SATYRN: A Platform for Analytics Augmented Generation

Marko Sterbentz **Cameron Barrie** **Shubham Shahi** **Abhratanu Dutta**
Donna Hooshmand **Harper Pack** **Kristian J. Hammond**
Northwestern University
{marko.sterbentz, cameron.barrie}@u.northwestern.edu
kristian.hammond@northwestern.edu

Abstract

Large language models (LLMs) are capable of producing documents, and retrieval augmented generation (RAG) has shown itself to be a powerful method for improving accuracy without sacrificing fluency. However, not all information can be retrieved from text. We propose an approach that uses the analysis of structured data to generate fact sets that are used to guide generation in much the same way that retrieved documents are used in RAG. This analytics augmented generation (AAG) approach supports the ability to utilize standard analytic techniques to generate facts that are then converted to text and passed to an LLM. We present a neurosymbolic platform, SATYRN, that leverages AAG to produce accurate, fluent, and coherent reports grounded in large scale databases. In our experiments, we find that SATYRN generates reports in which over 86% of claims are accurate while maintaining high levels of fluency and coherence, even when using smaller language models such as Mistral-7B, as compared to GPT-4 Code Interpreter in which just 57% of claims are accurate.

1 Introduction

In recent times, large language models (LLMs) have seen a meteoric rise in popularity due to their ability to generate fluent and coherent language. However, these models often struggle to produce truthful outputs (Ji et al., 2022; Huang et al., 2023; Borji, 2023). Methods such as retrieval augmented generation (RAG) (Lewis et al., 2021) have been developed to address such concerns. Yet, they too have an important limitation: they only work when the required information is present in textual form. For many sources of information, such as relational databases, the sought after information must be derived via computation and analysis of those data. Performing this analysis and providing the results to an LLM is another way to augment their generation with a far broader array of information. Such analytics augmented generation (AAG) supports the use of standard analytics techniques to generate facts that are converted to text and used to guide the generation of accurate responses.

This has motivated the development of methods in which an LLM invokes tools and external computation engines for deriving information from structured data (Schick et al., 2024; Masterman et al., 2024). This is also the approach of OpenAI's Code Interpreter[1] which can be used for generating reports that present information derived from structured data. However, these approaches let the LLMs decide when to call these tools and write the queries required to derive the information, an inherently non-deterministic procedure that can make them unreliable in producing the set of information to communicate in a report.

In this paper, we present SATYRN[2], a neurosymbolic platform which leverages AAG for producing factual, fluent, and coherent reports. SATYRN separates the analysis from the language generation process in order to steer LLMs using facts derived from relational databases. It deterministically produces the analysis plans required to satisfy the informational needs of a report, generates templated language describing the results, and *then* uses a language model to generate the final report. This separation of the analysis and generation steps along with the deterministic nature of the analysis planning improves factual accuracy compared to existing solutions like Code Interpreter. In our experiments on the report generation task, we see a factual accuracy of 86% using SATYRN compared to 57% using Code Interpreter.

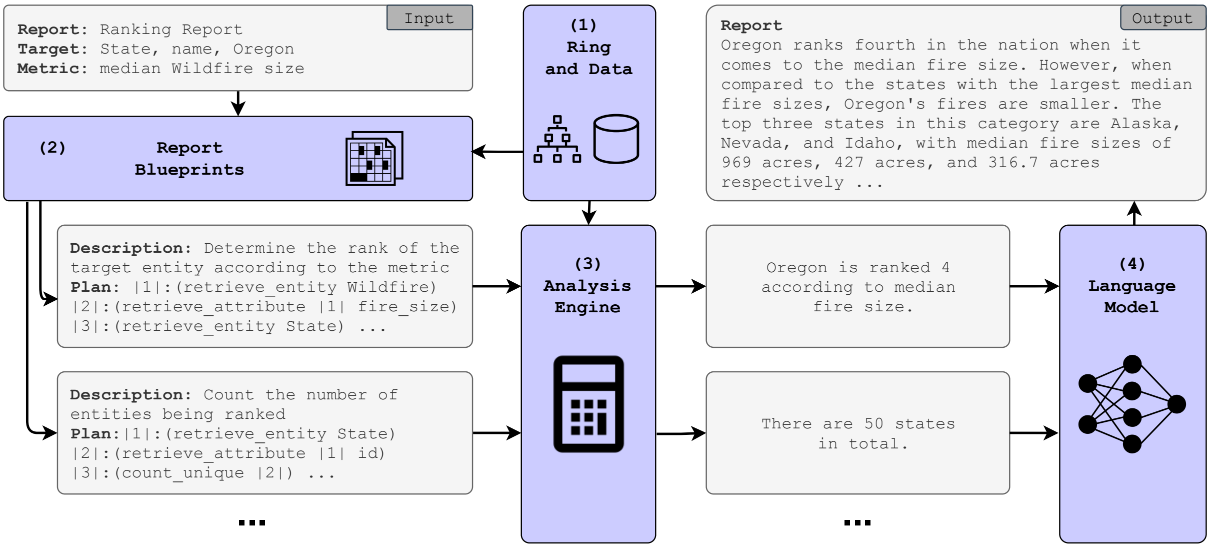
One of SATYRN's highlights is its ability to effectively utilize small, off-the-shelf models to achieve highly accurate results. In our experiments, the two local models we used were Mistral-7B and 4-bit

---

[1] https://platform.openai.com/docs/assistants/tools/code-interpreter

[2] https://github.com/nu-c3lab/satyrn

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 6360–6385
November 12-16, 2024 ©2024 Association for Computational Linguistics



Report
Oregon ranks fourth in the nation when it comes to the median fire size. However, when compared to the states with the largest median fire sizes, Oregon's fires are smaller. The top three states in this category are Alaska, Nevada, and Idaho, with median fire sizes of 969 acres, 427 acres, and 316.7 acres respectively ...

Description: Determine the rank of the target entity according to the metric
Plan: |1|:(retrieve_entity Wildfire) |2|:(retrieve_attribute |1| fire_size) |3|:(retrieve_entity State) ...

Description: Count the number of entities being ranked
Plan:|1|:(retrieve_entity State) |2|:(retrieve_attribute |1| id) |3|:(count_unique |2|) ...

Oregon is ranked 4 according to median fire size.

There are 50 states in total.

Figure 1: The high level approach of SATYRN and its analytics augmented generation.

quantized Mixtral-8x7B, each running on a single A6000 GPU. This showcases the ability of running SATYRN locally under resource constraints.

Our main contributions are as follows:

- First, we propose SATYRN, a scalable, domain-agnostic, and neurosymbolic platform for producing fluent and coherent reports that adhere to the source data (§2).
- Second, we demonstrate how SATYRN can be used for producing reports and compare it with other methods (§3). In our experiments, we generate 200 reports with over 3200 claims, across 3 broadly applicable report types and 8 domains and evaluate the reports for accuracy, fluency, and coherence.
- Third, we provide an analysis of the factual accuracy, fluency, and coherence of generated reports (§4). We find that reports produced by SATYRN strongly adhere to the data while maintaining high degrees of fluency and coherence, even when using smaller models.

2 Methods

In this section, we present the components of SATYRN that allow it to perform analytics on the data, irrespective of its domain, in order to generate accurate, fluent, and coherent reports grounded by this data. The high level architecture of SATYRN is shown in Figure 1. Beginning with the data in (1), we create a lightweight knowledge representation, called a *ring*, that provides simple semantic labels that describe how the data can be analyzed by SATYRN. We introduce a new plan representation language called Structured Question Representation (SQR) (pronounced *seeker*) that allows complex analyses to be specified according to the semantic enhancements.

The input to SATYRN is a structured information request and the output is a report satisfying this request with an answer and contextualizing information. This request is a JSON object describing the type of report to generate, the subject of the report, a metric to use when evaluating the subject, and any filters to apply to the data before analysis. Report blueprints (2) specify the information to be derived via analysis of the data and are used to produce a set of executable plans. With the analysis engine (3), each of these plans is executed in order to derive the information. The outputs are formatted as statements and used as input to the LLM (4) which generates the final report.

2.1 SATYRN Rings

SATYRN requires knowledge of the objects the data describes in order to apply analytics and produce information regardless of a dataset's domain.

To achieve this, we define a lightweight labeling called a *ring* which specifies the entities in the data, their attributes, and the relationships between these entities. This labeling allows SATYRN to run analyses on the data without the need for any domain information. New datasets can be utilized within SATYRN by creating a ring and filling in the necessary details described below. Figure 2 depicts

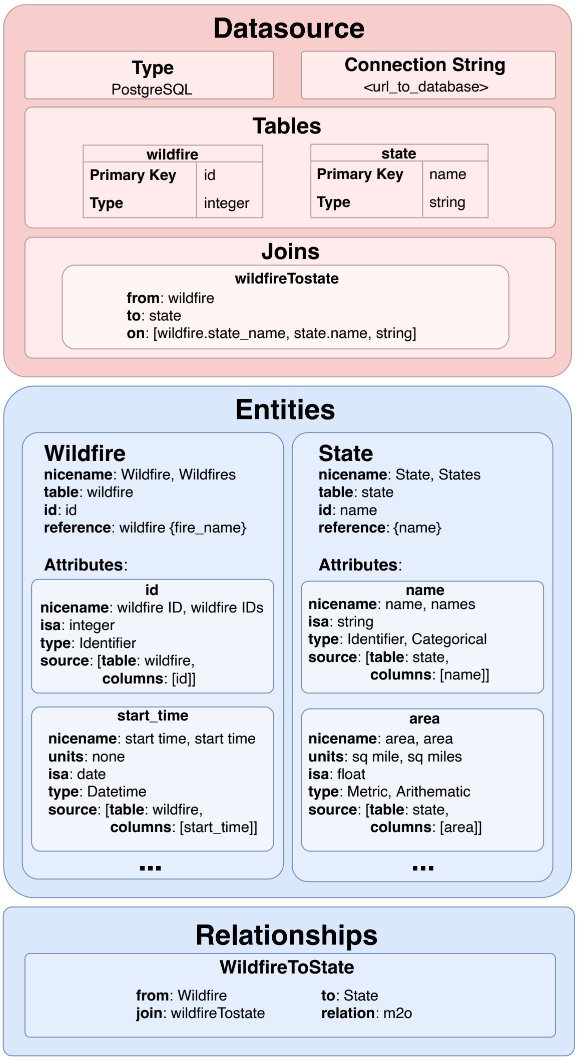


Figure 2: An example of a SATYRN ring with two entities defined: State and Wildfire.

an example of the makeup of a ring.

The main definitions in a SATYRN ring are:

- **Entities**: objects of interest described by the data, such as a person, place, or thing. For example, in Figure 2, there are entities for Wildfire and State.
- **Attributes**: properties of an entity. Each entity has one or more attributes that map to columns in the underlying database tables.
- **Relationships**: the connection between two entities in the ring.

In order for SATYRN to decide how to apply analytics to attributes, it needs a finer grained characterization of them than is provided by the database. For instance, performing arithmetic operations on an attribute requires it to be numerical. To capture this, we define six primary attribute types:

- *Arithmetic*: numerical values for which mathematical operations make sense.
- *Categorical*: discrete values typically denoting a class or category.
- *Datetime*: values designating a date and time.
- *Document*: values containing free text.
- *Identifier*: values meant to be used as a unique identifier for an entity.
- *Metric*: values meant to be used as a measure for ranking and comparing entities.

Rings also contain domain-specific information. We define *nicenames* for both attributes and entities to provide more descriptive labels than column headers. Units (e.g., dollars, acres) can also be defined for attributes. This domain knowledge is used to help SATYRN better express the information it derives in natural language.

The knowledge encapsulated by a ring makes it simple to specify and execute analytic operations. For instance, attributes of an entity can be present in multiple underlying data tables necessitating joins between these tables when retrieving these values. We define these joins inside the ring, abstracting away the need to specify them at runtime when performing operations that span multiple database tables. Similarly, when SATYRN performs an analytic operation involving multiple entities, joins are needed to connect the underlying database tables. For each pair of entities, relationships encapsulate the required joins so these details can be abstracted away during the specification of the analytics.

2.2 Structured Question Representation (SQR)

In order to leverage the knowledge in a ring and effectively apply analytics, an expressive and compositional plan representation in which plans can be reused across datasets is required. To satisfy these representational needs, we define an analytic plan representation language, SQR, which abstracts away specific implementation details of the underlying query language. This simplifies the syntax and makes it agnostic to the data storage format and corresponding query language. SQR allows for the specification of plans in which the entities and attributes defined in the ring are retrieved and analyzed using analytic operations.

A SQR plan is represented as a directed acyclic graph that specifies an ordered series of steps to carry out, wherein operations are chained together in order to retrieve and analyze data. Each node of the graph represents an analytic operation whose output is fed to later steps that require the result. The inputs and outputs of these analytic operations utilize attribute types in order to determine how operations can be chained together to form a complete SQR plan. Existing operations enable aggregation (e.g., average, count, sum), comparison (e.g., greater than, exact, not), and data manipulation (e.g., sort, groupby, limit). A full listing of the SQR operations that are currently implemented within SATYRN is provided in Appendix D. New operations can be added to incorporate new analytic capabilities. Arbitrarily complex plans can be composed to satisfy any information goal that can be described with the available data and analytics. A SQR plan example can be seen in Figure 3.

Query languages like SQL require joins between tables to be explicit in the query. However, joins are not needed within SQR. Any required joins are encoded within the ring and are used in the definition of relationships between entities. Such joins are automatically added to the final query against the database when parsing SQR to SQL.

```
|1|    (retrieve_entity State)
|2|    (retrieve_entity Wildfire)
|3|    (retrieve_attribute |1| name)
|4|    (retrieve_attribute |2| size)
|5|    (groupby |3|)
|6|    (average |4| |5|)
|7|    (retrieve_attribute |2| year)
|8|    (exact |7| 2020)
|9|    (collect |3| |6|)
|10|   (return |9| |8|)
```

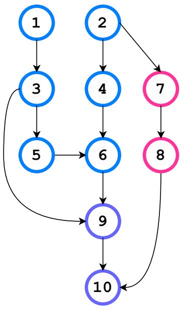


Figure 3: A SQR plan, in textual and graph forms, for determining average wildfire size for each state in 2020.

2.3 SQR Plan Templates

A SQR plan template is a SQR plan with slots to be filled with entities, attributes, values, and operations. All slots used within the SQR plan template correspond to the attribute types used in a ring, making the plan templates domain-agnostic and thus reusable with any ring. We define a set of SQR plan templates for use within SATYRN. Each SQR plan template has an associated statement template that describes the result of executing the plan template. These statement templates have slots for the natural language expression of specific entities, attributes and values, that correspond to the slots in the SQR plan templates.

2.4 Analysis Engine

SQR plans first need to be converted to a query format that is native to the data source (e.g., SQL for relational databases) in order to be executed to derive results. We build an analysis engine that is capable of parsing SQR plans, converting them to valid SQL queries against the underlying data, and producing the *nicenames* and units for all query results. Implementation details are provided in Appendix A.

2.4.1 Generation of Factual Statements

The output of the SQL query derived from a SQR plan is a set of raw result tuples that lack contextual information, making it difficult for an LLM to parse. To address this issue, the analysis engine can produce statements in natural language by utilizing the statement template associated with a SQR plan template and filling its slots with the values, *nicenames*, and units obtained by executing a SQR plan. Examples of generated statements can be seen in Figure 1 as the output of the Analysis Engine.

