MUTIPLE CHOICE QA

Tancredi Bosi - 0001121897

Repository Github

TEXT MINING AND LLM

SET-UP SETTINGS

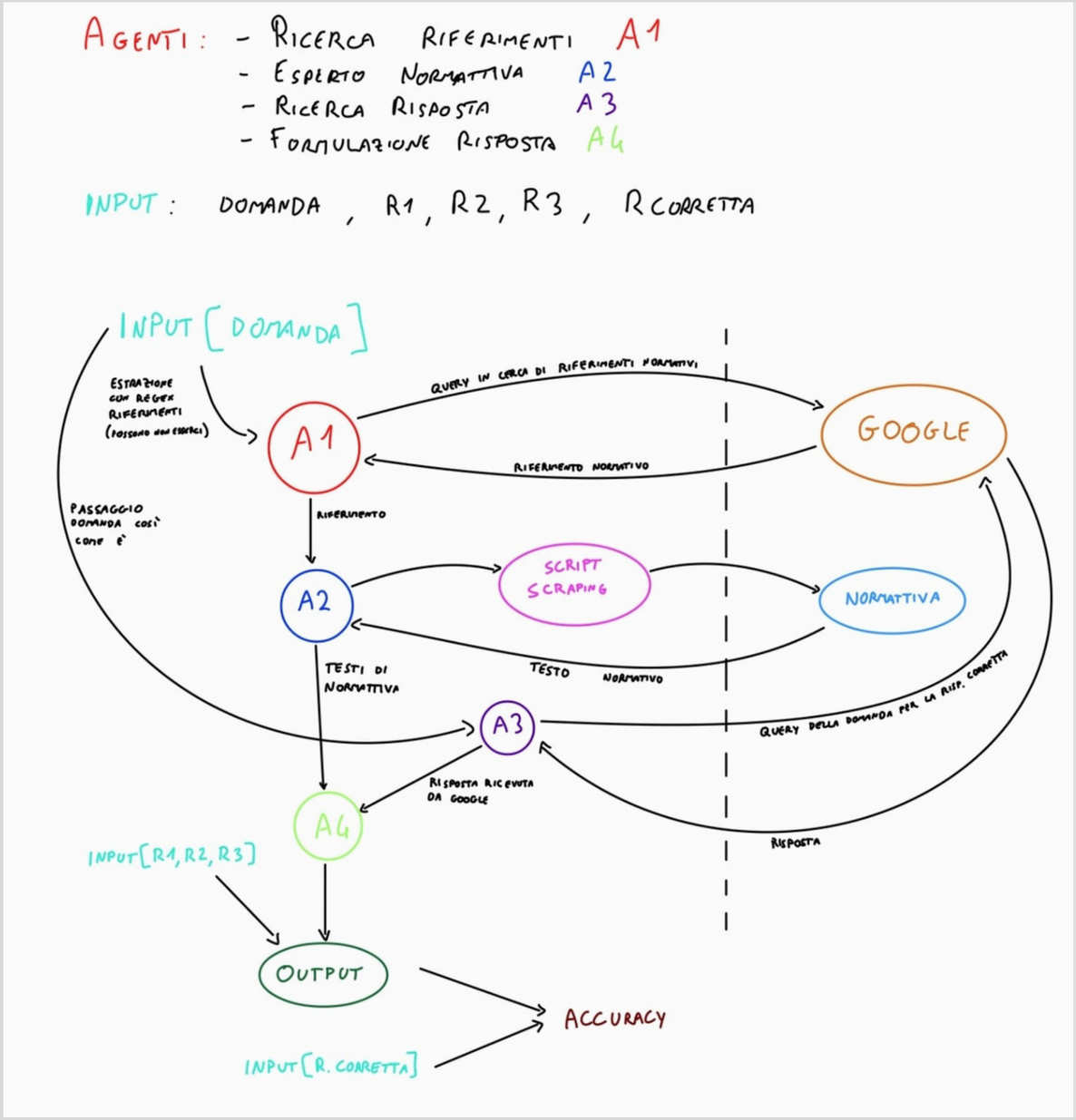
Files

- mcqa_codice_penale.json
- normattiva_scraper.py

  (slightly changed)

