

Analysis Augmented Report Generation

Marko Sterbentz Cameron Barrie Shubham Shahi Abhratanu Dutta
Donna Hooshmand Harper Pack Kristian J. Hammond
Northwestern University

Abstract

Over the past few years, we have seen a tremendous rise in the use of Large Language Models (LLMs). In parallel to this has been the rise in awareness that, while stunningly fluent and responsive, LLMs are often inaccurate. Retrieval Augmented Generation (RAG) has shown itself to be a powerful tool in the direction of language model generation, improving accuracy without sacrificing fluency. There are, however, three challenges that the RAG approach does not address: the use of non-textual data, information that must be inferred, and validation. In this work, we focus on the first two. 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 systems. This Analytics Augmented Generation (AAG) supports the ability to utilize standard analytic techniques to generate facts that are then mapped onto text and passed to an LLM. In this paper, we present a neurosymbolic system, **Satyrn**, that leverages AAG for producing reports based on data and analytics. We demonstrate Satyrn’s flexibility by producing reports in 8 distinct domains. At the core of this work is a lightweight semantic labeling that captures the data requirements of different analytic operations such that any analysis can be tested for applicability against any given data set. This provides the system with the ability to map user queries to analysis, and analysis to data supporting the generation of the facts in natural language that are needed to guide the accurate generation of responses. To evaluate this method, we generated 200 reports consisting of 3319 claims across 3 broadly applicable report types and 8 domains and tested them for accuracy. We find that in terms of factual accuracy of the generated reports, Satyrn’s were on average 30 absolute percentage points higher than a strong GPT-4 Code Interpreter baseline. Satyrn’s superior performance is maintained even when using small, off-the-shelf models.

1 Introduction

Retrieval augmented generation (RAG) methods (Lewis et al., 2021) have seen widespread success in steering language models towards accurate outputs (Shuster et al., 2021). RAG is a popular technique used to provide large language models (LLMs) with information they do not already have in order to have them adhere to that information. However, not all information can be retrieved. For many sources of information, such as knowledge graphs and 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. This Analytics Augmented Generation (AAG) supports the ability to utilize standard analytics techniques to generate facts that are then mapped onto text and used to guide the generation of accurate responses.

AAG requires a way to map a user’s information needs to the underlying source of data. RAG works by having a data source consisting of documents which contain information. The information retrieval process relies on statistical processes based on term frequency (Salton and Buckley, 1988) or dense text embeddings (Karpukhin et al., 2020) which are used to produce a set of documents which are likely to contain an answer. However, these mechanics are not applicable for structured data. Structured data requires determinism rather than likelihood due to the precision required for valid queries and computation.

While RAG is a recent development, producing reports based on data is a long-standing area of research (McKeown, 1985; Reiter and Dale, 1997) with current approaches focusing on end-to-end neural pipelines (Sharma et al., 2022). These methods have a number of shortcomings: 1) they are bespoke, as the models are fit to their training datasets,

2) they are limited to small-scale data tables, which are not representative of real world databases, and 3) they do not perform analytics reliably, instead hoping that a language model can complete any necessary analysis. This last concern has motivated the development of complementary symbolic computation engines, such as in OpenAI’s Code Interpreter¹. Code Interpreter can be used for generating reports. However, such developments remain constrained in the size and scope of data they can accommodate. Further, the most performant among them - Code Interpreter - requires the largest language models available. Even so equipped, it offers no guarantees of reliability as it remains a fundamentally probabilistic system.

In this paper, we present **Satyrn** (Figure 1), a neurosymbolic system which leverages AAG for producing reports based on data and analytics. Satyrn uses lightweight knowledge representations to derive information from data and augment an LLM’s generation to produce reports. These lightweight knowledge representations, called rings, contain knowledge both for mapping data to the operations of the analytics engine as well as for describing the data and their relationships in natural language. To leverage these rings, we develop a language where analysis plans are specified in terms of the entities and attributes defined in the ring rather than the particulars of the underlying data tables and columns. Our analysis engine then executes these analysis plans and expresses the results as simple natural language statements. Reports are specified as a collection of these domain-agnostic plans, the outputs of which – alongside simple instructions – are passed to an LLM for generation. Importantly, this allows for the same type of report to be generated for any dataset which has a ring with no additional work.

Consequently, Satyrn has a number of key strengths. Satyrn provides LLMs access to information derived from data rather than just text based sources. Importantly, this information is guaranteed to be the information required for the target report due to the non-probabilistic nature of its derivation. Additionally, by separating the information derivation process from the language generation, Satyrn can use small, off-the-shelf models to generate high quality reports irrespective of the database size. Finally, Satyrn is scalable and

domain-agnostic. We constrain all domain customization to the ring creation step and keep concerns of domain and analysis completely separate. Expansion of Satyrn’s capabilities can be done by adding new analyses, new datasets, or new report types. Any new analysis will scope across existing datasets with a ring. New datasets with a ring can have existing analyses applied to them and reports generated with them. New report types can be generated for existing datasets with a ring. This all ensures that Satyrn is easily scalable to new domains, new analyses, and new types of reports. Our contributions are as follows:

- First, we propose Satyrn, a scalable, domain agnostic, and neurosymbolic method for producing reports based on data and analytics (Sec. 2).
- Second, we demonstrate how Satyrn can be used for producing reports across 8 varied domains (Sec. 3).
- Third, we provide a thorough analysis of the factuality of generated reports (Sec. 4). We find that reports produced by Satyrn strongly adhere to the data and derived information, and demonstrate that it is possible to use small, off-the-shelf models for generating reports grounded by data.