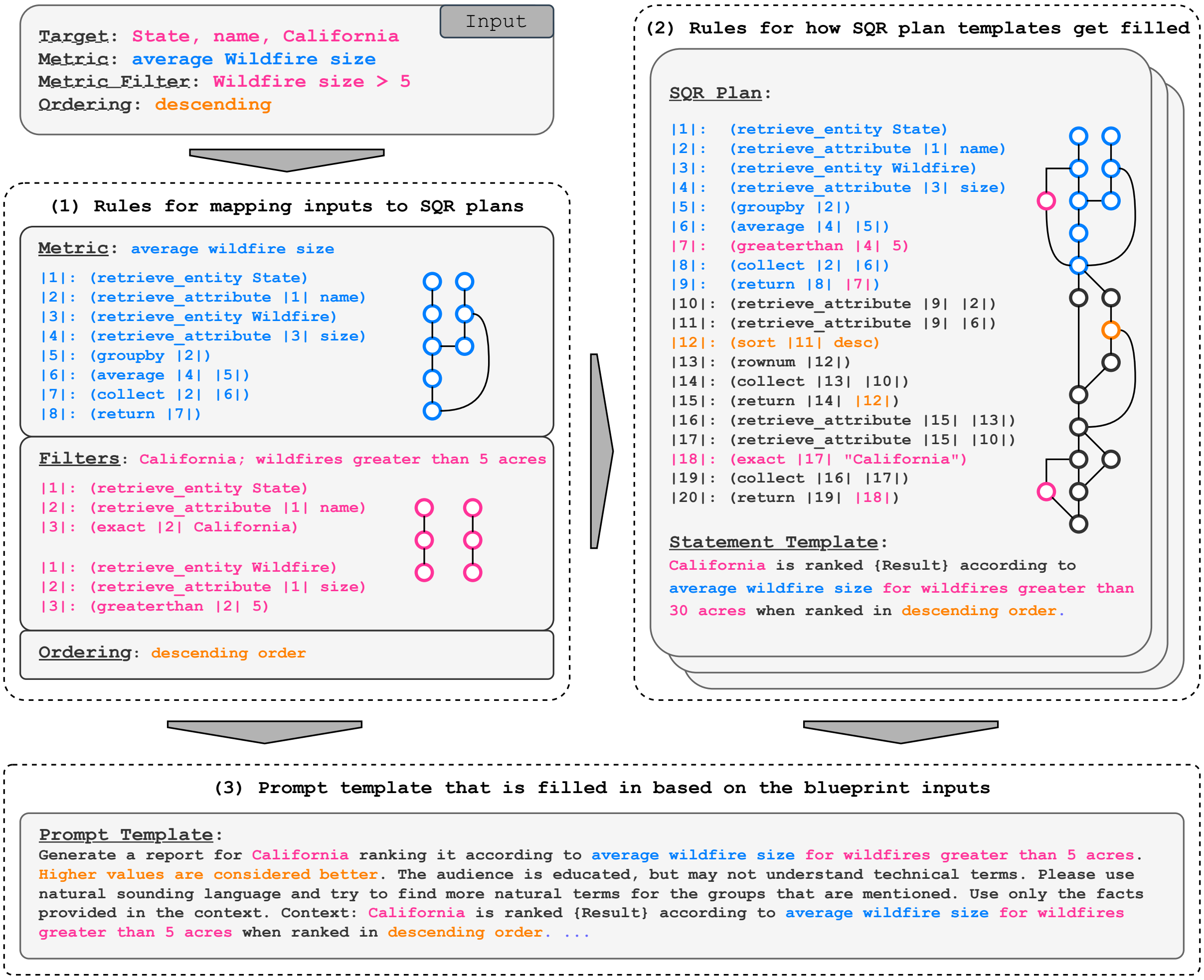
2.5 Domain-Agnostic Report Blueprints

For each kind of report, we create a report blueprint that defines the information that should be included in the report. Each blueprint comprises a collection of SQR plan templates that define the core analytics to perform in order to satisfy the information goals of this type of report. Inputs to a blueprint include the target entity, the metric of interest, and any filters that should be used to retrieve and filter the data for analysis. Additional inputs can also be defined as required.

Blueprints provide three key elements required for generating reports:

(1) **Rules for mapping inputs to SQR plans**. Rules map the target entity and any associated filters provided by the user to SQR plans for retrieving and filtering the data. For the metric, there is a rule that produces the SQR plan for deriving the value of this metric from the data.

(2) **Rules for how SQR plan templates get filled**. The SQR plans and filters that were produced with the rules in (1) are assigned to each SQR



```
Target: State, name, California
Metric: average Wildfire size
Metric_Filter: Wildfire size > 5
Ordering: descending
```

```
Metric: average wildfire size

|1|: (retrieve_entity State)
|2|: (retrieve_attribute |1| name)
|3|: (retrieve_entity Wildfire)
|4|: (retrieve_attribute |3| size)
|5|: (groupby |2|)
|6|: (average |4| |5|)
|7|: (collect |2| |6|)
|8|: (return |7|)
```

```
Filters: California; wildfires greater than 5 acres

|1|: (retrieve_entity State)
|2|: (retrieve_attribute |1| name)
|3|: (exact |2| California)

|1|: (retrieve_entity Wildfire)
|2|: (retrieve_attribute |1| size)
|3|: (greaterthan |2| 5)
```

```
Ordering: descending order
```

```
SQR Plan:

|1|:  (retrieve_entity State)
|2|:  (retrieve_attribute |1| name)
|3|:  (retrieve_entity Wildfire)
|4|:  (retrieve_attribute |3| size)
|5|:  (groupby |2|)
|6|:  (average |4| |5|)
|7|:  (greaterthan |4| 5)
|8|:  (collect |2| |6|)
|9|:  (return |8| |7|)
|10|: (retrieve_attribute |9| |2|)
|11|: (retrieve_attribute |9| |6|)
|12|: (sort |11| desc)
|13|: (rownum |12|)
|14|: (collect |13| |10|)
|15|: (return |14| |12|)
|16|: (retrieve_attribute |15| |13|)
|17|: (retrieve_attribute |15| |10|)
|18|: (exact |17| "California")
|19|: (collect |16| |17|)
|20|: (return |19| |18|)
```

```
Statement Template:
California is ranked {Result} according to
average wildfire size for wildfires greater than
30 acres when ranked in descending order.
```

(3) Prompt template that is filled in based on the blueprint inputs

```
Prompt Template:
Generate a report for California ranking it according to average wildfire size for wildfires greater than 5 acres.
Higher values are considered better. The audience is educated, but may not understand technical terms. Please use
natural sounding language and try to find more natural terms for the groups that are mentioned. Use only the facts
provided in the context. Context: California is ranked {Result} according to average wildfire size for wildfires
greater than 5 acres when ranked in descending order. ...
```

Figure 4: An instantiation of a report blueprint for ranking California against other states by average wildfire size. The resulting SQR plans will be executed and their results are inserted into the prompt to form the input to the LLM.

plan template according to the requirements of that plan template and the information goals of the report. Using these assignments, each SQR plan template is filled with the proper entities, attributes, and filters to produce an executable SQR plan.

(3) **A prompt template that is filled in based on the blueprint inputs**. This includes a description of the report to be generated, and the set of factual statements generated by SATYRN.

An example blueprint which shows how these three key elements are used for a specific user input can be seen in Figure 4. The blueprint rules make use of the attribute types defined in the ring. This allows inputs from the users to be mapped to plans irrespective of any domain specifics. As a result, blueprints scope across any domain and are reusable for any ring which is provided to SATYRN. Implementation details for the blueprints are provided in Appendix B.

2.6 Execution of a Report Blueprint

Blueprint execution requires an input of a target entity, relevant metric information, and filters that are to be applied. Using the defined rules, the plan templates in the blueprint gets populated to provide executable SQR plans. The analysis engine executes these plans and generates language describing the results using the statement templates associated with the plan templates. Finally, the prompt template defined in the blueprint gets populated with the generated factual statements and is sent to the language model to generate a coherent, fluent, and factual report. Example report outputs can be seen in Appendix F.

3 Experiments

We evaluate the performance of SATYRN along with two baselines on the task of generating reports for a specific information request. Reports are evaluated based on their accuracy, fluency, and coherence. We create three report blueprints: a **Ranking** report that ranks an entity's performance amongst its cohort, a **Time over Time** report that compares an entity's performance at two time periods, and a **Comparative Benchmark** report that compares an entity's performance with a benchmark value. Further details on the specific information requirements of each report type can be found in Appendix C.1. These three report blueprints are each used across eight domains: healthcare, environmental sustainability, urban housing, criminal justice, education, legal and judicial, socioeconomic, and business. A summary of the datasets can be found in Appendix C.4. We create a ring for each dataset.

3.1 Report Generation Modes

SATYRN: This generation mode utilizes SATYRN for generating the reports. It uses the factual statement generation detailed in Section 2.4.1 when building the prompt. We use three LLMs for generation: Mistral-7B (Jiang et al., 2023), Mixtral-8x7B (Jiang et al., 2024), and GPT-4. Mixtral-8x7B was quantized to 4-bits (Frantar et al., 2023).

SATYRN-Table (ablation): This mode is the same as SATYRN, except that factual statements are replaced by tables containing the results of executing the underlying SQL query. This is an ablation for determining the utility of the factual statements versus raw SQL results as prompting inputs.

Unaugmented GPT-4 (baseline): The first baseline generation mode tests the model's ability to use its parametric knowledge to generate a factual report. Instead of augmenting the generation with information derived via analysis, we provide the target entity, metric and a description of the information to be included in the report.

Code Interpreter (baseline): We also compare SATYRN with OpenAI's Code Interpreter which can take one or more data files, write code to extract information, and then generate natural language outputs. We provide the model with descriptions of the information to present in the prompt. Given the data and the prompt, it must determine how to derive this information and generate a report. We give the tool direct access to the same data used by SATYRN. However, due to limitations on the number of files and their sizes, we cannot generate reports for domains with larger datasets, namely the legal and judicial, environmental sustainability, and business domains. Thus, we generate reports with Code Interpreter for only five domains.

3.2 Report Evaluation

We use three metrics for evaluating reports: fluency, coherence, and accuracy. We evaluate the first two using the `summarization` checkpoint of UniEval (Zhong et al., 2022). The source documents used in the coherence scoring are the factual statements associated with that report. For determining the accuracy, we manually examine each report to find the percentage of claims made that are supported by facts derived from the data. The ground truth facts are produced by SATYRN and used when evaluating across all modes. Discrepancies in the reports generated by the baselines and the facts generated by SATYRN were manually checked by examining the data to ensure fairness. The accuracy evaluation was carried out in two steps: claim identification and claim classification. Each claim is classified as either **factual** (supported by the data), **refuted** (contradicted by the data), or **confabulated** (not inferrable from the data). For full class definitions and the rubric used for identifying and classifying claims, see Appendix E. Each report was evaluated by two of the authors, and we computed an inter-annotator agreement score with Krippendorff's α (Hayes and Krippendorff, 2007) using the ratio of factual to total claims, resulting in an α of 0.86.

4 Results

In terms of factual accuracy, SATYRN outperforms the closest baseline by 29 points, even with smaller models, while maintaining similarly high degrees of fluency and coherence as measured by UniEval. These primary results are captured by Table 1.

4.1 Factual Accuracy

Figure 5 shows the difference in the fraction of claims that were classified as factual for each generation mode. The two baselines, Unaugmented GPT-4 and Code Interpreter, had 48% and 57% factual accuracy respectively. The ablated SATYRN-Table mode was comparable with both of the baselines, even with the smallest model, Mistral-7B. In this ablation study, we observed that larger models were much better at processing information structured as tables. However, when we structure the information in natural language statements and pass these to the LLMs instead, the differences in accuracy between the small and large models vanish entirely. Mistral-7B benefits the most with an increase of 34 points from its performance in the ablation. SATYRN outperforms the closest baseline

Generation Mode	Model	Fraction Factual ↑	Fraction Refuted ↓	Fraction Confabulated ↓	Fluency ↑	Coherence ↑
Unaugmented	GPT-4	0.477	0.108	0.416	0.928	0.971
Code Interpreter*	GPT-4	0.570	0.337	0.093	0.904	0.919
Satyrn-Table	Mistral-7B	0.548	0.314	0.138	0.887	0.946
	Mixtral-8x7B	0.628	0.194	0.178	0.890	0.970
	GPT-4	0.832	0.032	0.136	0.944	0.979
Satyrn	Mistral-7B	**0.891**	0.038	**0.071**	0.887	**0.980**
	Mixtral-8x7B	0.863	0.034	0.103	0.878	0.929
	GPT-4	0.863	**0.011**	0.126	**0.946**	0.977

Table 1: Average accuracy, fluency, and coherence scores for all models and generation modes used in the evaluation. *Code Interpreter only evaluated on five of the eight domains due to data file size constraints.

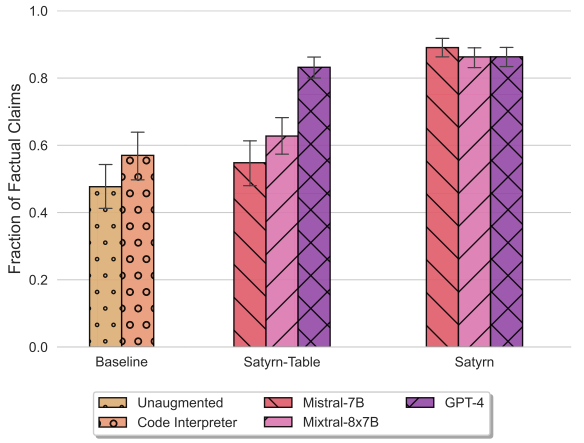


Figure 5: The fraction of claims classified as factual, rather than confabulated or refuted.

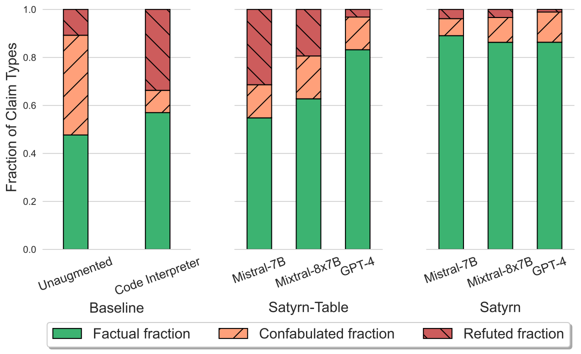


Figure 6: The fraction of claims that were classified as factual, confabulated, or refuted.

by 32 points with a much smaller model.

4.2 Fluency and Coherence

We find that reports produced by each generation mode exhibit high degrees of fluency as shown in Table 1. Reports generated by GPT-4 show slightly higher fluency scores than those generated by the smaller models. This is expected behavior since LLMs trained on larger datasets with more parameters are found to exhibit improved generative capabilities (Kaplan et al., 2020; Hoffmann et al., 2022). A similar result is found for the coherence of the reports, with all generation modes demonstrating high levels of coherence as shown in Table 1.

4.3 Claim Type Breakdown

In Figure 6, we normalize by the total number of claims, and present a breakdown of the fraction of factual, confabulated, and refuted claims. An interesting observation here is the distinction between the types of unsupported claims made by Unaugmented GPT-4 versus Code Interpreter. Unaugmented GPT-4 makes a lot of confabulated claims since it does not have access to any data or knowledge source. It also makes a lot of vague, but true statements (e.g., "This value is higher than the minimum value but not quite at the maximum value."). On the other hand, Code Interpreter is often unable to correctly derive information from the data, resulting in refuted claims.

We observe that the Unaugmented GPT-4 baseline has far fewer claims than other modes as seen in Figure 7. This is due to many of the sentences in its reports containing placeholder values and few concrete claims of fact that can be evaluated (e.g., "In comparison with Y, Bond County ranks Z.").

5 Satyrn Platform

Satyrn is a platform for enabling users to produce fluent, coherent, and accurate reports that are grounded by their data. Satyrn's capabilities can be expanded with new datasets, new analytics, and new report types.