Environment

- Python 3.11
- Ollama
- langroid
- ftfy
- selenium
- numpy
- pandas
- bs4


AGENTI: - RICERCA RIFERIMENTI A1
- ESPERTO NORMATTIVA A2
- RICERCA RISPOSTA A3
- FORMULAZIONE RISPOSTA A4
INPUT: DOMANDA, R1, R2, R3, R CORRETTA
INPUT [DOMANDA]
ESTRAZIONE CON REGEX RIFERIMENTI (POSSONO NON ESSERCI)
A1
QUERY IN CERCA DI RIFERIMENTI NORMATIVI
GOOGLE
RIFERIMENTO NORMATIVO
PASSAGGIO DOMANDA COSÌ COME È
RIFERIMENTO
A2
SCRIPT SCRAPING
NORMATIVA
TESTO NORMATIVO
TESTI DI NORMATIVA
A3
QUERY DELLA DOMANDA PER LA RISP. CORRETTA
RISPOSTA RICEVUTA DA GOOGLE
A4
RISPOSTA
INPUT[R1, R2, R3]
OUTPUT
ACCURACY
INPUT[R. CORRETTA]

Workflow

Agents

 ReferenceFinder

 NormattivaExpert

 AnswerScraper

 AnswerCompiler

LLM configuration

AnswerScraper

Model: phi3.5

Context length: 16000

Max output tokens: 1000

Min output tokens: 200

AnswerCompiler

Model: phi3.5

Context length: 16000

Max output tokens: 50

Min output tokens: 1

Temperature: 0

ReferenceFinder

---

find_legal_references

checks for legal references in the input string using regex.

search_web_for_references

Formulates a Google search query, scrapes titles and descriptions of search results and calls find_legal_references

process_question

Agent's policy that calls find_legal_references and, if no reference is found, calls search_web_for_references. It than outputs the reference found.

```python
class NormattivaExpert(lr.Agent):
    def __init__(self):
        super().__init__(config=NE_agent_config)
        self.scraper = NormattivaScraper('/usr/local/bin/chromedriver')

    def fetch_legal_text(self, reference):
        """
        Uses Normattiva scraper to retrieve legal text based on the reference.
        """
        try:
            law_number = reference["number"]
            year = reference["year"]
            art_number = reference['article_number']
            self.scraper.navigate_to_page("https://www.normattiva.it/ricerca/avanzata")
            text = self.scraper.get_article_text(law_number, year, art_number)
            self.scraper.close()
            final_text = ''
            if reference['comma'] is None:
                for comma in text['commas'].values():
                    final_text += comma + '\n'
            else:
                final_text = text['commas'][reference['comma']]

            if final_text:
                return final_text
            else:
                return None
        except Exception as e:
            return None
```

NormattivaExpert

Agent responsible to call the NormattivaScraper functions to find legal text on https://www.normattiva.it given the input reference.

AnswerScraper

System message:

“Sei un agente che risponde in italiano con un testo completo e senza perdere dettagli a una domanda, basandoti su i risultati di una ricerca google.”

search_google

performs a Google search given the query and returns the top-5 results

find_unique_answer

gives the question and the data found on Google to the LLM and outputs a unique answer to the question

clean_data

cleans the content of the Google searches given in input and returns a list of dictionaries

get_answer

policy to perform the whole task using the methods explained before

AnswerCompiler

System message:

"Sei un agente incaricato di rispondere in italiano a una domanda selezionando la risposta corretta tra tre opzioni fornite. Rispondi immediatamente e unicamente con il numero (1, 2 o 3) corrispondente alla risposta giusta, senza aggiungere commenti o spiegazioni.
La tua risposta deve essere solo il numero della risposta corretta. Baserai la tua scelta esclusivamente sui seguenti input:

- domanda
- opzioni
- testo normativo
- risposta google"

```python
class AnswerCompiler(lr.ChatAgent):
    def __init__(self):
        super().__init__(config=AC_agent_config)

    def compile_answer(self, final_answer, final_text, question, choices):
        """
        Uses the LLM to compare the final_answer and final_text with the given choices and returns the correct answer.
        """

        combined_text = (f"domanda:{question}\n"
                         F"opzioni:\n"
                         f"1) {choices[0]}\n"
                         f"2) {choices[1]}\n"
                         f"3) {choices[2]}\n\n"
                         f"testo normativo: {final_text}\n\n"
                         f"risposta google: {final_answer}\n\n"
                        )

        response = self.llm_response(combined_text)
        response = str(response)[12:]

        # Extract the number of the correct answer from the response using regex
        correct_answer_number = None
        pattern = r'\b([123])\b'

       # Search for the answer where it should be
        match = re.search(pattern, response[:5])
        if match:
            correct_answer_number = match.group(1)
        else:
            # Check the whole response in case the LLM does not immediatly respond with the number of the answer
            match = re.search(pattern, response)
            if match:
                correct_answer_number = match.group(1)
        if correct_answer_number is None:
            correct_answer_number = 0
        correct_answer_number = int(correct_answer_number)
        return correct_answer_number, response
```