2 Methods

In this section, we present the lightweight semantic enhancements that allow data to be mapped to analytics and vice versa. We introduce a new plan representation language called Structured Question Representation (SQR) that allows complex analyses to be specified according to the semantic enhancements. We also present an engine that allows SQR plans to be executed over relational databases. Finally, we describe how Satyrn generates reports via the composition and execution of SQR plans.

2.1 Semantics for Data

In order to effectively apply the analytic operations to data, we produce a lightweight mapping called a ring which specifies the entities that comprise the data, their attributes, and the relationships between these entities. Importantly, much of the information contained within is provided directly by the underlying database schema. This makes rings far more simple to specify relative to conventional domain ontologies and the process of creating them is more

¹<https://platform.openai.com/docs/assistants/tools/code-interpreter>

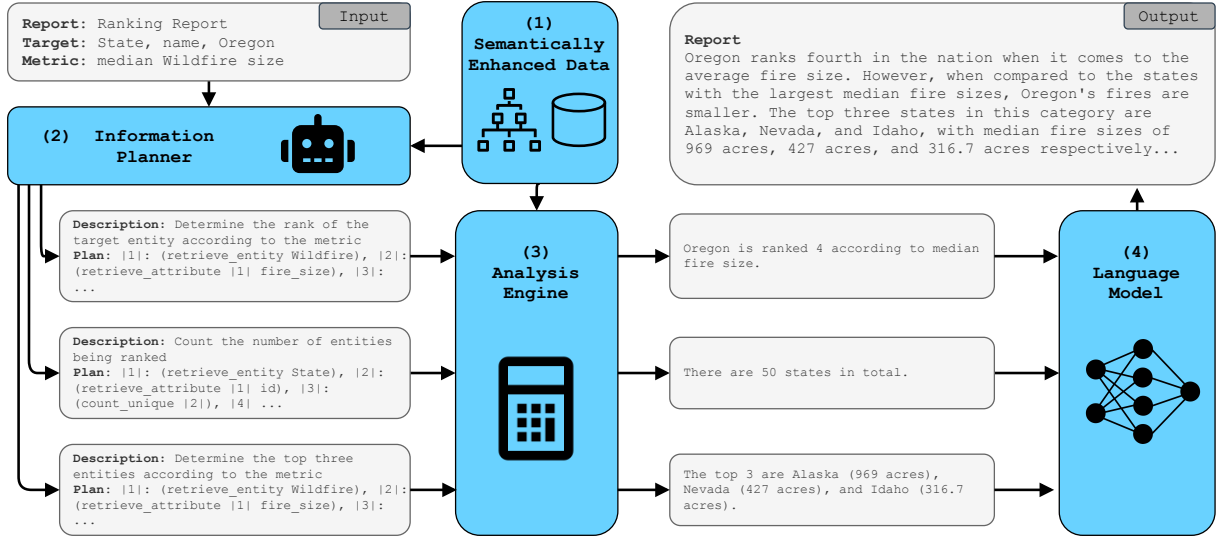


Figure 1: This figure shows the high level approach of Satyrn and its analytics augmented report generation. Beginning with the data in (1), a lightweight knowledge representation called a ring provides simple semantic labels that describe how the data can be mapped to analytic operations. Then, given a report type as input, the information planner (2) determines the information to be derived via analysis of the data and produces a set of executable plans. With the analysis engine (3), each of the plans is executed in order to derive the information. The outputs are formatted as simple statements and used as input to the language model (4) which generates the final report.

similar to tagging than the knowledge engineering that is applied when producing an ontology.

Entities An entity is a concept that groups relevant data together, potentially spanning multiple underlying data tables, into a single unified grouping. Each entity has a uniquely identifying name which is used to refer to it when performing analysis on its attributes. For an example of a complete ring, please see Appendix D.

Attributes Each entity has one or more attributes, each of which maps to a column of one of the underlying database tables comprising the entity. Each attribute has a data type (e.g., integer, float, string), units, and a “nickname” that provides a more descriptive label for the attribute than the column name.

Attribute Types Each attribute also has one or more attribute types used to connect it to the analytic operations that can be applied to it. We define six primary attribute types that provide the necessary information to determine if it makes sense to apply a piece of analysis: 1) **Arithmetic**: numerical values for which mathematical operations make sense, 2) **Categorical**: discrete values typically denoting a class or category, 3) **Datetime**: values designating a date and time, 4) **Document**: values containing free text, 5) **Identifier**: values meant to be used as a unique identifier for an entity, 6) **Metric**: values meant to be used as a measure. These

attribute types are leveraged to constrain the space of possible analytics that can be performed on the entities specified in a ring.

Relationships Relationships define the connection between two entities in the ring. These can be one-to-one, one-to-many, or many-to-many. In the specification of the ring, they are represented as an abstraction of one or more SQL joins. Adding relationships between entities forms a domain graph schema, and each provides relevant metadata (e.g. one Subject can have many Emergency Department Stays). For instance, with a one-to-many relationship, the entity on the many side can be grouped by the entity on the one side. This allows for aggregations to be applied to the entity on the one side.