To utilize a **new dataset**, a ring can be created. Given a ring, all existing analyses can be applied to the data and all report blueprints can be used to generate reports with them. The only exception is when the data itself does not support a particular kind of report. For example, the Time over

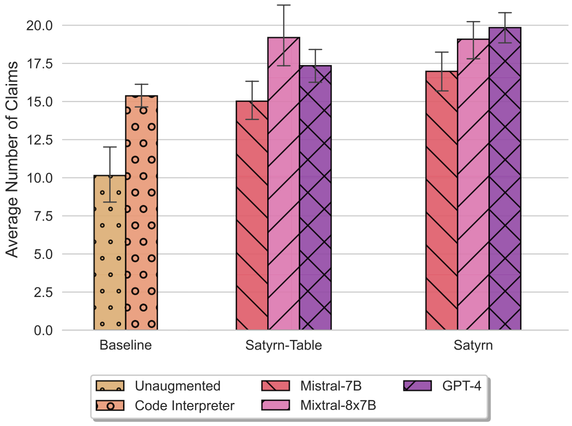


Figure 7: The average number of claims made for each configuration across all domains.

Time report used in the experiments requires the data be temporal so that analyses like change over time can be computed. To enable **new analytics** within Satyrn, additional SQR plan templates can be added to SATYRN. To enable a **new report type** to be generated, a report blueprint can be created. New blueprints can utilize any of the SQR plan templates available within SATYRN to deterministically guarantee the desired information is produced when generating a report. This cross-compatibility of all components of the platform ensures that SATYRN is scalable to new datasets, new analyses, and new report types.

6 Related Work

Domain and Analytics Representations Highly specialized ontologies have been developed for a diverse range of areas such as medicine (Salvadores et al., 2013), law (Casellas, 2011), food (Kamel Boulos et al., 2015), chemical engineering (Marquardt et al., 2010), and biological environments (Buttigieg et al., 2013). However, the production of ontologies such as these requires extensive expertise in ontology design and substantial amounts of time. (Patterson et al., 2019) have semantically enriched data science scripts with the goal of successfully modeling computer programs. However, their work focuses more on supporting automated reasoning about data science software rather than encoding core analytic knowledge and processes that can be used when mapping analytics onto data in a domain-agnostic fashion.

Knowledge Augmented Generation One key method to promote factual generation by an LLM is using an external knowledge source to augment the generation (Chen et al., 2017; Lewis et al., 2021; Shuster et al., 2021; Izacard et al., 2022; Siriwardhana et al., 2023). Knowledge graphs (Min et al., 2020; Baek et al., 2023), textual documents (Paranjape et al., 2021; Trivedi et al., 2023), preprocessed vectors (Verga et al., 2021), search engines (Nakano et al., 2021), and even other LLMs (Shwartz et al., 2020) have all been used as external knowledge bases. External symbolic engines have also been used to perform computation or reasoning, the results of which are used to augment an LLM's generation (Schick et al., 2024; Zhuang et al., 2023; Peng et al., 2023). While our work also utilizes a symbolic engine, we enhance relational databases with lightweight semantics in order to map the data to the operations of this engine and obtain results in natural language.

Data-to-Text Generation A related task of data-to-text generation where the goal is to generate descriptions of structured data organized in tables has been studied for a long time (Kukich, 1983; Reiter and Dale, 2000). Traditionally, template based algorithms were used to build data-to-text systems (Oh and Rudnicky, 2000; Stent et al., 2004; Kondadadi et al., 2013), while recent approaches have adopted a plan-then-generate procedure (Su et al., 2021). Currently, the most popular method is to use end-to-end neural pipelines where the model is fine-tuned to produce text from data (Puduppully et al., 2019; Yang et al., 2021; Ghosal et al., 2023; Zhao et al., 2023). Our approach differs in three key ways: 1) we target large scale databases rather than small tabular data, 2) we use an LLM for generation with no fine-tuning, and 3) our approach enables information not already present in the data to be computed at run-time for use in generation.

7 Conclusion

We present SATYRN, a system that leverages AAG to produce highly accurate, fluent, and coherent reports that adhere to information derived from data. We find that SATYRN generates reports that contain over 86% accurate claims as compared to GPT-4 Code Interpreter in which just 57% of claims are accurate. Notably, this is accomplished using a far smaller model, while preserving high levels of fluency and coherence. In the future, we plan to develop automatic validation methods to further improve the accuracy of generated reports. Additionally, we plan to allow language to be used as input rather than structured information requests.

Limitations

In this section, we discuss limitations and future directions for this work.

Input to SATYRN is not language: The input to SATYRN is currently a structured information request containing the type of report to generate, the subject of the report, a metric to use when evaluating the subject, and any filters to apply to the data before analysis. While this structure guarantees SATYRN will generate the correct kind of report, it is also more cumbersome to specify than a request in natural language. We plan to address this limitation by utilizing SATYRN's knowledge of the questions it can answer based on the available analytics to build a dataset question and SQR plans pairs in order to train a model for converting questions to SQR plans.

Report structure is encoded by model: One aspect of our report generation method that makes it somewhat limited is our reliance on the LLM to encode the structure of the report type rather than controlling for this ourselves. While we do consider this somewhat desirable, as it prevents us from having to define an explicit structure for each new type of report we wish the system to generate, it also means that we have little control over the document structure and as such, the quality and coherence of the structure depends on how well it is encoded in the LLM.

Report validation: Our approach has another limitation that it shares with RAG approaches: validation. Automatic fact extraction and claim verification is an active area of research (Min et al., 2023; Wei et al., 2024; Song et al., 2024). Measuring the factual accuracy of the reports generated in our experiments involved us manually labeling each claim as factual, confabulated or refuted. Further research in the direction will greatly help in evaluating systems like ours, and other natural language generation methods, at a larger scale.

Ethics Statement

In this work we present a system that automates the process of generating documents from data. While the aim of SATYRN is to ground generated reports in truth, particularly as compared to an unguided LLM, the system does not preclude the generation of reports from factually flawed data. As such, it is contingent on the system's user to verify the accuracy and validity of raw datasets fed to the system.

Additionally, we rely on an LLM to generate the language of a report. Such models have encoded biases that could crop up in the final report and could result in biases or other undesirable language. It is conceivable that bad faith actors could work to manipulate the presentation of information to ultimately misinform people with a highly skewed version of the truth. This remains a challenge for any approach which uses an LLM for generating a document.

Acknowledgments

We would like to thank the Center for Advancing the Safety of Machine Intelligence (CASMI) for funding this work. We also thank Mohammed A. Alam, David Demeter, Alexander Einarsson, and Sergio Servantez for providing constructive feedback as well as the ACL ARR reviewers for their valuable comments.

References

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. *arXiv preprint arXiv:2306.04136*.

Michael Bayer. 2012. Sqlalchemy. In Amy Brown and Greg Wilson, editors, *The Architecture of Open Source Applications Volume II: Structure, Scale, and a Few More Fearless Hacks*. aosabook.org.

Ali Borji. 2023. A categorical archive of chatgpt failures. *arXiv preprint arXiv:2302.03494*.

Pier Luigi Buttigieg, Norman Morrison, Barry Smith, Christopher J Mungall, and Suzanna E Lewis. 2013. The environment ontology: contextualising biological and biomedical entities. *Journal of biomedical semantics*, 4(1):1–9.

Núria Casellas. 2011. *Legal ontology engineering: Methodologies, modelling trends, and the ontology of professional judicial knowledge*, volume 3. Springer Science & Business Media.

Center for Homeland Defense and Security. 2023. Shooting incidents at k-12 schools (jan 1970-jun 2022) - chds school shooting safety compendium. `https://www.chds.us/sssc/data-map/`.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. Gptq: Accurate post-training quantization for generative pre-trained transformers.

Deepanway Ghosal, Preksha Nema, and Aravindan Raghuveer. 2023. ReTAG: Reasoning aware table to analytic text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6310–6324, Singapore. Association for Computational Linguistics.

Andrew F Hayes and Klaus Krippendorff. 2007. Answering the call for a standard reliability measure for coding data. *Communication methods and measures*, 1(1):77–89.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. An empirical analysis of compute-optimal large language model training. *Advances in Neural Information Processing Systems*, 35:30016–30030.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*.

Illinois State Board of Education. 2022. 2022 report card public data set. `https://www.isbe.net/_layouts/Download.aspx?SourceUrl=/Documents/2022-Report-Card-Public-Data-Set.xlsx`.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Atlas: Few-shot learning with retrieval augmented language models.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Delong Chen, Wenliang Dai, Andrea Madotto, and Pascale Fung. 2022. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55:1 – 38.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Alistair Johnson, Lucas Bulgarelli, Tom Pollard, Leo Anthony Celi, Steven Horng, Roger Mark, and Massachusetts Institute of Technology [MIT]. 2023. Mimic-iv-ed demo v2.2. `https://physionet.org/content/mimic-iv-ed-demo/2.2/`.

Maged N Kamel Boulos, Abdulslam Yassine, Shervin Shirmohammadi, Chakkrit Snae Namahoot, and Michael Brückner. 2015. Towards an “internet of food”: food ontologies for the internet of things. *Future Internet*, 7(4):372–392.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Ravi Kondadadi, Blake Howald, and Frank Schilder. 2013. A statistical NLG framework for aggregated planning and realization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1406–1415, Sofia, Bulgaria. Association for Computational Linguistics.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *21st Annual Meeting of the Association for Computational Linguistics*, pages 145–150, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks.

Wolfgang Marquardt, Jan Morbach, Andreas Wiesner, Aidong Yang, and Onto CAPE. 2010. *A Re-Usable Ontology for Chemical Process Engineering*. Springer.

Tula Masterman, Sandi Besen, Mason Sawtell, and Alex Chao. 2024. The landscape of emerging ai agent architectures for reasoning, planning, and tool calling: A survey. *arXiv preprint arXiv:2404.11584*.

Sewon Min, Danqi Chen, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2020. Knowledge guided text retrieval and reading for open domain question answering.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore. Association for Computational Linguistics.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Alice H. Oh and Alexander I. Rudnicky. 2000. Stochastic language generation for spoken dialogue systems.

In *ANLP-NAACL 2000 Workshop: Conversational Systems*.

Andrew Paley, Andong L Li Zhao, Harper Pack, Sergio Servantez, Rachel F Adler, Marko Sterbentz, Adam Pah, David Schwartz, Cameron Barrie, Alexander Einarsson, et al. 2021. From data to information: automating data science to explore the us court system. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*, pages 119–128.

Ashwin Paranjape, Omar Khattab, Christopher Potts, Matei Zaharia, and Christopher D. Manning. 2021. Hindsight: Posterior-guided training of retrievers for improved open-ended generation.

Evan Patterson, Ioana Baldini, Aleksandra Mojsilovic, and Kush R. Varshney. 2019. Teaching machines to understand data science code by semantic enrichment of dataflow graphs.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with entity modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2023–2035, Florence, Italy. Association for Computational Linguistics.

E. Reiter and R. Dale. 2000. *Building Natural Language Generation Systems (Studies in Natural Language Processing)*. Cambridge: Cambridge University Press.

Manuel Salvadores, Paul R Alexander, Mark A Musen, and Natalya F Noy. 2013. Bioportal as a dataset of linked biomedical ontologies and terminologies in rdf. *Semantic web*, 4(3):277–284.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2024. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36.

Karen C. Short. 2022. Spatial wildfire occurrence data for the united states, 1992-2020.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Unsupervised commonsense question answering with self-talk. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4615–4629, Online. Association for Computational Linguistics.

Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, Tharindu Kaluarachchi, Rajib Rana, and Suranga Nanayakkara. 2023. Improving the domain adaptation of retrieval augmented generation (rag) models for open domain question answering. *Transactions of the Association for Computational Linguistics*, 11:1–17.

Yixiao Song, Yekyung Kim, and Mohit Iyyer. 2024. Veriscore: Evaluating the factuality of verifiable claims in long-form text generation. *arXiv preprint arXiv:2406.19276*.

Amanda Stent, Rashmi Prasad, and Marilyn Walker. 2004. Trainable sentence planning for complex information presentations in spoken dialog systems. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 79–86.

Yixuan Su, David Vandyke, Sihui Wang, Yimai Fang, and Nigel Collier. 2021. Plan-then-generate: Controlled data-to-text generation via planning.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

U.S. Bureau of Economic Analysis. 2022. Personal income in cook county, il. `https://fred.stlouisfed.org/series/PI17031`.

U.S. Bureau of Labor Statistics. 2023. Unemployed persons in cook county, il. `https://fred.stlouisfed.org/series/LAUCN170310000000004A`.

U.S. Census Bureau. 2022a. Estimate of median household income for cook county, il. `https://fred.stlouisfed.org/series/MHIIL17031A052NCEN`.

U.S. Census Bureau. 2022b. Estimate of people age 0-17 in poverty in cook county, il. `https://fred.stlouisfed.org/series/PEU18IL17031A647NCEN`.

U.S. Census Bureau. 2022c. Estimate of people of all ages in poverty in cook county, il. `https://fred.stlouisfed.org/series/PEAAIL17031A647NCEN`.

Pat Verga, Haitian Sun, Livio Baldini Soares, and William Cohen. 2021. Adaptable and interpretable neural MemoryOver symbolic knowledge. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3678–3691, Online. Association for Computational Linguistics.

Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Hu, Dustin Tran, Daiyi Peng, Ruibo Liu, Da Huang, Cosmo Du, et al. 2024. Long-form factuality in large language models. *arXiv preprint arXiv:2403.18802*.

Yang Yang, Juan Cao, Yujun Wen, and Pengzhou Zhang. 2021. Table to text generation with accurate content copying. *Scientific reports*, 11(1):22750.

Yelp Inc. 2023. Yelp open dataset. `https://www.yelp.com/dataset`.

Yilun Zhao, Boyu Mi, Zhenting Qi, Linyong Nan, Minghao Guo, Arman Cohan, and Dragomir Radev. 2023. OpenRT: An open-source framework for reasoning over tabular data. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 336–347, Toronto, Canada. Association for Computational Linguistics.

Ming Zhong, Yang Liu, Da Yin, Yuning Mao, Yizhu Jiao, Pengfei Liu, Chenguang Zhu, Heng Ji, and Jiawei Han. 2022. Towards a unified multi-dimensional evaluator for text generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2023–2038, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2023. Toolqa: A dataset for llm question answering with external tools. *arXiv preprint arXiv:2306.13304*.

Zillow Group, Inc. 2023. Housing data - zillow research. `https://www.zillow.com/research/data/`.

A Analysis Engine Implementation

A.1 Plan Parsing and Execution

Conversion of the graph-structured SQR plan into SQL is done by first breaking the graph into "subplans", where the result of one subplan functions as a data source for subsequent subplans. From each subplan, the necessary information to form an executable query, including entities, their attributes, analytics operations, and filters, is then identified. For example, in the plan in Figure 3, the attributes *size* and *year* are retrieved from the *Wildfire* entity while *name* is retrieved from the *State* entity. The *average* operation is applied to *size*, grouped by *name*. A filter is constructed such that each *Wildfire* considered are from the *year* 2000. From this information, the query is constructed using a query abstraction library. For relational databases we use the SQLAlchemy Python package (Bayer, 2012). In this last step, the ring is leveraged to convert the entity and relationship abstractions to the proper tables and joins.

A.2 SQL Object Relational Mapping

While SATYRN's analysis engine is designed to be extendable to a variety of data source types, it is currently only configured to execute queries against relational databases. Upon initialization, for each selected ring, a corresponding object relational mapping (ORM) is built using SQLAlchemy. The ORM provides a programmatic interface between the information defined in the ring, and the data stored in a relational database.

The ORM is constructed using configuration mappings defined in the ring, specifically, the tables, columns, and joins between tables. Unlike objects defined in the ring, the ORM objects hold a direct one-to-one correspondence with database objects; ORM entities correspond to tables, attributes correspond to columns, and relationships correspond to groups of joins.

