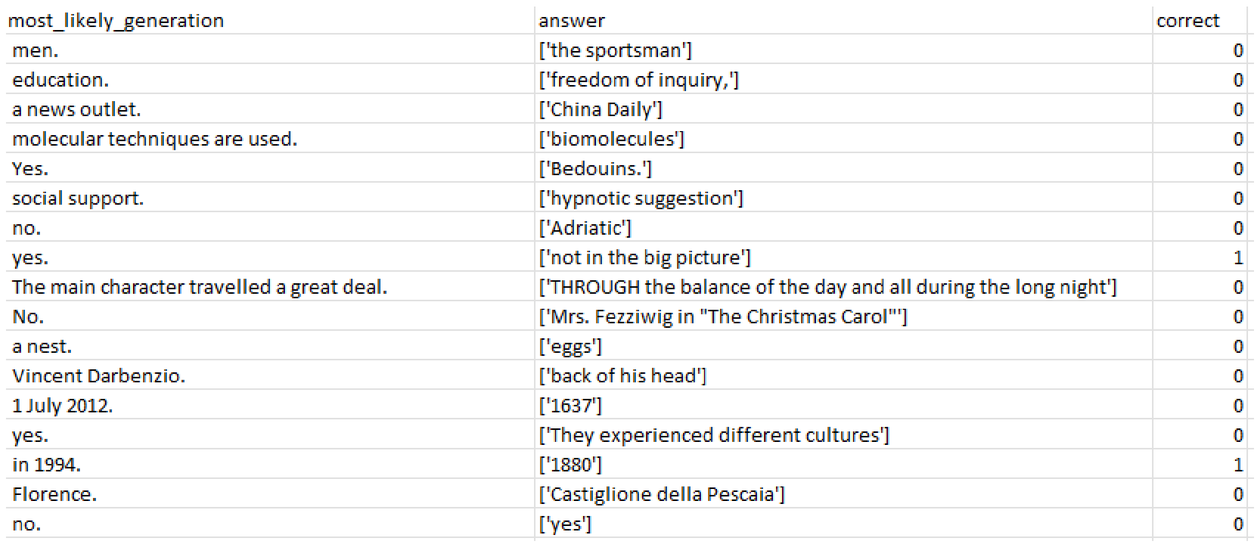
In our study on black-box LLMs, we aimed to quantify uncertainty in response quality across diverse questions. We developed and tested metrics for uncertainty and confidence. Our results indicate that using similarities from an NLI model, along with metrics assessing dispersion, effectively identifies challenging questions and confident answers.

OPT-125M (CoQA Dataset)



Run summary:

accuracy	0.27492
average_neg_llh_most_likely_gen_auroc	0.56815
average_rougeL_among_generations	0.08512
average_rougeL_among_generations_correct	0.09761
average_rougeL_among_generations_incorrect	0.08039
average_rougeL_auroc	0.49983
entropy_over_concepts_auroc	0.57556
ln_predictive_entropy_auroc	0.57007
margin_measure_auroc	0.59987
model_name	opt-125m
neg_llh_most_likely_gen_auroc	0.57811
number_of_semantic_sets_auroc	0.55835
number_of_semantic_sets_correct	1.31797
number_of_semantic_sets_incorrect	1.48589
predictive_entropy_auroc	0.5801
rougeL_based_accuracy	0.27492
run_name	run_1
unnormalised_entropy_over_concepts_auroc	0.58348