2.2 Semantics for Analytic Operations

In order to incorporate data into computation, we developed analysis representations of analytic operations, their functionality, and their inputs and outputs. Such a representation defines what analyses can be performed on which data, how those data will be transformed as a result of this analysis, and what analyses can be performed on that result.

Analytic operations take in attributes as inputs and produce a derived attribute as output. Each of the input and output attributes are constrained to have one or more attribute types. Not all an-

analytic operations can be or should be applied to all types of attributes. A key goal of the semantic enhancements is to provide this mapping between analytic operations and the data. A simple way to do it would be to use the type information from the database (float, integer, varchar). However, naively applying an analytic operation, like an average, to every integer or floating point attribute will result in useless information. In contrast, if we define attribute types for the data which indicate what an attribute is at a deeper level, then we can provide the necessary knowledge to an underlying analytic system so that it knows when it can effectively apply a piece of analysis in order to produce meaningful information. For a full listing of the operations, operation type, and their input and output attribute types, please see Appendix C.

2.3 Structured Question Representation (SQR)

Effective use of the ring and analytics requires an expressive and compositional plan representation. Existing representations like SQL are inadequate since the analytics implemented by Satyrn scopes beyond the operations supported within this query language. Additionally, plans should be reusable across datasets. To satisfy these representational needs, we define an analytic plan representation, SQR, which allows for the specification of plans in which the entities and attributes defined in the ring are retrieved and analyzed using the analytic operations currently implemented. SQR abstracts away specific implementation details of the specific underlying query language (e.g., SQL joins), simplifying the syntax and making it agnostic to the data storage format and corresponding query language. This allows underlying implementations to expand to other data sources in the future (e.g., knowledge graphs). A SQR plan example can be seen in Figure 2.

SQR plans are 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 operation whose output is fed to later steps that require the results. Arbitrarily complex plans can be composed to satisfy any information goal that can be described with the available data and analytics. Additional details on the structure of SQR is provided in Appendix E.

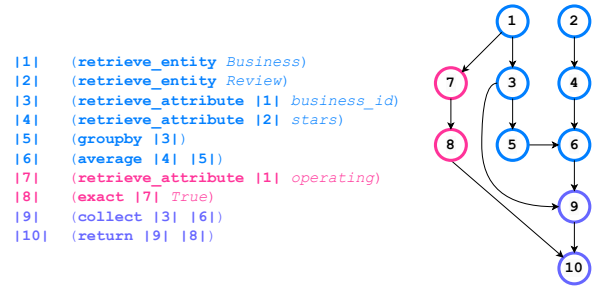


Figure 2: The SQR plan, in textual and graph forms, for determining the average star rating for each business that is operating. Steps 1-4 retrieve the data needed for analysis. Steps 5 and 6 perform the averaging operation for each of the businesses. Steps 7 and 8 produces a filter that ensures the businesses are currently operating. The inputs into Step 9 are the values to output. Step 10 takes in the values to output and any filters to be applied during the execution of the plan.

2.3.1 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. This also allows Satyrn to automatically produce additional attributes by applying analyses to these plans. For example, if "population" is present as an attribute, "average population" would be created since Satyrn knows it can apply the average operation to this attribute. For more details on this attribute augmentation process, please see Appendix G.

2.3.2 Base Plan Templates

Within Satyrn we define a set of analysis plan templates called **base plans**: templated SQR plans that define an analysis to perform, given particular inputs (one or more access plans). These access plans can have filters inserted into them which constrain the retrieval of these data to particular subsets as required. At run time, base plans and access plans are composed together to produce fully executable plans. Figure 3 shows an example of how access plans and base plans are composed. The base plans allow these reports to be specified in a manner where the analyses to be performed are completely separated from the domain, how the underlying data is structured, and how that data is made available for analysis.

2.4 Analytics Engine

Execution of analytic plans requires that they first be converted to a query format that is native to the datasource (e.g., SQL for relational databases) and

then executed to retrieve results. We build an analysis engine that is capable of parsing SQR plans, converting them to valid queries against the underlying data source, and producing the nice names and units for all query results. For more details about the implementation, please see Appendix F

2.5 Report Generation

Satyrn provides a platform for easily defining new types of reports that scope across any domain and dataset. Reports are defined by a set of information requirements that specify the analyses to derive from the data, and constitute the final report. Each requirement maps to a single SQR base plan. We generate the following three report types. A **Ranking** report that ranks an entity’s performance amongst its cohort, **Time over Time** report that compares an entity’s performance at two time periods, and **Comparative Benchmark** report that compares an entity’s performance with a specified benchmark. Report type details can be found in Appendix J.

2.5.1 Generating a Report

For generating each type of report, the inputs required for producing a report are: the entity to write the report about, the metric it should be evaluated upon, any filters that should be applied to the analysis, and any report specific detail such as ranking order, time frame of the analysis, or the benchmark target. Each of the inputs is processed and mapped to the corresponding SQR access plans and filter that will be composed with the set of base plans for the report. The composed SQR plans are executed by the analysis engine to produce the information to include in the report.