Implicit Joins A major benefit provided by the abstractions of the ring is that, for a given domain, joins between tables need only be defined once – in the configuration mapping of the ring. No join information of any kind is required in plan definitions. Instead, the system leverages the joins and relationships defined in the ring to determine which SQL joins to use when attributes are selected corresponding to columns of different tables.

All necessary joins between tables within an entity are identified. These intra-entity joins are necessary when multiple attributes are specified as belonging to the same entity, but correspond to columns from different tables. Then, for each pair of entities, relationships are defined by collecting joins along the shortest path connecting the tables of the two entities.

B Report Blueprint Implementation

SATYRN provides a platform for defining new types of reports that scope across any domain and dataset. These report types are encoded by a set of report blueprints, and the report generation process uses them to produce the information to be included in the final report. The primary purpose of a report blueprint is to provide the set of core information requirements, defined by a set of SQR plan templates, and rules for retrieving or producing the required *access plans*, filters, and language template fillers that will ultimately be composed with each SQR plan template. This composition process results in an executable SQR plan for each information requirement of the blueprint, and the execution of these plans with the analysis engine produces the raw set of facts used as part of a prompt for generating the final report with a language model.

B.1 Attribute Access Plans

When SATYRN loads a ring, it generates an *access plan* for each of the attributes: a SQR plan that specifies how to retrieve this attribute from the data.

SATYRN also automatically produces additional attributes by applying analyses to the existing attributes. By combining the set of attributes defined within the ring with the set of available analytics that can be applied to each of them based on their associated attribute types, it is possible to automatically produce an additional set of attributes. For example, if "population" is present as an attribute and has type "Arithmetic", then "average population" would be created since SATYRN knows it can apply the average operation to attributes that are "Arithmetic". Each of these derived attributes has an associated access plan as well.

B.2 SQR Plan Templates

The SQR plan templates (previously described in §2.3) serve as the base for defining the analytic steps to perform given one or more access plans as input. These input access plans can have filters inserted into them which constrain the retrieval of these data to particular subsets. Each SQR plan template also has a language template associated with it whose slots can be filled via the expression of entities, attributes, and values specified in the SQR plan template.

B.3 Identifying Plan Composition Inputs

The inputs to a report blueprint are provided by a user as a JSON object and vary depending on the information requirements of that report. An example of the inputs for a ranking report blueprint can be seen in Figure 8. All currently implemented blueprints require a target entity instance, metric, and metric set filter as part of their input. The user input specifying the target entity instance is used to create an instance filter, which are SQR plan steps used to filter for data relating to this entity instance. The blueprint uses its predefined rules to specify that this filter should be added to particular access plans and designated as input to a SQR plan template.

This is similarly true for the metric input for the blueprint. If an attribute in the ring has one of its attribute types defined as a Metric, then it is considered a valid input for the blueprint. The metric specified by the user is looked up in the ring, its access plan is retrieved, and then is designated as input to one or more SQR plan templates in the blueprint according to the blueprint rules. The underlying mechanics of how the metric gets mapped to inputs of the SQR plan templates in the blueprint are the same no matter the dataset which is used. There are corresponding rules within the blueprint defining how metric set filters specified by the user are added to the designated SQR plan templates. Additional report-specific inputs from the user are mapped to the SQR plan templates in a similar fashion. These blueprints ensure SATYRN's production of information from data is non-probabilistic, and a report blueprint is guaranteed to produce the same executable plans for the same user inputs.

B.4 Executable Plans via SQR Composition

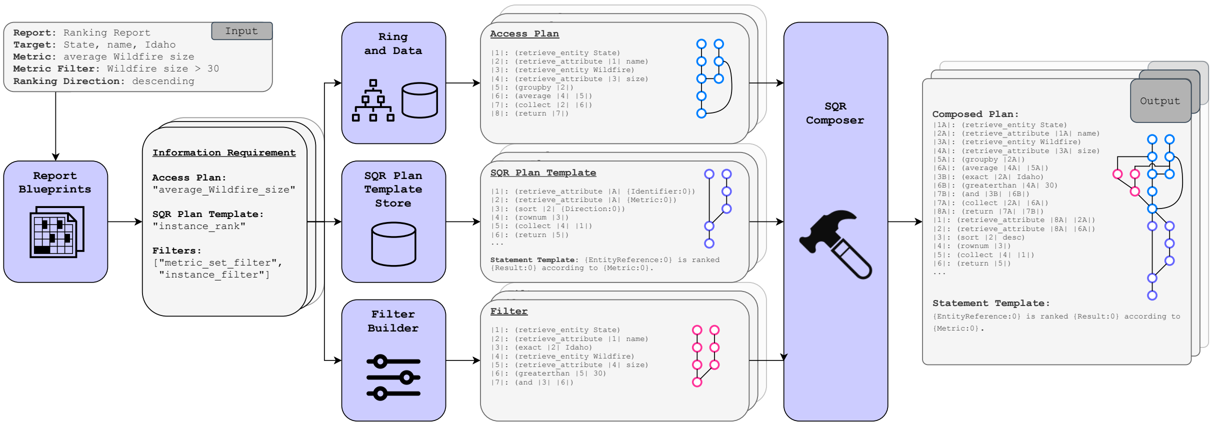
Ultimately, the rules encoded by the report blueprints, the retrieval of access plans from the ring, the retrieval of SQR plan templates from the SQR plan template store, and the production of filters by the filter builder result in composition specifications. These provide the raw materials for building an executable plan for each of the information requirements. The individual elements comprising the composition specification can be seen as input to the SQR Composer in Figure 8.

Prior to composition, the SQR Composer ensures that all the references of the access plans, SQR plan template, and filter are unique by appending a unique string to the step references of each SQR plan, preventing conflicts when combining them. Next, the SQR Composer updates the SQR plan template inputs with the correct access plan references (e.g. changing “|A|” to “|8A|” in Figure 8), and finally appends all the steps from the SQR plan template, access plan, and filters together to form an executable plan. SATYRN carries out this process for all information requirements of the report blueprint in order to produce a set of executable SQR plans that will be used for deriving the information that should be communicated in the report.

C Experimental Details

C.1 Report Types

We define three report blueprints that focus on common reporting use cases. Their information require-



```
Report: Ranking Report
Target: State, name, Idaho
Metric: average Wildfire size
Metric Filter: Wildfire size > 30
Ranking Direction: descending
```

```
Information Requirement

Access Plan:
"average_Wildfire_size"

SQR Plan Template:
"instance_rank"

Filters:
["metric_set_filter",
 "instance_filter"]
```

```
Access Plan

|1|: (retrieve_entity State)
|2|: (retrieve_attribute |1| name)
|3|: (retrieve_entity Wildfire)
|4|: (retrieve_attribute |3| size)
|5|: (groupby |2|)
|6|: (average |4| |5|)
|7|: (collect |2| |6|)
|8|: (return |7|)
```

```
SQR Plan Template

|1|: (retrieve_attribute |A| {Identifier:0})
|2|: (retrieve_attribute |A| {Metric:0})
|3|: (sort |2| {Direction:0})
|4|: (rownum |3|)
|5|: (collect |4| |1|)
|6|: (return |5|)
...

Statement Template: {EntityReference:0} is ranked
{Result:0} according to {Metric:0}.
```

```
Filter

|1|: (retrieve_entity State)
|2|: (retrieve_attribute |1| name)
|3|: (exact |2| Idaho)
|4|: (retrieve_entity Wildfire)
|5|: (retrieve_attribute |4| size)
|6|: (greaterthan |5| 30)
|7|: (and |3| |6|)
```

```
Composed Plan:
|1A|: (retrieve_entity State)
|2A|: (retrieve_attribute |1A| name)
|3A|: (retrieve_entity Wildfire)
|4A|: (retrieve_attribute |3A| size)
|5A|: (groupby |2A|)
|6A|: (average |4A| |5A|)
|3B|: (exact |2A| Idaho)
|6B|: (greaterthan |4A| 30)
|7B|: (and |3B| |6B|)
|7A|: (collect |2A| |6A|)
|8A|: (return |7A| |7B|)
|1|: (retrieve_attribute |8A| |2A|)
|2|: (retrieve_attribute |8A| |6A|)
|3|: (sort |2| desc)
|4|: (rownum |3|)
|5|: (collect |4| |1|)
|6|: (return |5|)
...

Statement Template:
{EntityReference:0} is ranked {Result:0} according to
{Metric:0}.
```

Figure 8: This figure shows the process SATYRN uses to generate a set of executable SQR plans to derive the information required for a report. First, the specified report type and its requirements are looked up in the set of report blueprints. The SQR plan template associated with each requirement is looked up from the SQR plan template store. Access plans are looked up in the attribute augmented ring. A filter plan is produced and all three components are passed to the SQR composer which assembles them into a single executable plan.

ments are shown in Table 2.

C.2 Hyperparameter Tuning

When performing the report generation with the language models, no hyperparameter search was performed, and we used default values for all model parameters with the exception of the following. For GPT-4 generations, we used a temperature of 0.0. For Mistral-7B and Mixtral 8x7B, we used a temperature of 0.2 and a top-p of 0.1.

C.3 Repeated Generation for Code Interpreter

Code Interpreter would often fail to produce any meaningful outputs or reports when given a prompt to do so. In order to ensure a fair comparison for evaluation, many of the prompts needed to be ran repeatedly in order for a report to be generated.

C.4 Datasets

Table 3 lists the 8 datasets with brief descriptions, in the 8 domains used in our experiments.

D List of SQR Operations

A listing of analytic operations that make up the SQR primitives can be seen in Table 4. This set of operations is easily extensible to new analytics.

E Accuracy Evaluation Details

In this section we present the definition of a claim (§E.1), the types these claims can be classified as (§E.2), and the full rubric used during the evaluation of each report's factual accuracy (§E.3).

E.1 Claim Identification

We define a claim as any assertion of truth involving some retrieval or analytic processing of information from the data. Interstitial writing that provides a transition between content, titles, and broad introductions to the report are not considered claims. We manually examine each generated report and identify the claims being made.

E.2 Claim Classification

Once a claim has been identified, we determine which of the following mutually exclusive categories this claim belongs to based on the factual statements generated for that kind of report.

Factual: The claim is directly stated by a fact in the context, or the claim can be directly inferred from the facts in the context via an unambiguous analytic process.

Refuted: The claim is directly refuted by a statement in the context, or the claim could be directly inferred from the facts in the context via an unambiguous analytic process, but is incorrect.

Confabulation: The claim is not directly pulled from the facts in the prompt, or is not inferred from the facts in the prompt.

E.3 Claim Evaluation Rubric

A copy of the rubric that was used for evaluation can be seen in Figure 9. It includes the instructions used for identifying claims made in a report as well as classifying those claims as factual, refuted, or confabulation.

Note that the confabulation category includes

Document Type	Information Requirements
Ranking	The value of the metric for the target entity
	The total number of entities being ranked
	The rank of the entity instance according to the metric
	The top three entity instances according to the metric
	How much lower the target entity is than the highest
	The average, minimum, and maximum value of the metric for all entities
	Whether or not the target entity is greater than the average value of the metric across all entities
Comparative Benchmark	The value of the metric for the target entity
	Whether this value is greater than the target benchmark value
	What the minimum, maximum, average, and median value of the metric is for all entities
	Whether or not the metric value for the target entity is greater than the average value of the metric for all entities
	Whether or not the metric value for the target entity is greater than the median value of the metric for all entities
	What the standard deviation of the metric is for all entities
Time over Time	The values of the metric at the start and end times for the target entity
	The percent change in these values
	The average, minimum, and maximum value of the metric for all entities at the start and end times
	The percent change between of the average between the start and end time for all entities
	Whether or not the percent change was greater for the target entity instance or the average for all entities

Table 2: The information requirements for each of the three report types used in our experiments.

Domain	Dataset	Description
Environmental Sustainability	Wildfire Occurrence	The Wildfire Data (Short, 2022) provides 2.3 million geo-referenced records on U.S. wildfires from 1992 to 2020, covering 180 million burned acres with key identifiers for data linkage.
Healthcare	MIMIC-IV-ED-Demo	MIMIC-IV-ED (Johnson et al., 2023) is a deidentified critical care database from BIDMC with 40,000+ patient records, organized modularly for easy access to diverse data sources while complying with HIPAA Safe Harbor.
Urban Housing	Zillow Observed Rent Index	Zillow Observed Rent Index (Zillow Group, Inc., 2023) is a representative dollar-denominated rental index, calculated from listed rents in the 40th to 60th percentile for all housing types in various regions.
Criminal Justice	School Shooting Incidents	The dataset (Center for Homeland Defense and Security, 2023) covers publicly available data on shooting incidents from 1970 to June 2022, including any instance of gun brandishing, firing, or bullets hitting school property, regardless of outcomes or timing.
Education	Illinois Report Card	The Illinois Report Card (Illinois State Board of Education, 2022), issued by the Illinois State Board of Education, provides annual educational progress data for the state, schools, and districts.
Legal and Judicial	SCALES	The SCALES dataset (Paley et al., 2021) combines data from PACER, including ten years of docket reports (2007-2016) from Northern Illinois district courts and 2016 district court reports, with the Federal Judicial Center's judge metadata.
Socioeconomic	Income Disparity	The U.S. Bureau of Economic Analysis' report (U.S. Bureau of Economic Analysis, 2022; U.S. Bureau of Labor Statistics, 2023; U.S. Census Bureau, 2022b,a,c) captures personal income data for various regions, showing income received by residents in those areas based on their place of residence.
Business	Yelp Open Dataset	The Yelp Open Dataset (Yelp Inc., 2023) comprises 5.9 million reviews, 188,593 businesses, and 280,992 pictures, provided by Yelp for personal, educational, and academic purposes.

Table 3: The eight datasets used in our experiments.

subjective statements or conclusion (e.g. "We should be doing...", "This should be done...", etc.) and definitional statements that are not provided to the language model in the context. It is important to note that not all confabulations are bad. If the model were to define standard deviation in the document, this can be a useful piece of information. However, we do not want to rely on the model to present this, especially if these definitions are inadequate or imprecise. Rather, this is an indication that we should seek to present this definition directly to the model.