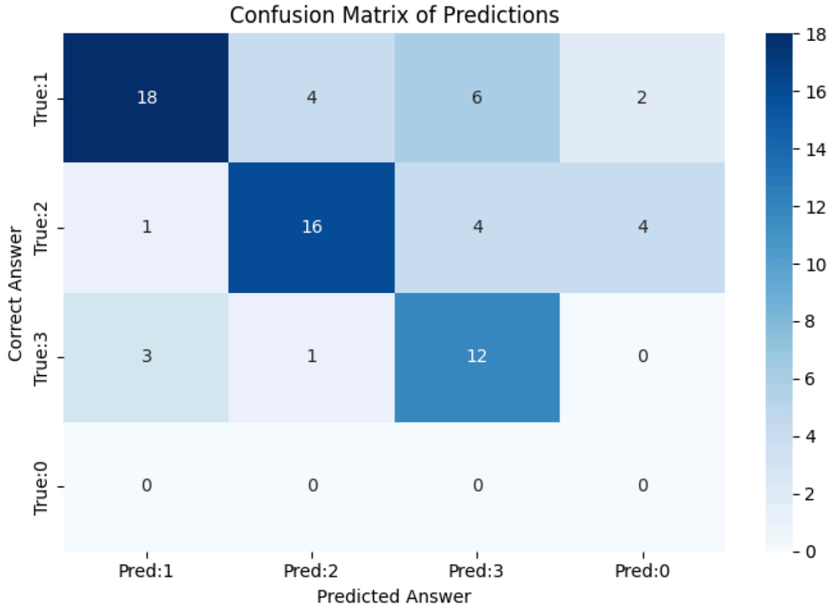
Results

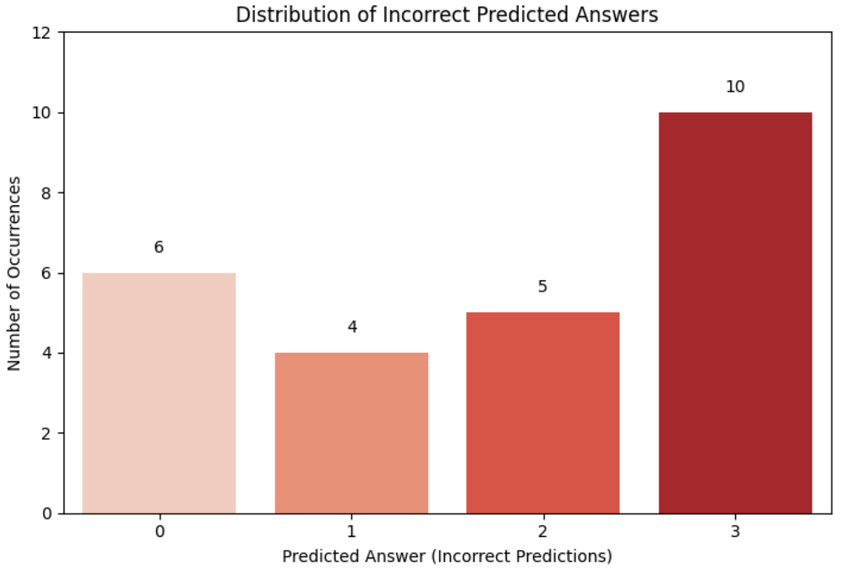
Accuracy: 64.79%

We can see that there is an high amount of LLM responses that do not produce an explicit answer and therefore is classified as 0, leading to loose something in performance terms.
Beside of that, the performances achieved are good, considering the fact that the LLM used by the agents is not specialized in legal domain, and in particular in italian legal domain.

Number of empty TESTO NORMATTIVA sections: 0

Number of empty RISPOSTA DA GOOGLE sections: 0





Ablation Studies

LLM without agents (2)

No Normatttiva text

No Google web search

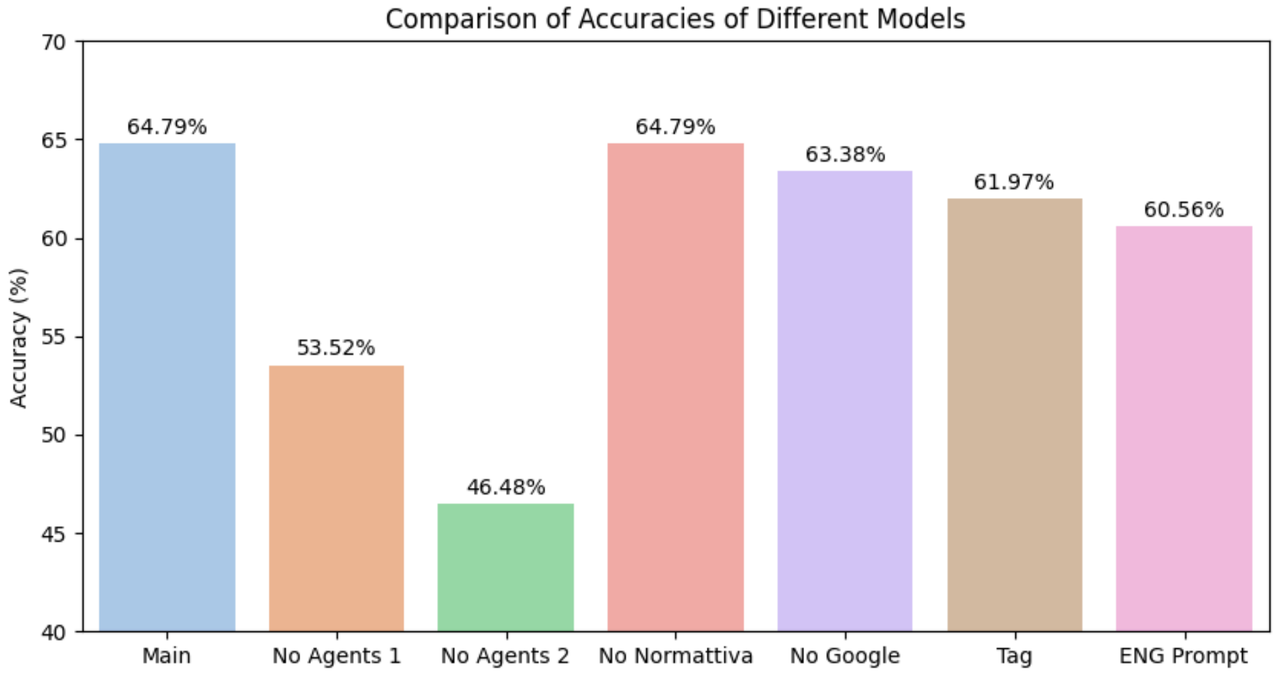
Answer after a tag

English prompts

Results Comparison

This bar chart compares the accuracies of various configurations.