2.5.2 Generation of Factual Statements

While the execution of SQL queries produces a set of raw numbers in tabular format, language models generate cleaner output language when instead provided with input in natural language. For instance, the column names of these tables can contain names with abbreviations, underscores, aliases, and acronyms, and the results themselves have no units. Rather than relying on the language model to interpret these column names and infer the appropriate units, we produce simple statements in natural language that provide necessary details for conditioning the generation. We leverage knowledge provided by the semantically enhanced analytic operations, which provides simple language

templates for analytic operations, and the domain-specific ring, which provides descriptive names for entity attributes and units for the attributes.

Specifically, the base plan associated with each information requirement of a report has a natural language statement template associated with it. This statement contains slots which can be filled in so that the resulting statement describes what information that plan provides. For example, the statement template "{EntityReference:0} is ranked {Result:0} according to {Metric:0}." describes the plan that calculates the rank of a particular entity instance amongst others of its type. Some slots, such as {EntityReference:0} and {Metric:0} are filled by ring information (e.g. "Oregon" and "average fire size", respectively), while {Result:0} slots are filled by execution results (e.g. "4th"). Using these values to fill the slot would result in the statement: Oregon is ranked 4th according to average fire size. Numeric suffixes are employed to aid in unique identification when multiple slots of the same type are included in the same template. A list of slots are given in Appendix B.

2.5.3 Prompt Generation

Once a set of factual statements is generated, the system deterministically builds a prompt based on these facts and the type of document to produce. The prompt consists of three key parts. First, a description of the document to generate, typically including the name of the target entity, the metric, any filters that are applied, and any report specific information (e.g., the time frame for the Time over Time report). Second, instructions for the model on how the report should be generated, such as to “use natural sounding language” and information about the intended audience. Third, a set of factual statements that provide the information to convey in the report. The final prompt is then passed to an LLM to generate a report. Refer to Appendix M for examples of prompts used in our experiments.

3 Experiments

In this section, we describe the experiments used to evaluate Satyrn. Reports are generated across 8 domains and evaluated based on their accuracy as measured by their adherence to the information provided to the LLM. We compare reports generated by Satyrn with those generated by two distinct baselines. The first is a setting where a LLM is provided with no access to the data, but is given

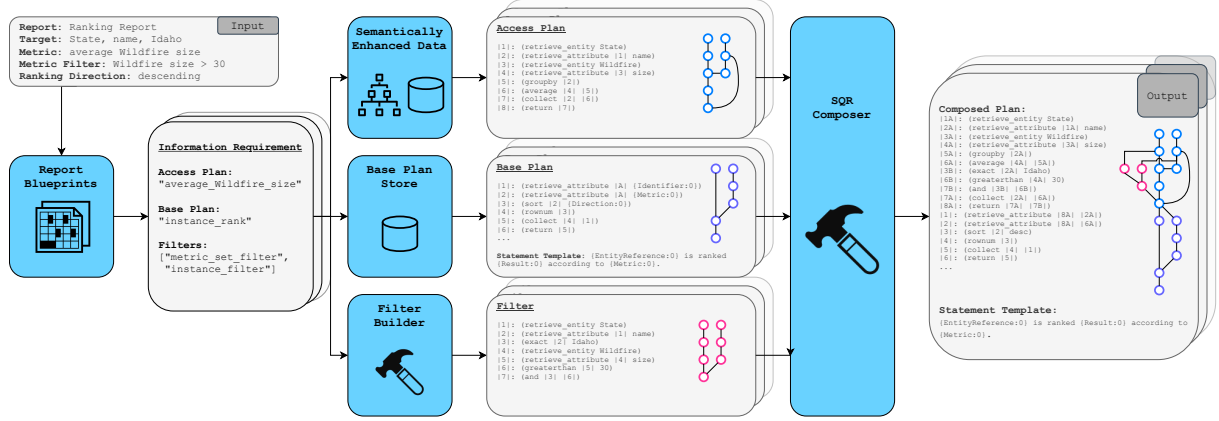


Figure 3: This figure shows the process Satyrn uses to generate a set of executable SQR plans to produce the information required for a report. First, the specified report type and its requirements are retrieved from a set of report blueprints. The base plans associated with each requirement are retrieved from a base plan store. Access plans are retrieved from 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.

detailed instructions on the kind of information to include in its reports. The second baseline is Code Interpreter using GPT-4 with access to the datasets.

3.1 Datasets and Domains

We generate reports based on spanning 8 domains: healthcare, environmental sustainability, urban housing, criminal justice, education, legal and judicial, socioeconomic, and business. A summary of the datasets can be found in Table 1 in Appendix A. We generate a ring for each domain.

3.2 Report Generation Modes

We generate reports using four distinct modes across the three report types listed in Section 2.5.

Satyrn: This generation mode utilizes Satyrn for generating the reports. It uses the factual statement generation detailed in Section 2.5.2 when building the prompt. We use three LLMs for generation: Mistral-7B, Mixtral 8x7B, and GPT-4.

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

Unaugmented GPT-4 (baseline): Our baseline generation mode is prompting a language model to generate a report about a target entity and metric. This basic request is the same as the one used when prompting Satyrn. However, instead of augmenting the generation with information derived via analysis, we instead provide a description of

the information to be included in the report. We use the most capable model, GPT-4, for generating reports.

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. As in the No Augmentation baseline, 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 produce this information and generate a report.