Operation	**Operation Type**	**Input Arity**	**Input Attribute Types**	**Output Arity**	**Output Attribute Types**
Average	Aggregation	1 ≤ 1	[Arithmetic, Metric] [Grouping]	1	[Arithmetic, Metric]
Correlation	Aggregation	2 ≤ 1	[Arithmetic, Metric, Datetime] [Grouping]	1	[Arithmetic, Metric, Datetime]
Count	Aggregation	1 ≤ 1	[Arithmetic, Metric] [Grouping]	1	[Arithmetic, Metric]
Count Unique	Aggregation	1 ≤ 1	[Arithmetic, Metric] [Grouping]	1	[Arithmetic, Metric]
Get One	Aggregation	1 ≤ 1	[Arithmetic, Metric, Datetime] [Grouping]	1	[Arithmetic, Metric, Datetime]
Max	Aggregation	1 ≤ 1	[Arithmetic, Metric, Datetime] [Grouping]	1	[Arithmetic, Metric, Datetime]
Median	Aggregation	1 ≤ 1	[Arithmetic, Metric, Datetime] [Grouping]	1	[Arithmetic, Metric, Datetime]
Min	Aggregation	1 ≤ 1	[Arithmetic, Metric, Datetime] [Grouping]	1	[Arithmetic, Metric, Datetime]
Standard Deviation	Aggregation	1 ≤ 1	[Arithmetic, Metric] [Grouping]	1	[Arithmetic, Metric]
String Aggregation	Aggregation	1 ≤ 1	[Arithmetic, Metric, Datetime] [Grouping]	1	[Arithmetic, Metric, Datetime]
Sum	Aggregation	1 ≤ 1	[Arithmetic] [Grouping]	1	[Arithmetic, Metric]
And	Boolean	≥ 1	[Filter]	1	[Filter]
Contains	Boolean	1 1	[Attribute] [Metric]	1	[Filter]
Exact	Boolean	2	[Arithmetic, Metric, Categorical, String, Datetime, Identifier]	1	[Filter]
Greater Than	Boolean	2	[Arithmetic, Metric, Datetime]	1	[Filter]
Grtr. Than Equal	Boolean	2	[Arithmetic, Metric, Datetime]	1	[Filter]
Less Than	Boolean	2	[Arithmetic, Metric, Datetime]	1	[Filter]
Less Than Equal	Boolean	2	[Arithmetic, Metric, Datetime]	1	[Filter]
Not	Boolean	1	[Filter]	1	[Filter]
Or	Boolean	≥ 1	[Filter]	1	[Filter]
Absolute Value	Arithmetic	1	[Arithmetic, Metric]	1	[Arithmetic, Metric]
Add	Arithmetic	≥ 2	[Arithmetic, Metric]	1	[Arithmetic, Metric, Datetime]
Divide	Arithmetic	≥ 2	[Arithmetic, Metric]	1	[Arithmetic, Metric, Datetime]
Duration	Arithmetic	1 1	[Datetime] [Datetime]	1	[Arithmetic, Metric]
Multiply	Arithmetic	≥ 2	[Arithmetic, Metric]	1	[Arithmetic, Metric, Datetime]
Percent Change	Arithmetic	2	[Arithmetic, Metric]	1	[Arithmetic, Metric]
Square Root	Arithmetic	1	[Arithmetic, Metric, Datetime]	1	[Arithmetic, Metric, Datetime]
Subtract	Arithmetic	≥ 2	[Arithmetic, Metric, Datetime]	1	[Arithmetic, Metric, Datetime]
Collect	Data Operation	≥ 1	[Attribute]	1	[AttributeCollection]
Groupby	Data Operation	≥ 1	[Categorical, Datetime]	1	[Grouping]
Limit	Data Operation	1	[Attribute]	1	[Limit]
Return	Data Operation	1 ≤ 1 ≤ 1 ≤ 1	[AttributeCollection] [Filter] [Sort] [Limit]	1	[Entity]
Row Number	Data Operation	1	[Sort]	1	[RowNum]
Sort	Data Operation	≥ 1 1	[Attribute] [String]	1	[Sort]
Retrieve Attribute	Retrieval	1 1	[Entity] [String]	1	[Attribute]
Retrieve Entity	Retrieval	1	[String]	1	[Entity]

Table 4: The set of operations currently implemented and used within SQR plans. This includes operations used for retrieval, analysis, filtering, and data transformations.

F Examples of Generated Reports

F.1 Ranking Report across Multiple Domains

We present an example of the domain-agnostic usage of report blueprints. Figure 10 showcases the factual statements generated by the same **Ranking** report blueprint, on three different domains: urban housing, criminal justice, and environmental sustainability. The information requirements of the Ranking report blueprint remain unchanged across domains. The target entity information and the metric are specific to the domain and are provided as inputs to the blueprint. Figure 11 presents the corresponding reports generated by the language model when prompted with the corresponding instruction and factual statements from Figure 10.

F.2 Report from All Generation Modes

We present examples of the same Time over Time report in the Socioeconomics domain generated with various combinations of prompt types and language models. All examples shown in this section are generated for the same report. The target entity is Lake County, IL, the metric is average percent of people in poverty, and the dataset is filtered to only include counties with resident populations greater than 100,000. This is a Time over Time report for the time period ranging from 2010 to 2020.

For each example, we provide the report and the prompt that was used to generate it. For clarity, we have color coordinated the facts provided in the prompt with the corresponding text that was generated as part of the reports to enable quick and direct comparisons between the facts provided to the model and its outputs.

The first report, shown in Figure 12, was generated by SATYRN using GPT-4 and facts provided as natural language statements.

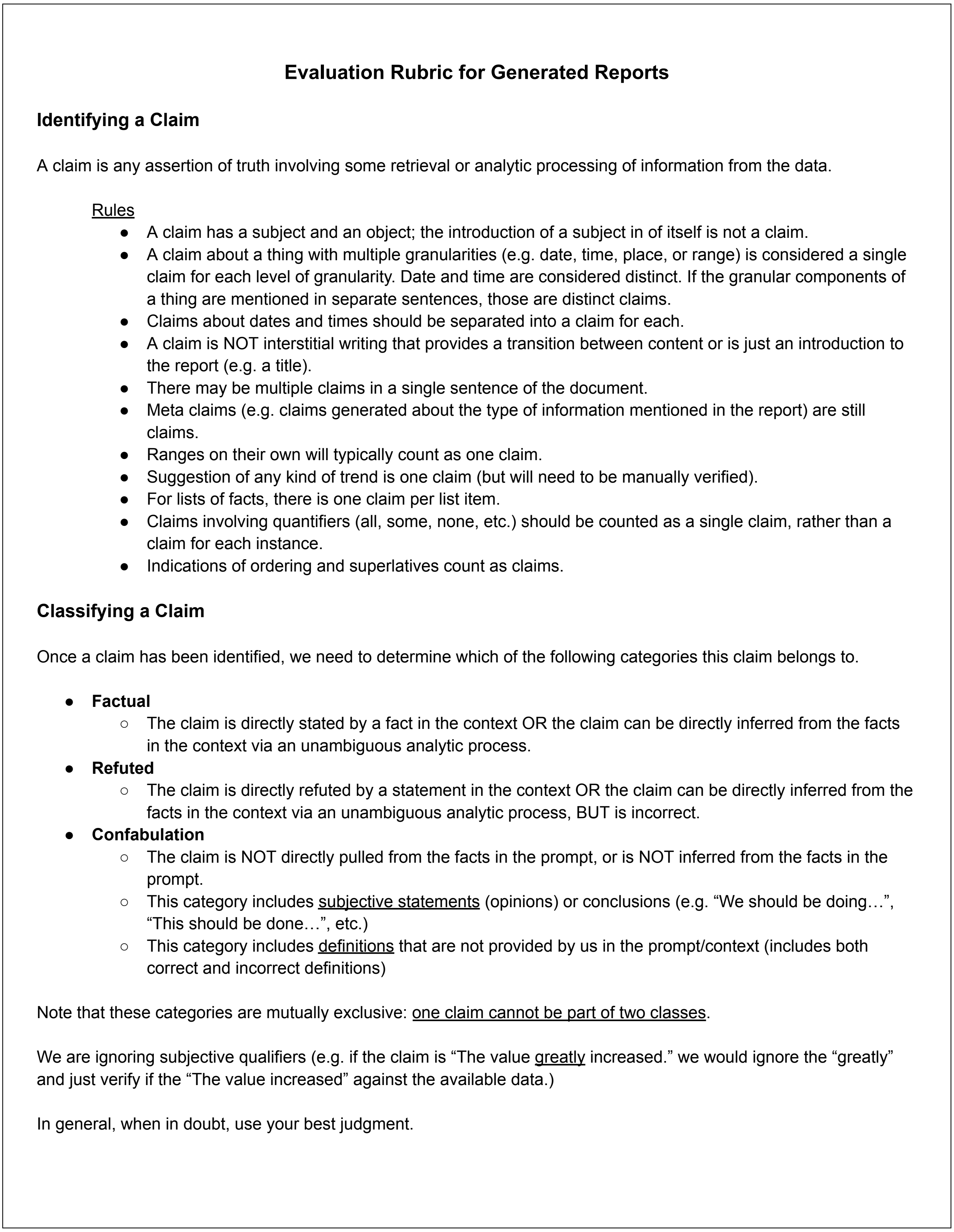
The second report, shown in Figure 13, was generated with the Unaugmented GPT-4 baseline. It was observed that Unaugmented baseline reports had a placeholder for the values which can be used if correct data is provided.

The third report, shown in Figure 14, was generated by Code Interpreter using GPT-4 with access to the data. Notice that in the report, Code Interpreter is able to come up with values for the desired analysis, but is not correct all the time.

The fourth report, shown in Figure 16, was generated by SATYRN using GPT-4 with facts provided as tables rather than natural language statements. The prompt used to generate it is shown in Figure 15.

The fifth report, shown in Figure 17, was generated by SATYRN using Mistral-7B and facts provided as tables rather than natural language statements. The prompt used to generate it is shown in Figure 15. With Mistral-7B, we observed that the table format was not always understood by the model and the language was sometimes confusing. Whereas Mistral-7B performed well in generating reports when prompts included facts provided as natural language statements (Figure 18.

It is worth noting that, in general, the models' generations presented the facts in the same order in which they were provided within the prompt. This property could likely be exploited to improve the structure of the reports by having a planning module determine what information is best seen first. We leave this for future work.



Evaluation Rubric for Generated Reports

Identifying a Claim

A claim is any assertion of truth involving some retrieval or analytic processing of information from the data.

Rules

- A claim has a subject and an object; the introduction of a subject in of itself is not a claim.
- A claim about a thing with multiple granularities (e.g. date, time, place, or range) is considered a single claim for each level of granularity. Date and time are considered distinct. If the granular components of a thing are mentioned in separate sentences, those are distinct claims.
- Claims about dates and times should be separated into a claim for each.
- A claim is NOT interstitial writing that provides a transition between content or is just an introduction to the report (e.g. a title).
- There may be multiple claims in a single sentence of the document.
- Meta claims (e.g. claims generated about the type of information mentioned in the report) are still claims.
- Ranges on their own will typically count as one claim.
- Suggestion of any kind of trend is one claim (but will need to be manually verified).
- For lists of facts, there is one claim per list item.
- Claims involving quantifiers (all, some, none, etc.) should be counted as a single claim, rather than a claim for each instance.
- Indications of ordering and superlatives count as claims.

Classifying a Claim

Once a claim has been identified, we need to determine which of the following categories this claim belongs to.

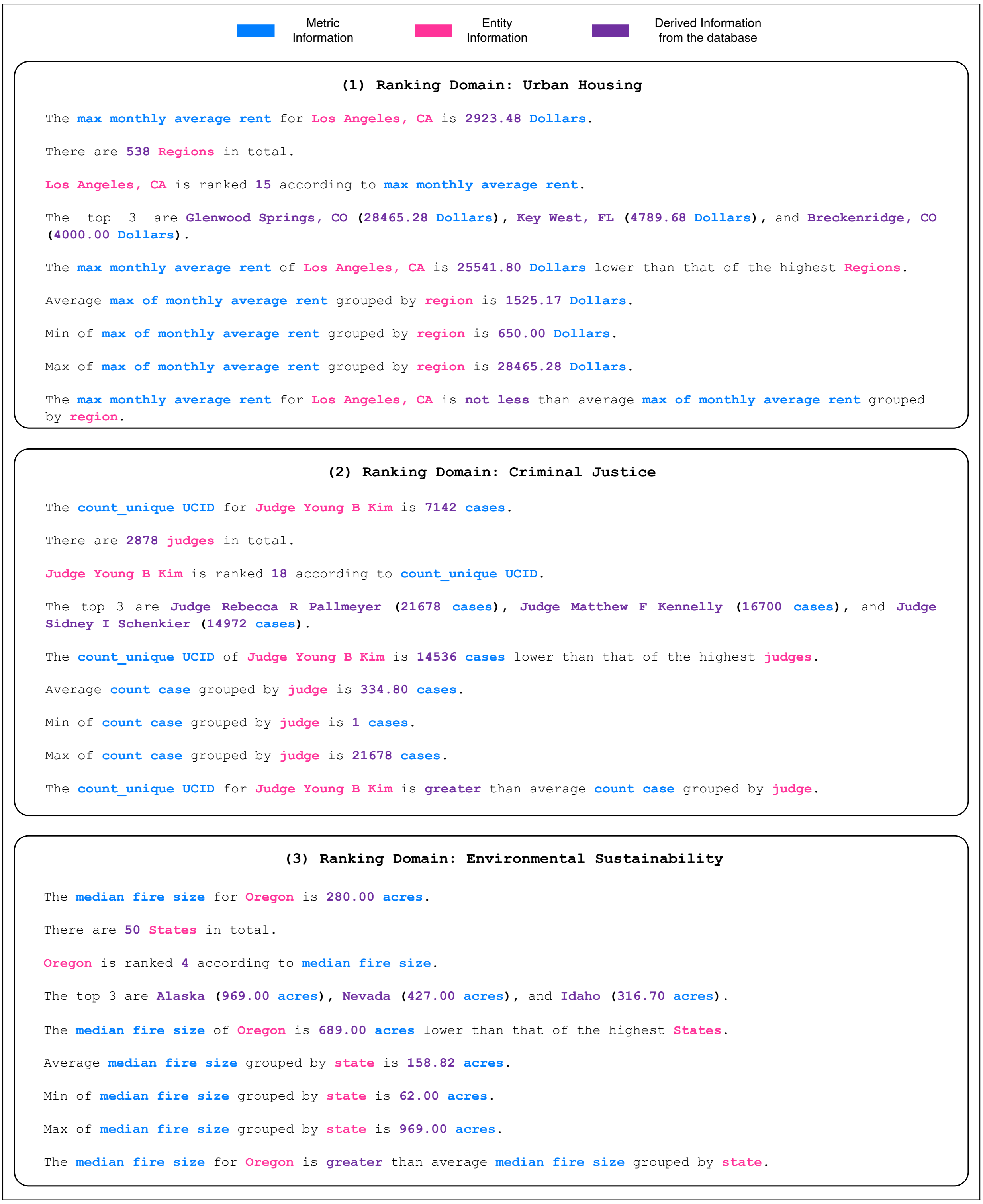
- **Factual**
  - The claim is directly stated by a fact in the context OR the claim can be directly inferred from the facts in the context via an unambiguous analytic process.
- **Refuted**
  - The claim is directly refuted by a statement in the context OR the claim can be directly inferred from the facts in the context via an unambiguous analytic process, BUT is incorrect.
- **Confabulation**
  - The claim is NOT directly pulled from the facts in the prompt, or is NOT inferred from the facts in the prompt.
  - This category includes subjective statements (opinions) or conclusions (e.g. “We should be doing…”, “This should be done…”, etc.)
  - This category includes definitions that are not provided by us in the prompt/context (includes both correct and incorrect definitions)

Note that these categories are mutually exclusive: one claim cannot be part of two classes.

We are ignoring subjective qualifiers (e.g. if the claim is “The value greatly increased.” we would ignore the “greatly” and just verify if the “The value increased” against the available data.)

In general, when in doubt, use your best judgment.

Figure 9: A copy of the rubric that was used for evaluating the reports. This includes both the identification of claims as well as their classification.



🟦 Metric Information 🟥 Entity Information 🟪 Derived Information from the database

(1) Ranking Domain: Urban Housing

The max monthly average rent for Los Angeles, CA is 2923.48 Dollars.

There are 538 Regions in total.

Los Angeles, CA is ranked 15 according to max monthly average rent.

The top 3 are Glenwood Springs, CO (28465.28 Dollars), Key West, FL (4789.68 Dollars), and Breckenridge, CO (4000.00 Dollars).

The max monthly average rent of Los Angeles, CA is 25541.80 Dollars lower than that of the highest Regions.

Average max of monthly average rent grouped by region is 1525.17 Dollars.

Min of max of monthly average rent grouped by region is 650.00 Dollars.

Max of max of monthly average rent grouped by region is 28465.28 Dollars.

The max monthly average rent for Los Angeles, CA is not less than average max of monthly average rent grouped by region.

(2) Ranking Domain: Criminal Justice

The count_unique UCID for Judge Young B Kim is 7142 cases.

There are 2878 judges in total.

Judge Young B Kim is ranked 18 according to count_unique UCID.