- The "Main" model achieves the highest accuracy of 64.79%, tied with the "No Normattiva" model. Also, the "No Google" model predicts answers at the same level (only 1 correct prediction less).
- Using the LLM without agents results in significant performance drops: "No Agents 1" achieves 53.52%, and "No Agents 2" further decreases to 46.48%, probably because it tends to allucinate while producing the sentence before the tag.
- The "Tag" and "ENG Prompt" configurations maintain relatively good performance, with accuracies of 61.97% and 60.56%, respectively, but still fall short of the "Main" model, highlighting the effectiveness of give the LLM italian prompts and the difficulty that the LLM has while dealing with producing more text.

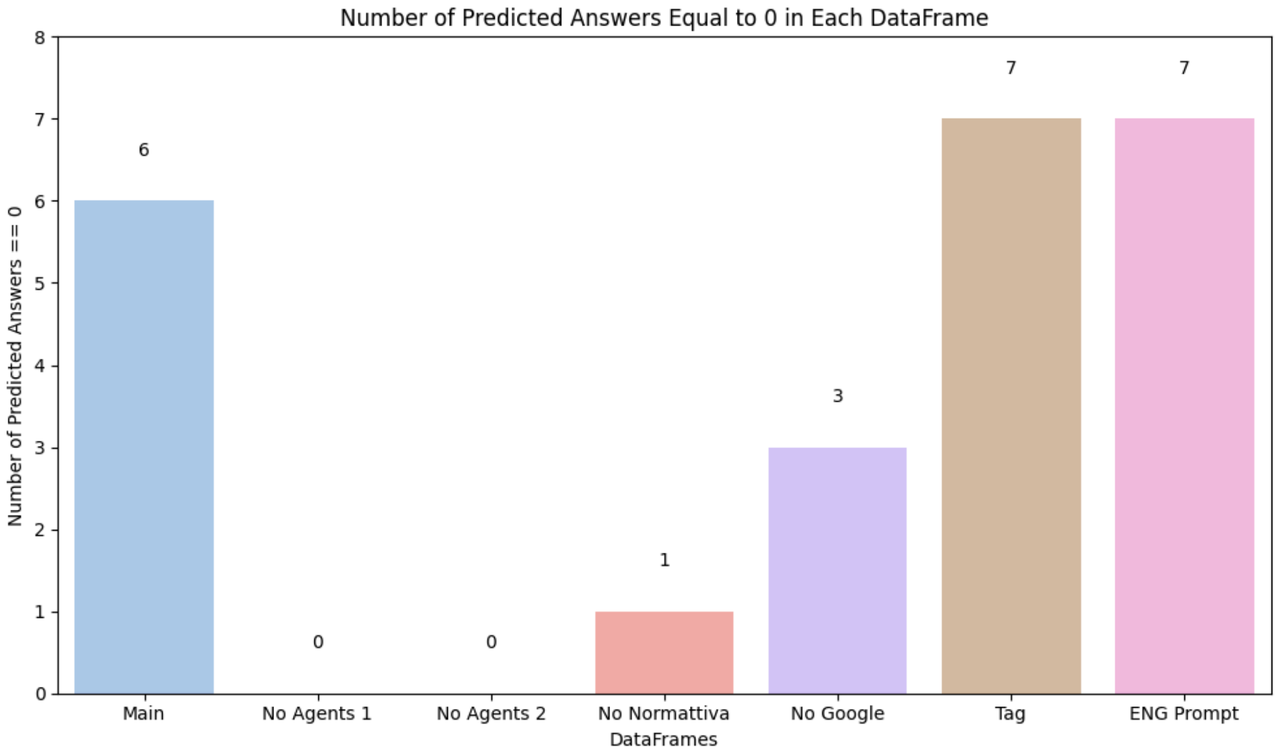


Overall, the chart emphasizes the importance of using a multi-agent system, giving some important context to the LLM.