For the Code Interpreter baseline, we give the tool direct access to the same data used by Satyrn when generating the analyses. 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.

For both of these baselines, we do not provide any analysis results in the prompt context, but instead only include a description of each of the types of analyses that should be included in the report.

3.3 Report Evaluation

The key metric for evaluating the generated report is 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 baselines. 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 evaluation process was carried out in two steps: claim identification and claim classification.

3.3.1 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 document and identify the claims being made.

3.3.2 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.

For the full rubric used for evaluation, please refer to Appendix L.

4 Results and Discussion

Our main results are captured in Figure 4. The two baselines (GPT 4 without augmentation and Code Interpreter) had 47% and 58% factual accuracy respectively. Notably, even in the ablation study (the middle group in Figure 4, marked 'Satyrn:table') where just a table with the results are passed to the LLM instead of the natural language facts, the smallest model we tried, Mistral-7B, was comparable to both of the baselines. 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. The smallest model, Mistral-7B, benefits the most with an increase of 36 points from its performance in the ablation. In factual accuracy, Satyrn outperforms

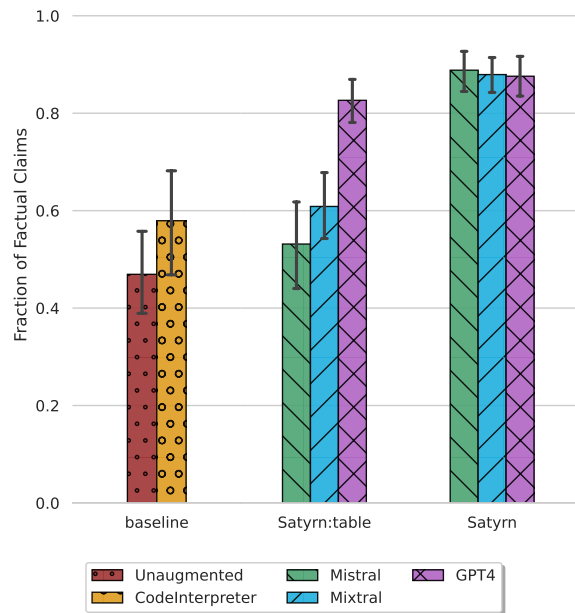


Figure 4: The fraction of claims that were classified as factual, as opposed to either confabulated or refuted for each report generation configuration. Reports generated with Satyrn using statements are consistently higher than all other approaches, even when using small models like Mistral-7B. Error bars are 95% confidence intervals.

the closest baseline by 30 points even with small language models.

In Figure 5, 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. As a result, it 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 run correct queries against the databases resulting in refuted claims. Additionally, reports by unaugmented GPT-4 contained lots of non-claims with placeholders like (e.g. "In comparison with Y, Bond County ranks Z."). This can be further seen in Figure 6 where we can see that unaugmented GPT-4 has the lowest number of claims made on average.

An examination of the accuracy for each generation mode by domain is provided in Appendix H. We also provide an examination of the number of claims in Appendix I.

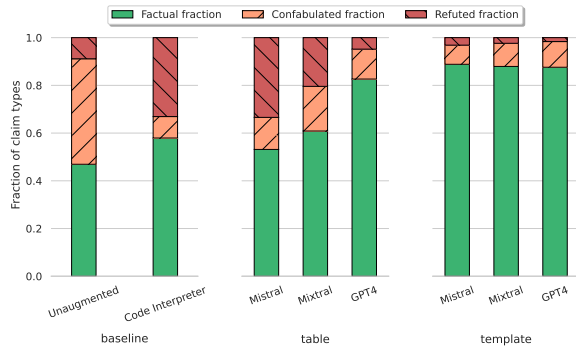


Figure 5: The fraction of claims that were classified as factual, confabulated, or refuted for each report generation configuration.

5 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 Boulous 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 of the key methods to promote factual generation by the language model 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), pre-processed vectors (Verga et al., 2021), search engines (Nakano et al., 2021), and even other language models (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 a language model’s generation (Schick et al., 2023; Zhuang et al., 2023; Peng et al., 2023). While our work also utilizes a symbolic engine, we enhance large scale relational

databases with lightweight semantics in order to effectively map the data to the operations of this engine and obtain results in simple natural language rather than raw results. Our approach also contrasts with the aforementioned approaches in that ours centers around generating long-form reports grounded by data.

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 planning then generation procedure (Su et al., 2021). Currently, the most popular method is to use an 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 from past approaches 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.

6 Conclusion

In this paper, we present a neurosymbolic system, Satyrn, that leverages analytics augmented generation (AAG) to produce factually accurate reports that adhere to information derived from data. This system leverages lightweight knowledge representations in order to map data to analytics and vice versa. Satyrns uses these representations to derive information from data in the form of simple natural language statements. These – along with instructions – are given to a LLM in order to produce reports. We tested this approach by generating 200 reports across 3 types of reports and 8 distinct domains. We found that in terms of factual accuracy of generated reports, Satyrn’s were on average 30 absolute percentage points higher than a strong GPT-4 Code Interpreter baseline. Satyrn’s superior performance is maintained even when using small, off-the-shelf models. In the future, we plan to develop automatic validation methods. Since Satyrn has control over the generation of the initial set of facts, it can make use of these when testing the accuracy of the final output.