The top 3 are Judge Rebecca R Pallmeyer (21678 cases), Judge Matthew F Kennelly (16700 cases), and Judge Sidney I Schenkier (14972 cases).

The count_unique UCID of Judge Young B Kim is 14536 cases lower than that of the highest judges.

Average count case grouped by judge is 334.80 cases.

Min of count case grouped by judge is 1 cases.

Max of count case grouped by judge is 21678 cases.

The count_unique UCID for Judge Young B Kim is greater than average count case grouped by judge.

(3) Ranking Domain: Environmental Sustainability

The median fire size for Oregon is 280.00 acres.

There are 50 States in total.

Oregon is ranked 4 according to median fire size.

The top 3 are Alaska (969.00 acres), Nevada (427.00 acres), and Idaho (316.70 acres).

The median fire size of Oregon is 689.00 acres lower than that of the highest States.

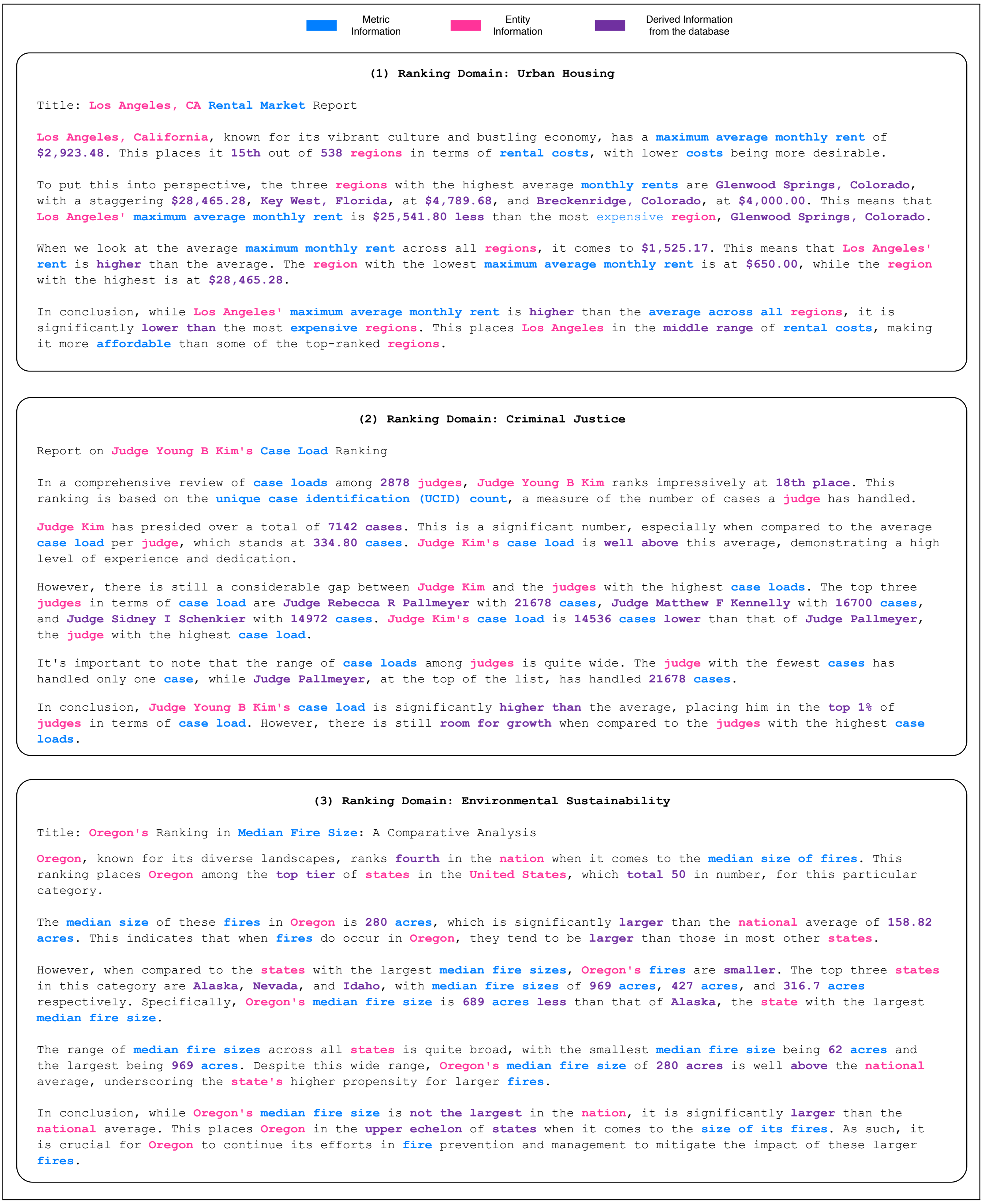
Average median fire size grouped by state is 158.82 acres.

Min of median fire size grouped by state is 62.00 acres.

Max of median fire size grouped by state is 969.00 acres.

The median fire size for Oregon is greater than average median fire size grouped by state.

Figure 10: Factual statements generated across three domains for the same ranking blueprint.


Metric Information
Entity Information
Derived Information from the database

(1) Ranking Domain: Urban Housing

Title: Los Angeles, CA Rental Market Report

Los Angeles, California, known for its vibrant culture and bustling economy, has a maximum average monthly rent of $2,923.48. This places it 15th out of 538 regions in terms of rental costs, with lower costs being more desirable.

To put this into perspective, the three regions with the highest average monthly rents are Glenwood Springs, Colorado, with a staggering $28,465.28, Key West, Florida, at $4,789.68, and Breckenridge, Colorado, at $4,000.00. This means that Los Angeles' maximum average monthly rent is $25,541.80 less than the most expensive region, Glenwood Springs, Colorado.

When we look at the average maximum monthly rent across all regions, it comes to $1,525.17. This means that Los Angeles' rent is higher than the average. The region with the lowest maximum average monthly rent is at $650.00, while the region with the highest is at $28,465.28.

In conclusion, while Los Angeles' maximum average monthly rent is higher than the average across all regions, it is significantly lower than the most expensive regions. This places Los Angeles in the middle range of rental costs, making it more affordable than some of the top-ranked regions.

(2) Ranking Domain: Criminal Justice

Report on Judge Young B Kim's Case Load Ranking

In a comprehensive review of case loads among 2878 judges, Judge Young B Kim ranks impressively at 18th place. This ranking is based on the unique case identification (UCID) count, a measure of the number of cases a judge has handled.

Judge Kim has presided over a total of 7142 cases. This is a significant number, especially when compared to the average case load per judge, which stands at 334.80 cases. Judge Kim's case load is well above this average, demonstrating a high level of experience and dedication.

However, there is still a considerable gap between Judge Kim and the judges with the highest case loads. The top three judges in terms of case load are Judge Rebecca R Pallmeyer with 21678 cases, Judge Matthew F Kennelly with 16700 cases, and Judge Sidney I Schenkier with 14972 cases. Judge Kim's case load is 14536 cases lower than that of Judge Pallmeyer, the judge with the highest case load.

It's important to note that the range of case loads among judges is quite wide. The judge with the fewest cases has handled only one case, while Judge Pallmeyer, at the top of the list, has handled 21678 cases.

In conclusion, Judge Young B Kim's case load is significantly higher than the average, placing him in the top 1% of judges in terms of case load. However, there is still room for growth when compared to the judges with the highest case loads.

(3) Ranking Domain: Environmental Sustainability

Title: Oregon's Ranking in Median Fire Size: A Comparative Analysis

Oregon, known for its diverse landscapes, ranks fourth in the nation when it comes to the median size of fires. This ranking places Oregon among the top tier of states in the United States, which total 50 in number, for this particular category.

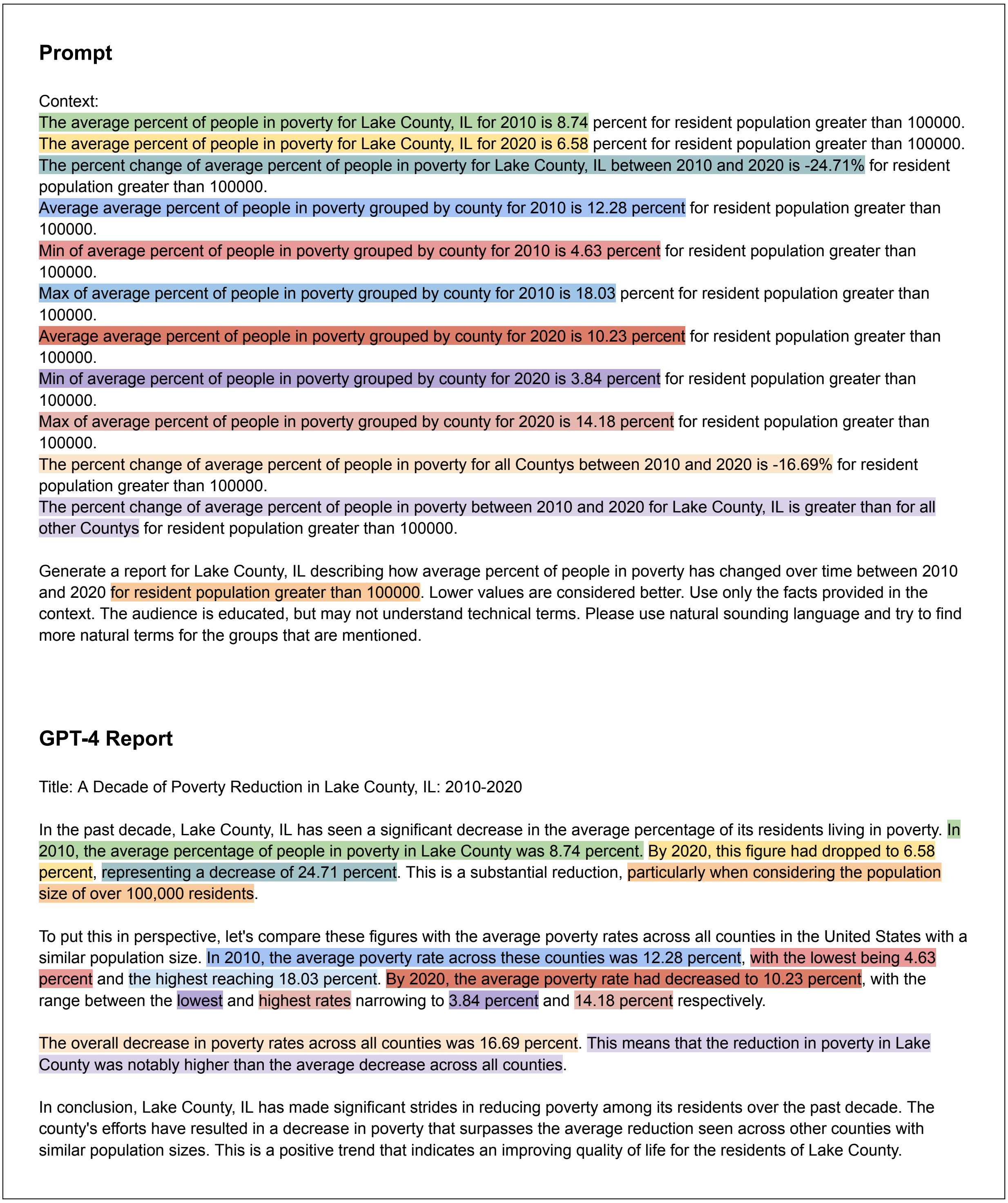
The median size of these fires in Oregon is 280 acres, which is significantly larger than the national average of 158.82 acres. This indicates that when fires do occur in Oregon, they tend to be larger than those in most other states.

However, when compared to the states with the largest median fire sizes, Oregon's fires are smaller. The top three states in this category are Alaska, Nevada, and Idaho, with median fire sizes of 969 acres, 427 acres, and 316.7 acres respectively. Specifically, Oregon's median fire size is 689 acres less than that of Alaska, the state with the largest median fire size.

The range of median fire sizes across all states is quite broad, with the smallest median fire size being 62 acres and the largest being 969 acres. Despite this wide range, Oregon's median fire size of 280 acres is well above the national average, underscoring the state's higher propensity for larger fires.

In conclusion, while Oregon's median fire size is not the largest in the nation, it is significantly larger than the national average. This places Oregon in the upper echelon of states when it comes to the size of its fires. As such, it is crucial for Oregon to continue its efforts in fire prevention and management to mitigate the impact of these larger fires.

Figure 11: Ranking Reports generated across three domains using generalized blueprint.



Prompt

Context:
The average percent of people in poverty for Lake County, IL for 2010 is 8.74 percent for resident population greater than 100000.
The average percent of people in poverty for Lake County, IL for 2020 is 6.58 percent for resident population greater than 100000.
The percent change of average percent of people in poverty for Lake County, IL between 2010 and 2020 is -24.71% for resident population greater than 100000.
Average average percent of people in poverty grouped by county for 2010 is 12.28 percent for resident population greater than 100000.
Min of average percent of people in poverty grouped by county for 2010 is 4.63 percent for resident population greater than 100000.
Max of average percent of people in poverty grouped by county for 2010 is 18.03 percent for resident population greater than 100000.
Average average percent of people in poverty grouped by county for 2020 is 10.23 percent for resident population greater than 100000.
Min of average percent of people in poverty grouped by county for 2020 is 3.84 percent for resident population greater than 100000.
Max of average percent of people in poverty grouped by county for 2020 is 14.18 percent for resident population greater than 100000.
The percent change of average percent of people in poverty for all Countys between 2010 and 2020 is -16.69% for resident population greater than 100000.
The percent change of average percent of people in poverty between 2010 and 2020 for Lake County, IL is greater than for all other Countys for resident population greater than 100000.

Generate a report for Lake County, IL describing how average percent of people in poverty has changed over time between 2010 and 2020 for resident population greater than 100000. Lower values are considered better. Use only the facts provided in the context. The audience is educated, but may not understand technical terms. Please use natural sounding language and try to find more natural terms for the groups that are mentioned.