Results Comparison

This bar chart illustrates the number of times the LLM fails to produce a valid answer:

- The "Main", "Tag" and "ENG Prompt" configurations exhibit the highest failure rates, indicating that these setups are least reliable in consistently providing answers. That is probably because these configurations are the ones that give the LLM a longer prompt, having either the Normattiva Legal text and the Google Answer. The performances can probably be improved with the use of a bigger LLM that can deal better with long input text.
- The trend of "long prompt --> more 0 predictions" is consistent also with the other configurations: the Normattiva Legal text tends to be longer than the Google Answer and therefore the configuration with only Normattiva Legal text produces more 0 predictions than the one with only the Google Answer.
- According with the considerations done, the two configurations with no input text, only the basic prompt, are the ones that produce no 0 prediction.

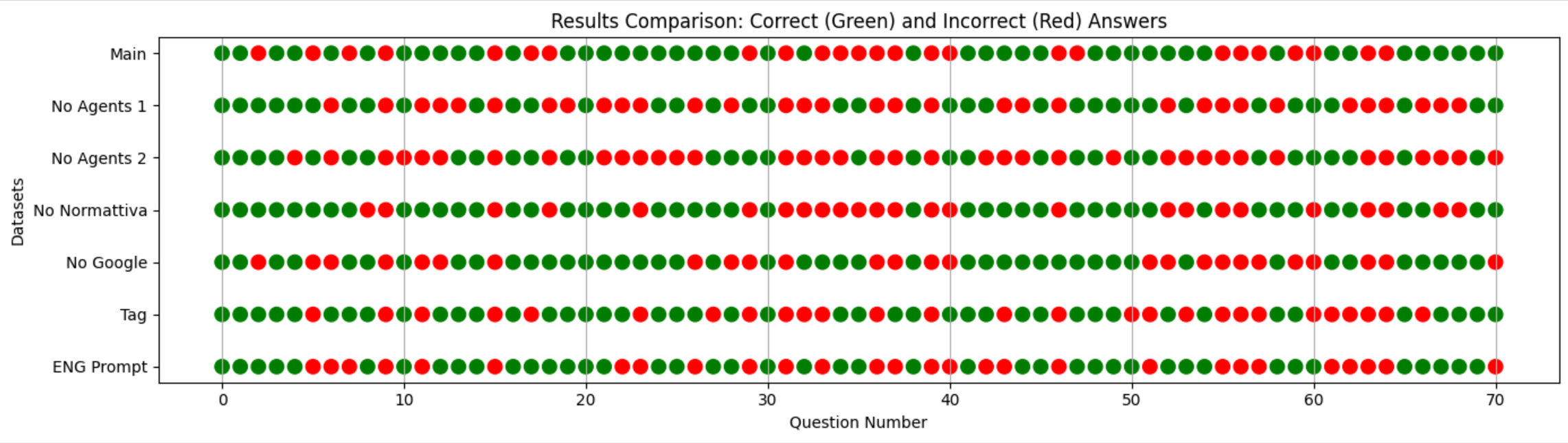


This chart underscores the trade-offs between accuracy and the ability to always generate a prediction.

Results Comparison

This plot shows the correct and incorrect answers performed by each model.

In particular, it can be seen that there are certain questions that are predicted in a wrong way by all the models or the majority of them. This is due to the question and multiple choice structures, being ambiguous, very similar or not simple to compute also thorugh the context given in input.



Conclusions

The results show that the "Main" complete system achieved the highest accuracy, making it the most effective setup. However, even when certain agents were removed, the system still performed well, showing the fact that it is important to have some useful context. In the study there also tested simpler configurations where only the LLM was used without agents. These setups had lower accuracy and more errors, highlighting the importance of the agents in improving performance.

One challenge identified was the issue of "zero predictions," where the system fails to choose an answer. This problem happened more often when the system had to process long and detailed prompts, such as those including both legal references and web search results. This suggests that using larger, more advanced LLMs or improving how the system presents information to the model could help reduce these errors.

Overall, this project demonstrates that combining agents with LLMs is a powerful approach for solving difficult tasks like legal question answering. The results provide valuable insights into how these systems can be improved further, such as by refining the agents, optimizing the way information is structured and presented to the model, or maybe add some fine-tuning in the legal domain.