Limitations

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.

Our approach has another limitation that it incidentally shares with RAG approaches - namely, validation. Automatic fact extraction and claim verification is an active area of research. Measuring the factual accuracy of the reports generated in our experiments involved us manually labelling each claim as factual, confabulated or refuted. Further research in the direction will greatly help in evaluating systems like ours at a larger scale.

Ethics Statement

In this work we present a system that automates the process of generating documents from data, using a taxonomy of analytic operations to aid in guiding what information gets presented in the report. Of the 8 datasets on which our system was evaluated, 7 of them are publicly available and the other is scheduled to be publicly available in the coming months.

Additionally, while the aim of our system is to ground automatically generated reports in fact, 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 accuracy of the data fed to the system.

For reproducibility purposes, we plan to publicly release our code repository in the near future.

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.

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.

Deepanway Ghosal, Preksha Nema, and Aravindan Raghuvier. 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.

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](#).

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.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. [Dense passage retrieval for open-domain question answering](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Ravikumar 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.
- 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.
- Kathleen R McKeown. 1985. Discourse strategies for generating natural-language text. *Artificial intelligence*, 27(1):1–41.
- Sewon Min, Danqi Chen, Luke Zettlemoyer, and Hananeh Hajishirzi. 2020. [Knowledge guided text retrieval and reading for open domain question answering](#).
- 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.
- Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.
- Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.
- Manuel Salvadores, Paul R Alexander, Mark A Musen, and Natalya F Noy. 2013. Biportal 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, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.
- Mandar Sharma, Ajay Gogineni, and Naren Ramakrishnan. 2022. Innovations in neural data-to-text generation. *arXiv preprint arXiv:2207.12571*.
- 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.
- 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.
- 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/MHIL17031A052NCEN>.

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 memory over 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.

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.

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 Datasets

Table 1 lists the 8 datasets with brief descriptions, in the 8 domains across which we evaluated our system.

Wildfire Occurrence The Wildfire Data (Short, 2022) is a comprehensive collection of information regarding wildfires in the United States spanning from 1992 to 2020. This dataset comprises 2.3 million records that are geographically referenced and represents 180 million acres of land that were consumed by wildfires over the course of 29 years. Additionally, it contains unique identifiers that enable the connection of individual wildfire data points to larger fire perimeter datasets and operational situation reports, providing a holistic view of the wildfire incidents.

MIMIC-IV-ED-Demo The Medical Information Mart for Intensive Care (MIMIC-IV-ED) database (Johnson et al., 2023) is a comprehensive repository of critical care data that includes information from more than 40,000 patients. The data has been deidentified to remove any patient identifiers in compliance with the Health Insurance Portability and Accountability Act (HIPAA) Safe Harbor provision, ensuring patient privacy. The data in MIMIC-IV-ED is derived from patients who were admitted to intensive care units at the Beth Israel Deaconess Medical Center (BIDMC).

Zillow Observed Rent Index The Zillow Observed Rent Index (Zillow Group, Inc., 2023) is a rental price index designed to accurately represent the entire rental housing market, rather than just the properties currently listed for rent. This index assigns dollar values by averaging the rents of homes and apartments that fall within the 40th to 60th percentile range for a specific region. It covers national, metropolitan, county, city, and zip code levels, with appropriate weighting to ensure it reflects the overall rental housing stock.

School Shooting Incidents This dataset (Center for Homeland Defense and Security, 2023) encompasses information regarding shooting incidents that have occurred from the start of 1970 up to June 2022. These incidents encompass a broad range of situations, including instances where firearms were displayed, discharged, or where bullets struck school property, irrespective of the number of casualties, the time of day, or the day of the week.

Illinois Report Card This dataset (Illinois State Board of Education, 2022) is an annual report published by the Illinois State Board of Education. It

provides information about the progress of the state, individual schools, and districts in achieving various educational objectives, as well as school funding information at the state and federal level. The report offers a comprehensive overview of student and school performance to assist families and communities in understanding and aiding their local schools.

SCALES SCALES-OKN (Paley et al., 2021) utilizes two primary datasets: PACER, which serves as the official source for electronic federal judicial records, and the Federal Judicial Center’s (FJC) database containing information about federally appointed judges. The SCALES-OKN dataset incorporates specific docket reports from PACER, encompassing ten years of docket data from Northern Illinois district courts between 2007 and 2016, as well as docket reports from all district courts for the year 2016. Additionally, it includes a variety of judge-related metadata, such as birthdate, gender, race/ethnicity, appointment history, appointing parties, educational background, and professional career details.

Income Disparity The Personal Income by County, Metro, and Other Areas report by the U.S. Bureau of Economic Analysis (U.S. Bureau of Economic Analysis, 2022; U.S. Bureau of Labor Statistics, 2023; U.S. Census Bureau, 2022b,a,c) provides information about the earnings of individuals residing in specific geographic regions, such as counties and metropolitan areas. This data reflects the income received by these residents or on their behalf, offering estimates based on their place of residence.

Yelp Open Dataset The Yelp Open Dataset (Yelp Inc., 2023) is a collection of Yelp’s data, including reviews, information about businesses, and user data. It’s available for personal, educational, and academic use. This dataset was compiled by Yelp and at the time of collection included 5,996,996 reviews, details about 188,593 businesses, and 280,992 pictures from the Yelp platform.