GPT-4 Report

Title: A Decade of Poverty Reduction in Lake County, IL: 2010-2020

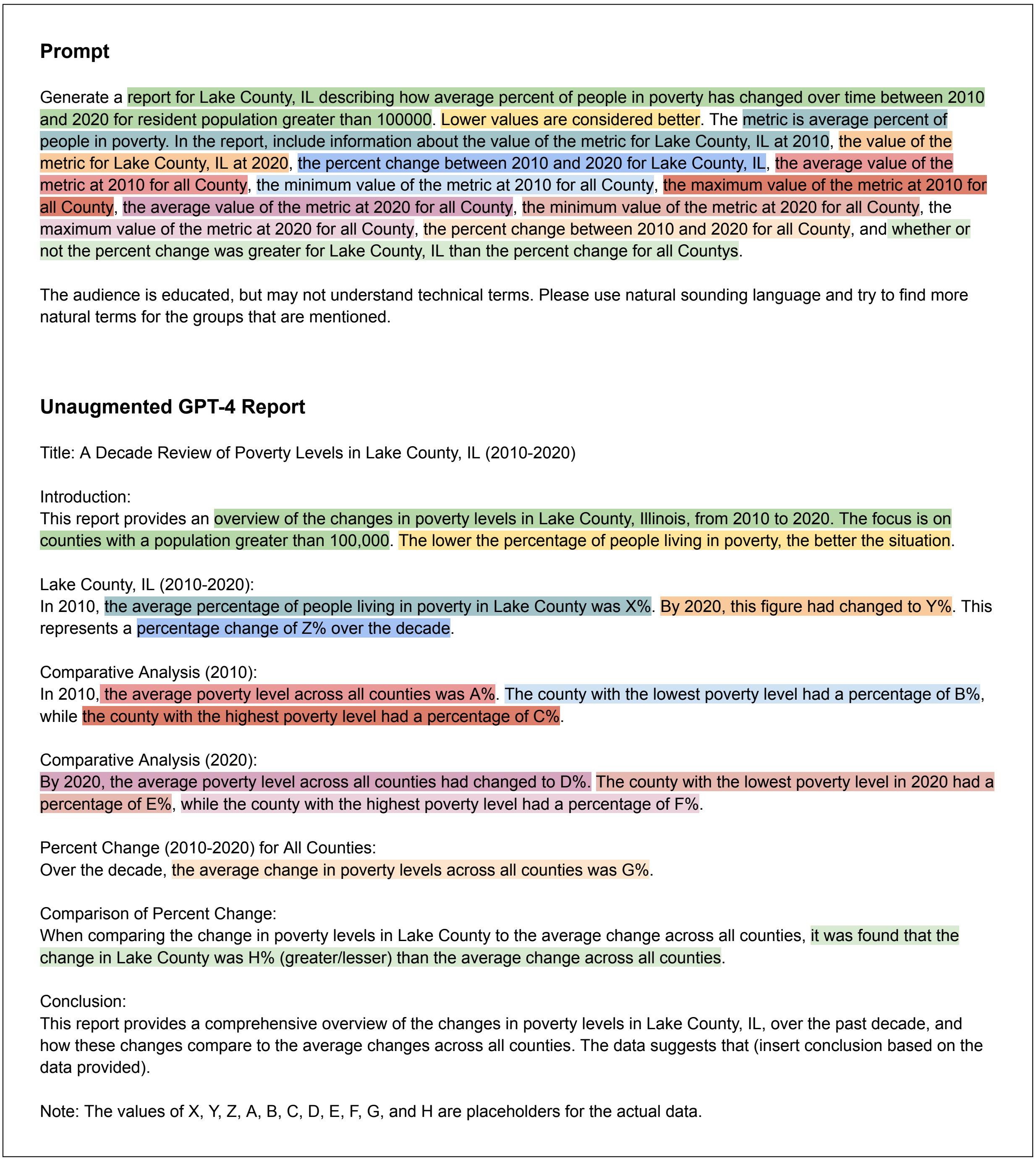
In the past decade, Lake County, IL has seen a significant decrease in the average percentage of its residents living in poverty. In 2010, the average percentage of people in poverty in Lake County was 8.74 percent. By 2020, this figure had dropped to 6.58 percent, representing a decrease of 24.71 percent. This is a substantial reduction, particularly when considering the population size of over 100,000 residents.

To put this in perspective, let's compare these figures with the average poverty rates across all counties in the United States with a similar population size. In 2010, the average poverty rate across these counties was 12.28 percent, with the lowest being 4.63 percent and the highest reaching 18.03 percent. By 2020, the average poverty rate had decreased to 10.23 percent, with the range between the lowest and highest rates narrowing to 3.84 percent and 14.18 percent respectively.

The overall decrease in poverty rates across all counties was 16.69 percent. This means that the reduction in poverty in Lake County was notably higher than the average decrease across all counties.

In conclusion, Lake County, IL has made significant strides in reducing poverty among its residents over the past decade. The county's efforts have resulted in a decrease in poverty that surpasses the average reduction seen across other counties with similar population sizes. This is a positive trend that indicates an improving quality of life for the residents of Lake County.

Figure 12: A Time over Time report generated with SATYRN using GPT-4 and facts provided as natural language statements.



Prompt

Generate a report for Lake County, IL describing how average percent of people in poverty has changed over time between 2010 and 2020 for resident population greater than 100000. Lower values are considered better. The metric is average percent of people in poverty. In the report, include information about the value of the metric for Lake County, IL at 2010, the value of the metric for Lake County, IL at 2020, the percent change between 2010 and 2020 for Lake County, IL, the average value of the metric at 2010 for all County, the minimum value of the metric at 2010 for all County, the maximum value of the metric at 2010 for all County, the average value of the metric at 2020 for all County, the minimum value of the metric at 2020 for all County, the maximum value of the metric at 2020 for all County, the percent change between 2010 and 2020 for all County, and whether or not the percent change was greater for Lake County, IL than the percent change for all Countys.

The audience is educated, but may not understand technical terms. Please use natural sounding language and try to find more natural terms for the groups that are mentioned.

Unaugmented GPT-4 Report

Title: A Decade Review of Poverty Levels in Lake County, IL (2010-2020)

Introduction:
This report provides an overview of the changes in poverty levels in Lake County, Illinois, from 2010 to 2020. The focus is on counties with a population greater than 100,000. The lower the percentage of people living in poverty, the better the situation.

Lake County, IL (2010-2020):
In 2010, the average percentage of people living in poverty in Lake County was X%. By 2020, this figure had changed to Y%. This represents a percentage change of Z% over the decade.

Comparative Analysis (2010):
In 2010, the average poverty level across all counties was A%. The county with the lowest poverty level had a percentage of B%, while the county with the highest poverty level had a percentage of C%.

Comparative Analysis (2020):
By 2020, the average poverty level across all counties had changed to D%. The county with the lowest poverty level in 2020 had a percentage of E%, while the county with the highest poverty level had a percentage of F%.

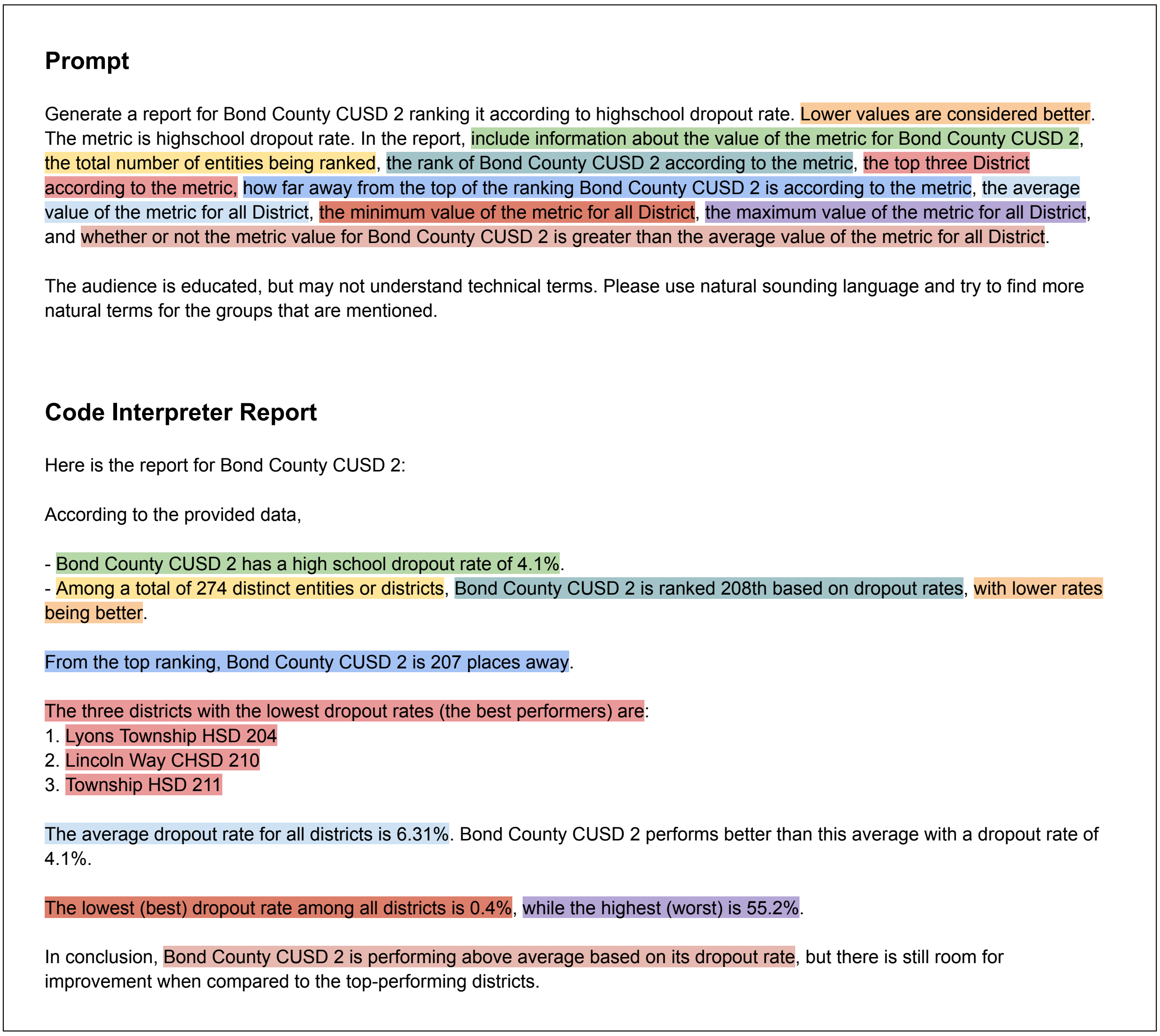
Percent Change (2010-2020) for All Counties:
Over the decade, the average change in poverty levels across all counties was G%.

Comparison of Percent Change:
When comparing the change in poverty levels in Lake County to the average change across all counties, it was found that the change in Lake County was H% (greater/lesser) than the average change across all counties.

Conclusion:
This report provides a comprehensive overview of the changes in poverty levels in Lake County, IL, over the past decade, and how these changes compare to the average changes across all counties. The data suggests that (insert conclusion based on the data provided).

Note: The values of X, Y, Z, A, B, C, D, E, F, G, and H are placeholders for the actual data.

Figure 13: A Time over Time report generated with the Unaugmented GPT-4 baseline. Note that the color coding here is used to highlight where the information should have been if no placeholder values were inserted by the model. This contrasts with the other example outputs which have actual claims with values.



Prompt

Generate a report for Bond County CUSD 2 ranking it according to highschool dropout rate. Lower values are considered better. The metric is highschool dropout rate. In the report, include information about the value of the metric for Bond County CUSD 2, the total number of entities being ranked, the rank of Bond County CUSD 2 according to the metric, the top three District according to the metric, how far away from the top of the ranking Bond County CUSD 2 is according to the metric, the average value of the metric for all District, the minimum value of the metric for all District, the maximum value of the metric for all District, and whether or not the metric value for Bond County CUSD 2 is greater than the average value of the metric for all District.

The audience is educated, but may not understand technical terms. Please use natural sounding language and try to find more natural terms for the groups that are mentioned.

Code Interpreter Report

Here is the report for Bond County CUSD 2:

According to the provided data,

- Bond County CUSD 2 has a high school dropout rate of 4.1%.
- Among a total of 274 distinct entities or districts, Bond County CUSD 2 is ranked 208th based on dropout rates, with lower rates being better.

From the top ranking, Bond County CUSD 2 is 207 places away.

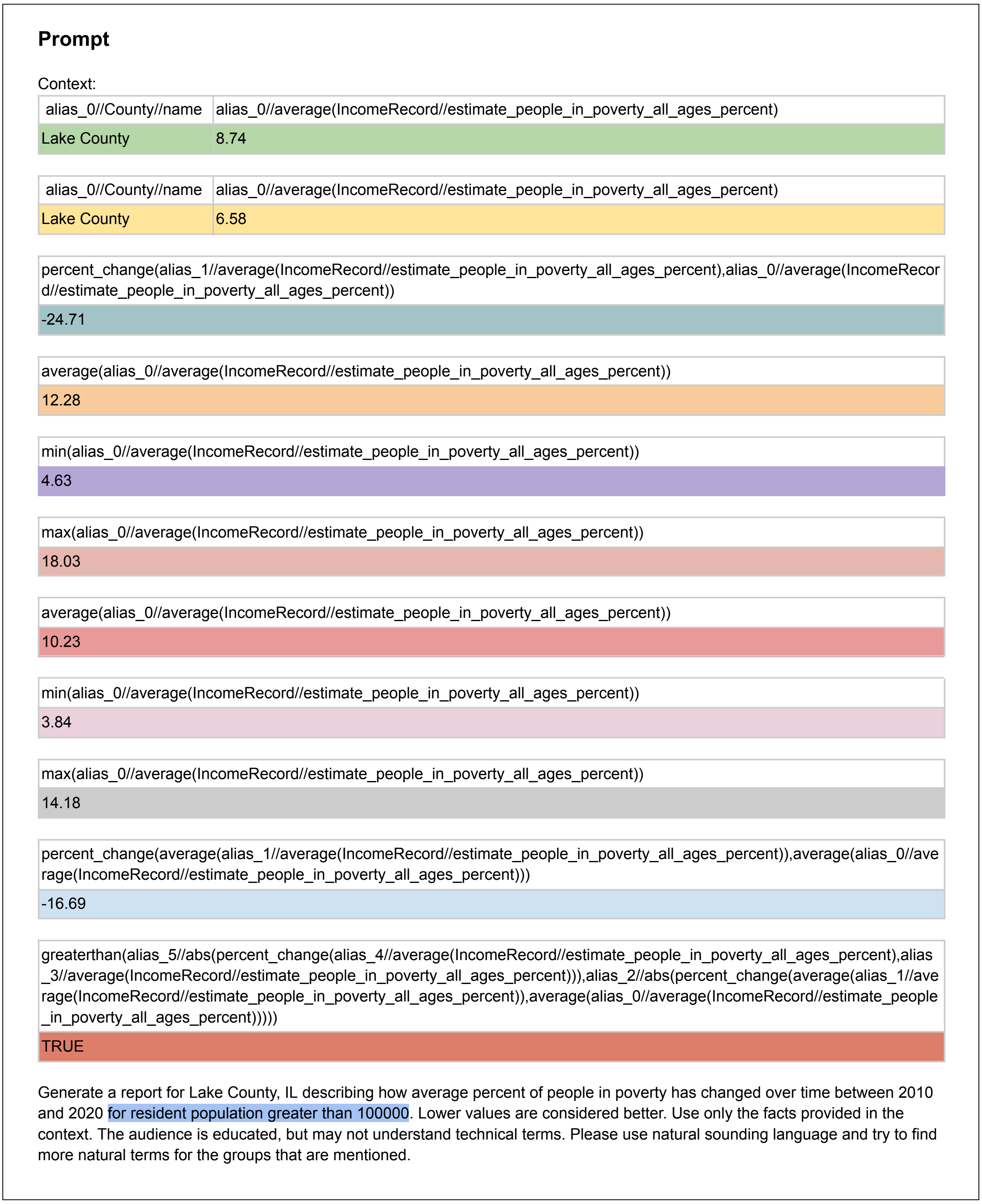
The three districts with the lowest dropout rates (the best performers) are:
1. Lyons Township HSD 204
2. Lincoln Way CHSD 210
3. Township HSD 211

The average dropout rate for all districts is 6.31%. Bond County CUSD 2 performs better than this average with a dropout rate of 4.1%.

The lowest (best) dropout rate among all districts is 0.4%, while the highest (worst) is 55.2%.

In conclusion, Bond County CUSD 2 is performing above average based on its dropout rate, but there is still room for improvement when compared to the top-performing districts.