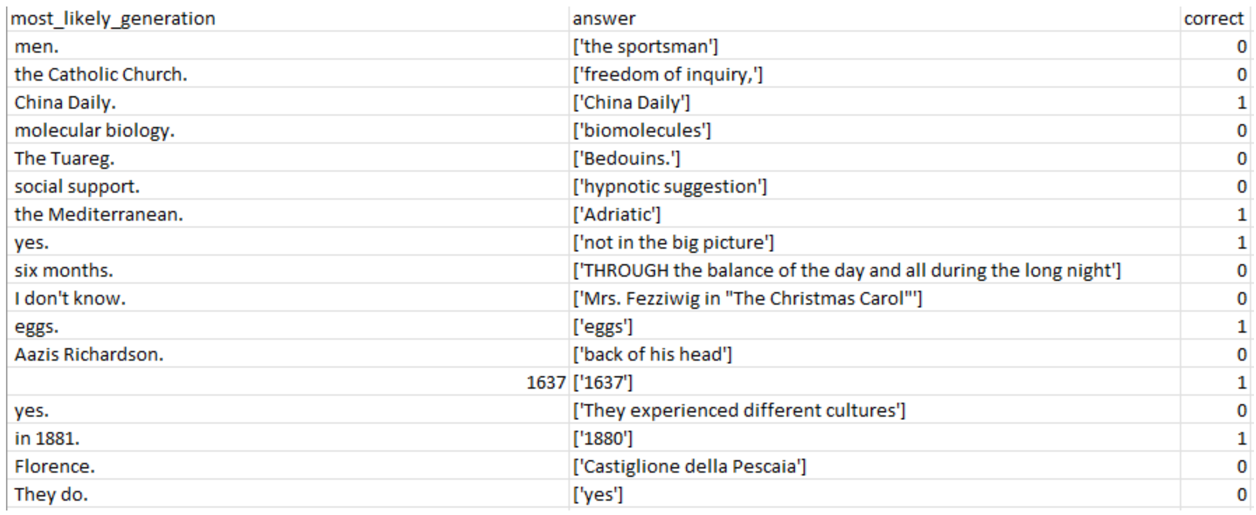


most_likely_generation	answer	correct
men.	['the sportsman']	0
education.	['freedom of inquiry,']	0
a news outlet.	['China Daily']	0
molecular techniques are used.	['biomolecules']	0
Yes.	['Bedouins.']	0
social support.	['hypnotic suggestion']	0
no.	['Adriatic']	0
yes.	['not in the big picture']	1
The main character travelled a great deal.	['THROUGH the balance of the day and all during the long night']	0
No.	['Mrs. Fezziwig in "The Christmas Carol"']	0
a nest.	['eggs']	0
Vincent Darbenzio.	['back of his head']	0
1 July 2012.	['1637']	0
yes.	['They experienced different cultures']	0
in 1994.	['1880']	1
Florence.	['Castiglione della Pescaia']	0
no.	['yes']	0

OPT-350M (CoQA Dataset)

Run summary:

accuracy	0.3988
average_neg_llh_most_likely_gen_auroc	0.59308
average_rougeL_among_generations	0.11742
average_rougeL_among_generations_correct	0.13015
average_rougeL_among_generations_incorrect	0.10897
average_rougeL_auroc	0.49073
entropy_over_concepts_auroc	0.62563
ln_predictive_entropy_auroc	0.62683
margin_measure_auroc	0.58221
model_name	opt-350m
neg_llh_most_likely_gen_auroc	0.58449
number_of_semantic_sets_auroc	0.58246
number_of_semantic_sets_correct	1.33613
number_of_semantic_sets_incorrect	1.57583
predictive_entropy_auroc	0.60187
rougeL_based_accuracy	0.3988
run_name	run_1
unnormalised_entropy_over_concepts_auroc	0.60835



most_likely_generation	answer	correct
men.	['the sportsman']	0
the Catholic Church.	['freedom of inquiry,']	0
China Daily.	['China Daily']	1
molecular biology.	['biomolecules']	0
The Tuareg.	['Bedouins.']	0
social support.	['hypnotic suggestion']	0
the Mediterranean.	['Adriatic']	1
yes.	['not in the big picture']	1
six months.	['THROUGH the balance of the day and all during the long night']	0
I don't know.	['Mrs. Fezziwig in "The Christmas Carol"']	0
eggs.	['eggs']	1
Aazis Richardson.	['back of his head']	0
1637	['1637']	1
yes.	['They experienced different cultures']	0
in 1881.	['1880']	1
Florence.	['Castiglione della Pescaia']	0
They do.	['yes']	0

As it is clear from the results, the OPT-350M model provides us with a better accuracy of 39.88% as opposed to OPT-125M model which is only 27.49%. In general, for the CoQA dataset OPT-350M outperforms OPT-125M in all metrics. On further research, we found out that this improved performance might be the result of the concept of hallucination in LLM. Basically, the 125M model forgets the context after generating a certain amount of responses and that leads to vague and random answers which further lead to low scores.

In conclusion, it's essential to acknowledge certain limitations and challenges associated with our work that still need to be addressed. One notable limitation is that our evaluation of uncertainty and confidence metrics is currently confined to question-answering tasks. Expanding this evaluation to broader contexts, such as open-ended conversations, presents challenges. Specifically, determining the reliability of a response in open-ended dialogues is complex, as it's often challenging to obtain definitive labels on whether a given response can be deemed "reliable" or not.