B Statement Template Slot Types

The following is a list of slot types that are present in the statement templates associated with the base plans.

- **Identifier** - name of the attribute that uniquely identifies the corresponding entity

- **Metric** - name of an attribute that serves as a metric for that entity
- **Filter** - natural language expression of the filter used in the analysis plan
- **EntityReference** - natural language expression uniquely identifying the entity instance
- **Quantity** - generic slot for numeric values
- **String** - generic slot for text
- **Result** - a result of the plan execution
- **Unit** - a unit corresponding to a result from the plan

C Analytic Operations

A listing of the operations currently available within the analytics taxonomy can be seen in Table 2. This taxonomy was designed with extensibility in mind, and currently implements most major SQL operations, as well as some operations not supported defined in SQL.

D Ring Example

An example ring for the environmental sustainability can be found in Figure 6. Here, two entities (State and Wildfire) are defined, each comprising a single corresponding table. A single relationship, defined by the join that connects the underlying tables, links those two entities.

Rings are divided into three main sections: metadata about the ring (shown in purple), data source schema information (shown in pink), and the data abstraction layer definition (shown in blue). The metadata includes basic information such as the name of the ring, a brief description, and the ID. The data source schema information defines the tables and joins in the database. The data abstraction layer definition defines the *entities* and *relationships* between them. Entities are defined in terms of one or more tables in the database, and their attributes are defined in terms of one or more columns from the entity’s tables. Relationships are defined in terms of one or more joins.

Note that construction of a ring is semi-automated by having a user fill in the blanks (e.g. entities, attributes, relationships, joins, etc.) in a pre-annotated CSV file and then passing this to a script that converts this into the final JSON configuration format that actually gets ingested by Satyrn

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 1: Datasets used for the evaluation which specify the underlying data used for each of the 8 domains we tested with.

upon startup. We are currently working to make this process more streamlined via an interactive interface, which is left as future work.

E SQR Structure

Leveraging the analytics semantics allows this representation to support standard operations in the underlying query language (e.g., SQL), such as retrieving, aggregating, grouping, filtering, sorting, etc. Entities and attributes described by the ring can be retrieved using the *retrieve_entity* and *retrieve_attribute* operations. Analytic operations take these attributes as input and produce derived attributes (i.e., ones which are not present in the ring), which can be used as input to subsequent operations. In this way, arbitrarily complex analysis can be composed by chaining operations together. Data manipulation operations such as sort and limit are also supported, as are filtering operations like comparison and boolean operators.

Both entity attributes and derived attributes can be passed to the collect operation in order to stage them for output in the final results. The return operation takes in the attributes to be collected, along with any sort, filter, and limit operations that were

specified. Each return operation denotes the end of a plan and results in a structure which is analogous to a single SQL query. Subsequent queries can retrieve attributes collected by prior queries. In this way, subplans, which are analogous to nested queries or “subqueries” found in SQL, can be represented.

F Analytics Engine Implementation

F.1 Plan Parsing and Execution

Conversion of the graph-structured analytics plan into a query language 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 2, the attributes *business_id* and *operating* are retrieved from the Business entity while *stars* is retrieved from the Review entity. The average operation is applied to *stars*, grouped by *business_id*. A filter is constructed such that each Business considered should have an operating value of True. From this information, the query is constructed using a

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 1 ≤ 1 ≤ 1	[Attribute] [AttributeCollection] [Filter] [Sort] [Limit]	1 1 1 1	[Limit] [Entity]
Return	Data Operation	1 ≤ 1 ≤ 1 ≤ 1	[AttributeCollection] [Filter] [Sort] [Limit]	1 1 1 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 2: The set of all operations implemented as part of the analytics taxonomy. This includes operations used for retrieval, analysis, filtering, and data transformations.

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.

F.2 SQL Object Relational Mapping

While the system’s analytics engine is designed to be extendable to a variety of data source types, it is currently only configured to execute queries