Figure 14: A Time over Time report generated with GPT-4 Code Interpreter with access to the data



Prompt

Context:

alias_0//County//name	alias_0//average(IncomeRecord//estimate_people_in_poverty_all_ages_percent)
Lake County	8.74

alias_0//County//name	alias_0//average(IncomeRecord//estimate_people_in_poverty_all_ages_percent)
Lake County	6.58

percent_change(alias_1//average(IncomeRecord//estimate_people_in_poverty_all_ages_percent),alias_0//average(IncomeRecord//estimate_people_in_poverty_all_ages_percent))
-24.71

average(alias_0//average(IncomeRecord//estimate_people_in_poverty_all_ages_percent))
12.28

min(alias_0//average(IncomeRecord//estimate_people_in_poverty_all_ages_percent))
4.63

max(alias_0//average(IncomeRecord//estimate_people_in_poverty_all_ages_percent))
18.03

average(alias_0//average(IncomeRecord//estimate_people_in_poverty_all_ages_percent))
10.23

min(alias_0//average(IncomeRecord//estimate_people_in_poverty_all_ages_percent))
3.84

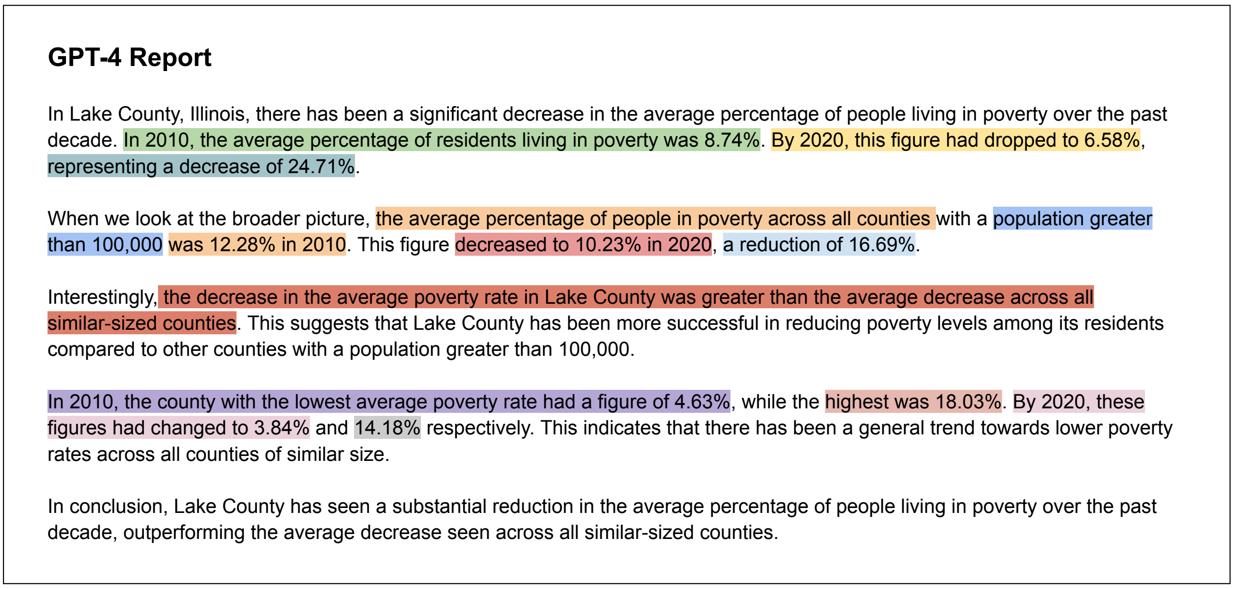
max(alias_0//average(IncomeRecord//estimate_people_in_poverty_all_ages_percent))
14.18

percent_change(average(alias_1//average(IncomeRecord//estimate_people_in_poverty_all_ages_percent)),average(alias_0//average(IncomeRecord//estimate_people_in_poverty_all_ages_percent)))
-16.69

greaterthan(alias_5//abs(percent_change(alias_4//average(IncomeRecord//estimate_people_in_poverty_all_ages_percent),alias_3//average(IncomeRecord//estimate_people_in_poverty_all_ages_percent))),alias_2//abs(percent_change(average(alias_1//average(IncomeRecord//estimate_people_in_poverty_all_ages_percent)),average(alias_0//average(IncomeRecord//estimate_people_in_poverty_all_ages_percent)))))
TRUE

Generate a report for Lake County, IL describing how average percent of people in poverty has changed over time between 2010 and 2020 for resident population greater than 100000. Lower values are considered better. Use only the facts provided in the context. The audience is educated, but may not understand technical terms. Please use natural sounding language and try to find more natural terms for the groups that are mentioned.

Figure 15: A prompt used for generating a Time over Time report with SATYRN where the facts are provided as tables rather than natural language statements.



GPT-4 Report

In Lake County, Illinois, there has been a significant decrease in the average percentage of people living in poverty over the past decade. In 2010, the average percentage of residents living in poverty was 8.74%. By 2020, this figure had dropped to 6.58%, representing a decrease of 24.71%.

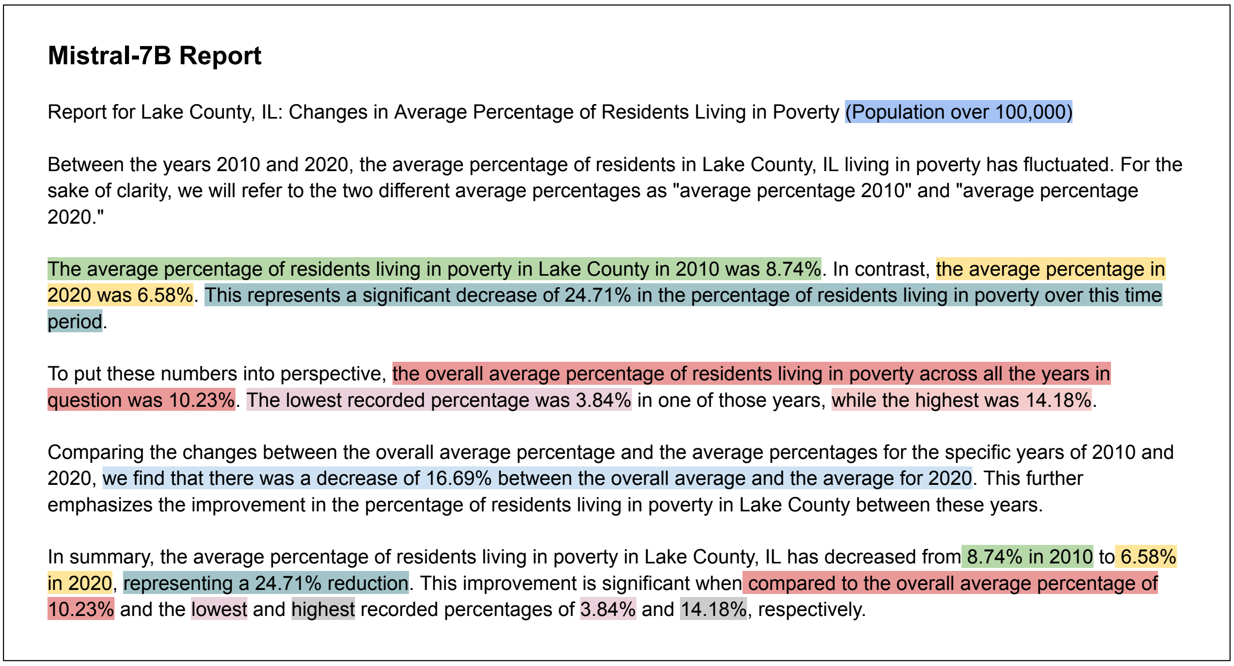
When we look at the broader picture, the average percentage of people in poverty across all counties with a population greater than 100,000 was 12.28% in 2010. This figure decreased to 10.23% in 2020, a reduction of 16.69%.

Interestingly, the decrease in the average poverty rate in Lake County was greater than the average decrease across all similar-sized counties. This suggests that Lake County has been more successful in reducing poverty levels among its residents compared to other counties with a population greater than 100,000.

In 2010, the county with the lowest average poverty rate had a figure of 4.63%, while the highest was 18.03%. By 2020, these figures had changed to 3.84% and 14.18% respectively. This indicates that there has been a general trend towards lower poverty rates across all counties of similar size.

In conclusion, Lake County has seen a substantial reduction in the average percentage of people living in poverty over the past decade, outperforming the average decrease seen across all similar-sized counties.

Figure 16: A Time over Time report generated with SATYRN using GPT-4 and facts provided as tables rather than natural language statements.



Mistral-7B Report

Report for Lake County, IL: Changes in Average Percentage of Residents Living in Poverty (Population over 100,000)

Between the years 2010 and 2020, the average percentage of residents in Lake County, IL living in poverty has fluctuated. For the sake of clarity, we will refer to the two different average percentages as "average percentage 2010" and "average percentage 2020."

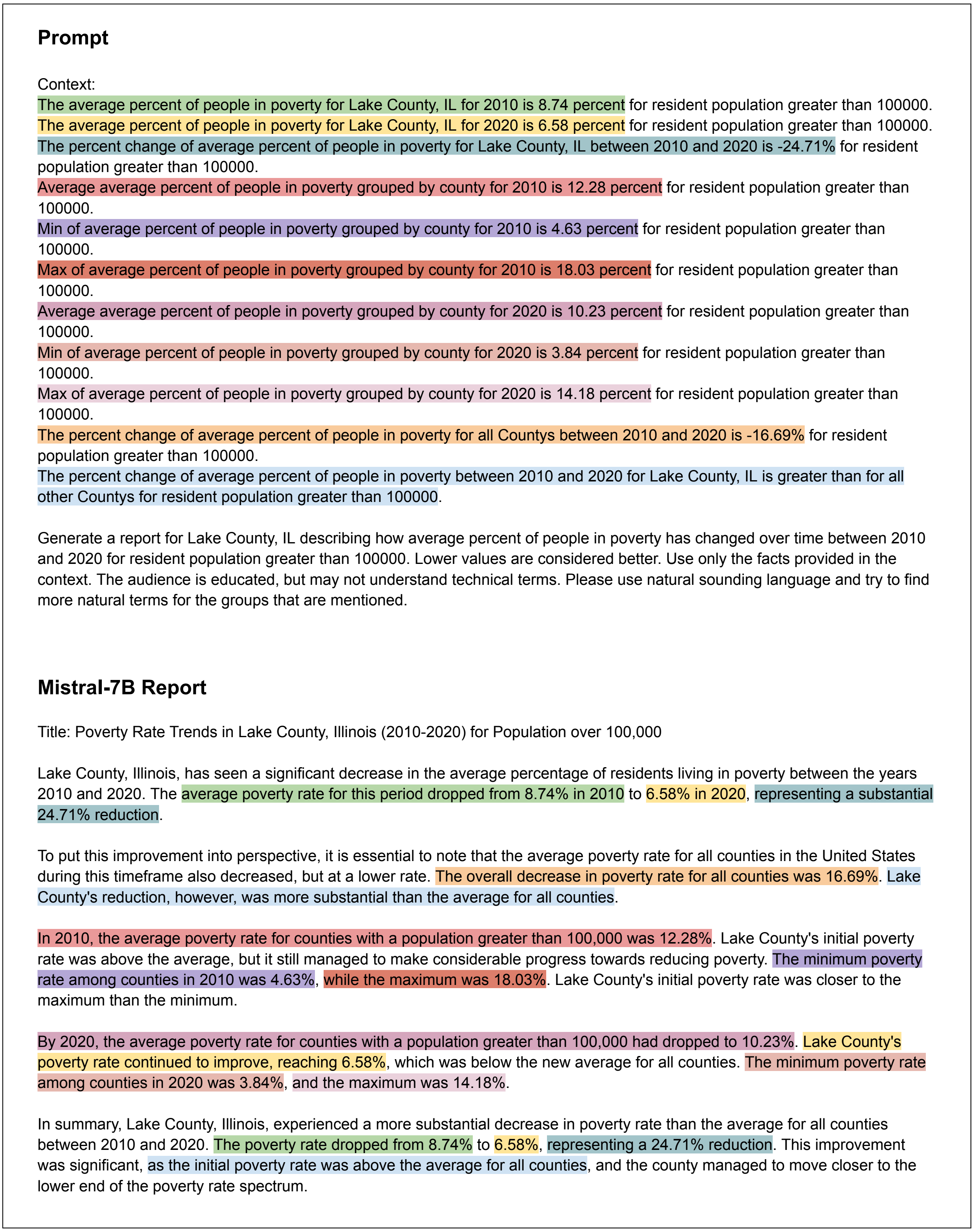
The average percentage of residents living in poverty in Lake County in 2010 was 8.74%. In contrast, the average percentage in 2020 was 6.58%. This represents a significant decrease of 24.71% in the percentage of residents living in poverty over this time period.

To put these numbers into perspective, the overall average percentage of residents living in poverty across all the years in question was 10.23%. The lowest recorded percentage was 3.84% in one of those years, while the highest was 14.18%.

Comparing the changes between the overall average percentage and the average percentages for the specific years of 2010 and 2020, we find that there was a decrease of 16.69% between the overall average and the average for 2020. This further emphasizes the improvement in the percentage of residents living in poverty in Lake County between these years.

In summary, the average percentage of residents living in poverty in Lake County, IL has decreased from 8.74% in 2010 to 6.58% in 2020, representing a 24.71% reduction. This improvement is significant when compared to the overall average percentage of 10.23% and the lowest and highest recorded percentages of 3.84% and 14.18%, respectively.

Figure 17: A Time over Time report generated with SATYRN using Mistral-7B and facts provided as tables rather than natural language statements.



Prompt

Context:
The average percent of people in poverty for Lake County, IL for 2010 is 8.74 percent for resident population greater than 100000.
The average percent of people in poverty for Lake County, IL for 2020 is 6.58 percent for resident population greater than 100000.
The percent change of average percent of people in poverty for Lake County, IL between 2010 and 2020 is -24.71% for resident population greater than 100000.
Average average percent of people in poverty grouped by county for 2010 is 12.28 percent for resident population greater than 100000.
Min of average percent of people in poverty grouped by county for 2010 is 4.63 percent for resident population greater than 100000.
Max of average percent of people in poverty grouped by county for 2010 is 18.03 percent for resident population greater than 100000.
Average average percent of people in poverty grouped by county for 2020 is 10.23 percent for resident population greater than 100000.
Min of average percent of people in poverty grouped by county for 2020 is 3.84 percent for resident population greater than 100000.
Max of average percent of people in poverty grouped by county for 2020 is 14.18 percent for resident population greater than 100000.
The percent change of average percent of people in poverty for all Countys between 2010 and 2020 is -16.69% for resident population greater than 100000.
The percent change of average percent of people in poverty between 2010 and 2020 for Lake County, IL is greater than for all other Countys for resident population greater than 100000.

Generate a report for Lake County, IL describing how average percent of people in poverty has changed over time between 2010 and 2020 for resident population greater than 100000. Lower values are considered better. Use only the facts provided in the context. The audience is educated, but may not understand technical terms. Please use natural sounding language and try to find more natural terms for the groups that are mentioned.

Mistral-7B Report

Title: Poverty Rate Trends in Lake County, Illinois (2010-2020) for Population over 100,000

Lake County, Illinois, has seen a significant decrease in the average percentage of residents living in poverty between the years 2010 and 2020. The average poverty rate for this period dropped from 8.74% in 2010 to 6.58% in 2020, representing a substantial 24.71% reduction.

To put this improvement into perspective, it is essential to note that the average poverty rate for all counties in the United States during this timeframe also decreased, but at a lower rate. The overall decrease in poverty rate for all counties was 16.69%. Lake County's reduction, however, was more substantial than the average for all counties.

In 2010, the average poverty rate for counties with a population greater than 100,000 was 12.28%. Lake County's initial poverty rate was above the average, but it still managed to make considerable progress towards reducing poverty. The minimum poverty rate among counties in 2010 was 4.63%, while the maximum was 18.03%. Lake County's initial poverty rate was closer to the maximum than the minimum.

By 2020, the average poverty rate for counties with a population greater than 100,000 had dropped to 10.23%. Lake County's poverty rate continued to improve, reaching 6.58%, which was below the new average for all counties. The minimum poverty rate among counties in 2020 was 3.84%, and the maximum was 14.18%.

In summary, Lake County, Illinois, experienced a more substantial decrease in poverty rate than the average for all counties between 2010 and 2020. The poverty rate dropped from 8.74% to 6.58%, representing a 24.71% reduction. This improvement was significant, as the initial poverty rate was above the average for all counties, and the county managed to move closer to the lower end of the poverty rate spectrum.

Figure 18: A Time over Time report generated with SATYRN using Mistral-7B and facts provided as natural language statements.