10 Challenges (What did not work)

Throughout the course of our research, we encountered several challenges that shaped our methodologies and outcomes:

1. **Complex Contextual Understanding:** Grasping the intricate context and nuances of the foundational paper was our initial hurdle. The depth and complexity of the concepts presented a steep learning curve.
2. **Data Volume in TriviaQA:** The sheer volume of records in the TriviaQA dataset posed a significant challenge. Given its extensive size, processing and analyzing the data became a task, especially when juxtaposed with our limited computational resources. Although we started working
3. **Compute Constraints:** Our reliance on Google Colab, which offers limited compute units (100 hours for $10), frequently led to computational bottlenecks. This constraint necessitated a hybrid approach, where we had to judiciously allocate tasks between local machines and Colab based on their computational demands.
4. **RAM Requirements for Large Models:** The Opt125m and Opt350m models demanded substantial memory, requiring up to 40Gb of GPU RAM. This intensive requirement acted as a bottleneck, and due to these constraints, we had to forgo the use of the even larger Opt1.3b model.
5. **Graph Laplacian Implementation:** Our initial intent to implement the Graph Laplacian concept had to be shelved due to its complexity and our resource constraints. The alternative solution, while functional, proved to be non-trivial and was less efficient in terms of storage and RAM utilization.

These challenges, while formidable, provided valuable insights and shaped the trajectory of our research, ensuring a more robust and grounded outcome.

11 Future Work

Uncertainty and confidence quantification for large language models presents us with a plethora of opportunities specially when intertwined with the prospects of its use in prompt engineering.[18] which explains how calibrating models, ensuring that a model's confidence is aligned with its actual performance. Similarly we can Investigate how varying prompts, in terms of verbosity, specificity, or structure, influence the uncertainty and confidence levels of LLMs. This could provide insights into optimal prompt design for specific tasks.There are other opportunities as well where uncertainty quantification can be used model calibration or even iteratively increasing a models resilience and improving its results.

12 Refrences

1. Yuxia Wang, Daniel Beck, Timothy Baldwin, and Karin Verspoor. Uncertainty estimation and reduction of pre-trained models for text regression. Transactions of the Association for Computational Linguistics, 10:680–696, 2022.
2. Shrey Desai and Greg Durrett. Calibration of pre-trained transformers. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 295–302, Online, November 2020. Association for Computational Linguistics.
3. Amita Kamath, Robin Jia, and Percy Liang. Selective question answering under domain shift. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 5684–5696, Online, July 2020. Association for Computational Linguistics.
4. Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. arXiv preprint arXiv:2207.05221, 2022.
5. Stephanie C. Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words. ArXiv, abs/2205.14334, 2022.
6. Sabrina J. Mielke, Arthur Szlam, Y-Lan Boureau, and Emily Dinan. Linguistic calibration through metacognition: aligning dialogue agent responses with expected correctness. CoRR, abs/2012.14983, 2020.
7. Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In The Eleventh International Conference on Learning Representations, 2023.
8. Zhen Lin1, Shubhendu Trivedi, Jimeng Sun, Generating with Confidence: Uncertainty Quantification for Black-box Large Language Models.
9. Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pp. 255–269, 2021.
10. Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020.
11. Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, Jimmy Ba, LARGE LANGUAGE MODELS ARE HUMAN-LEVEL PROMPT ENGINEERS .

12. J. Florian Wellmann and Klaus Regenauer-Lieb. Uncertainties have a meaning: Information entropy as a quality measure for 3-D geological models. Tectonophysics, 526:207–216, 2012.
13. Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In Advances in Neural Information Processing Systems, 2017.
14. Dennis Ulmer, Lotta Meijerink, and Giovanni Cinà. Trust issues: Uncertainty estimation does not enable reliable ood detection on medical tabular data. In Emily Alsentzer, Matthew B. A. McDermott, Fabian Falck, Suproteem K. Sarkar, Subhrajit Roy, and Stephanie L. Hyland, editors, Proceedings of the Machine Learning for Health NeurIPS Workshop, volume 136 of Proceedingsof Machine Learning Research, pages 341–354. PMLR, 11 Dec 2020.
15. Malik Sajjad Ahmed Nadeem, Jean-Daniel Zucker, and Blaise Hanczar. Accuracy-rejection curves (ARCs) for comparing classification methods with a reject option. In Sašo Džeroski, Pierre Guerts, and Juho Rousu, editors, Proceedings of the third International Workshop on Machine Learning in Systems Biology, volume 8 of Proceedings of Machine Learning Research, pages 65–81, Ljubljana, Slovenia, 05–06 Sep 2009. PMLR.
16. Timo Schick, Jane Dwivedi-Yu, Roberto Dessì†, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, Thomas Scialom, Toolformer: Language Models Can Teach Themselves to Use Tools.
17. Generating with Confidence: Uncertainty Quantification for Black-box Large Language Models. Zhen Lin, Shubhendu Trivedi, Jimeng Sun
18. Calibration of Neural Networks using Splines. Kuleshov et al.