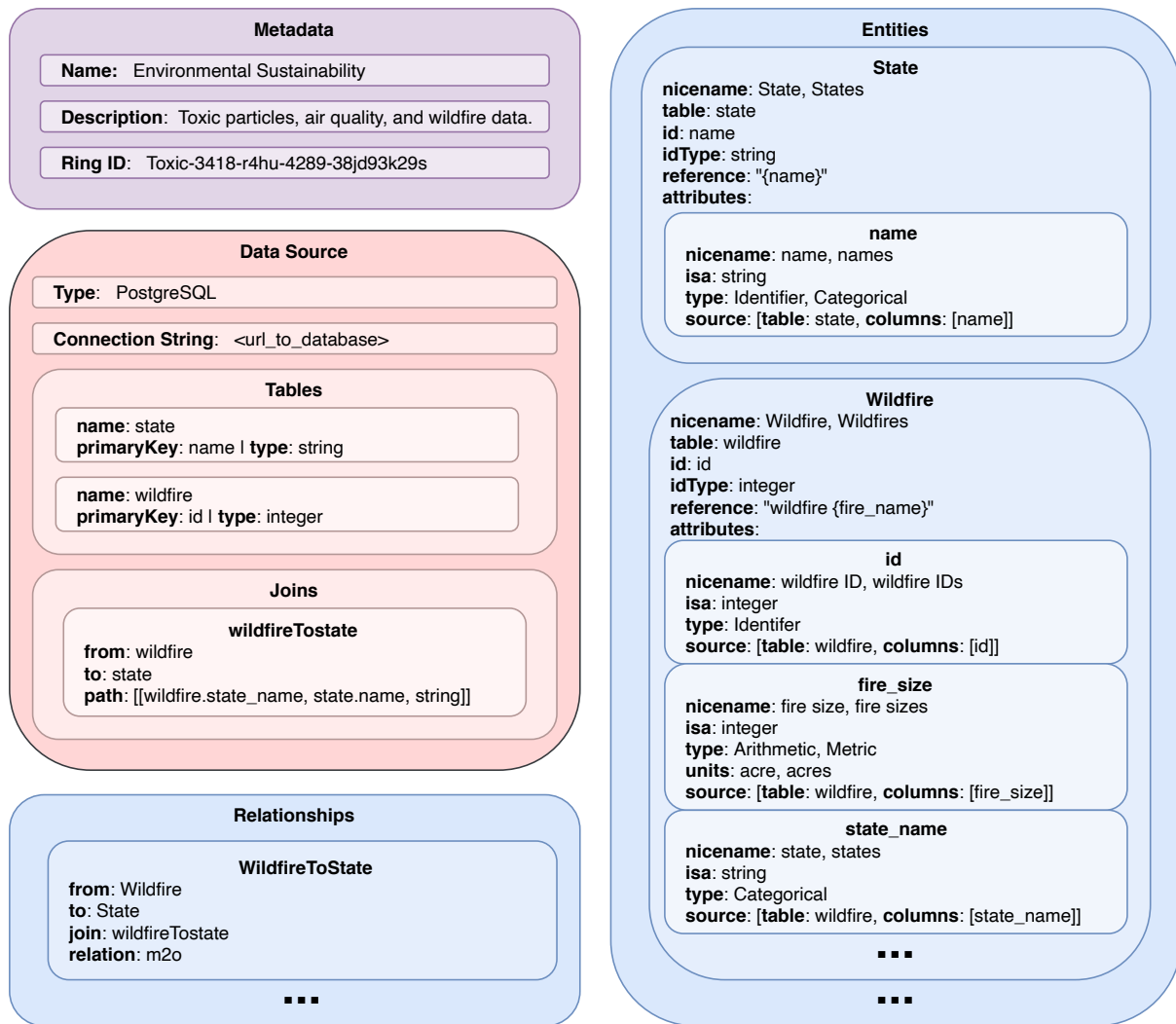


Figure 6: Ring for the Environmental Sustainability domain. This is a truncated example showing just two of the entities that are defined within this domain: State and Wildfire. Metadata about the ring is shown in purple, the data source schema information is shown in pink, and the data abstraction layer is shown in blue.

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, etc.

Implicit Joins A major benefit provided by the abstractions of the ring is that, for a given domain, relationships 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.

First, 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, all joins between each pair of entities in a given plan are identified by collecting joins along the shortest path of relationship links between those entities.

G Attribute Augmentation

In addition to the attributes explicitly defined by the ring, it is possible to automatically derive an additional set of useful attributes and make these

available for analysis. 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. This includes attributes that result from aggregating values associated with a related entity.

For example, assume there exists a ring with entities State and Wildfire where there is a one-to-many relationship between these two entities, State has an attribute called name with attribute type Identifier, and Wildfire has an attribute called *fire_size* with attribute type Arithmetic. Based on this knowledge, the system knows it is possible to apply aggregations such as an average to this attribute since it requires an Arithmetic attribute as input. The system also knows it could group this aggregation by State name since it's an Identifier attribute. As a result, by using the knowledge provided by the ring, the system can automatically produce a derived attribute called *average_wildfire_fire_size* wherein the averaging aggregation is grouped by the Identifier of the State entity. This derived attribute would be an attribute of the State entity and would have an associated access plan that specifies how to produce this information in the form of a brief analytic plan. Storing only the access plan allows the system to have access to this information whenever it is required, without requiring the underlying database to compute and store these values in its tables explicitly. Additional properties of the derived attribute, such as nickname and type, are produced based on the original attributes and the analysis that was applied.

H Report Accuracy by Domain

Figure 7 shows the fraction of claims which are factual, confabulated, and refuted for reports generated by Satyrn using natural language statements for each of the 8 domains. We find that Satyrn has a high degree of accuracy across all domains.

I Examination of the Number of Claims

Focusing on the table-based generation study in Figure 8 we see that the smallest LLM, Mistral-7B, has the highest fraction of refuted claims made, indicating the difficulties of small LLMs in parsing non-textual data. Once we generate the factual statements however, Mistral-7B's fraction of refuted claims drop drastically and becomes com-

parable to those of much larger models. This further pinpoints the benefits of Satyrn in augmenting small, locally runnable LLMs and enabling them to disseminate information derived from data truthfully.

J Report Types

We focus on three common reporting use cases and present their information requirements below.

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, and whether or not the target entity is greater than the average value of the metric across 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, and whether or not the percent change was greater for the target entity instance or the average for 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, and what the standard deviation of the metric is for all entities.

K Report Generation Details

K.1 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.

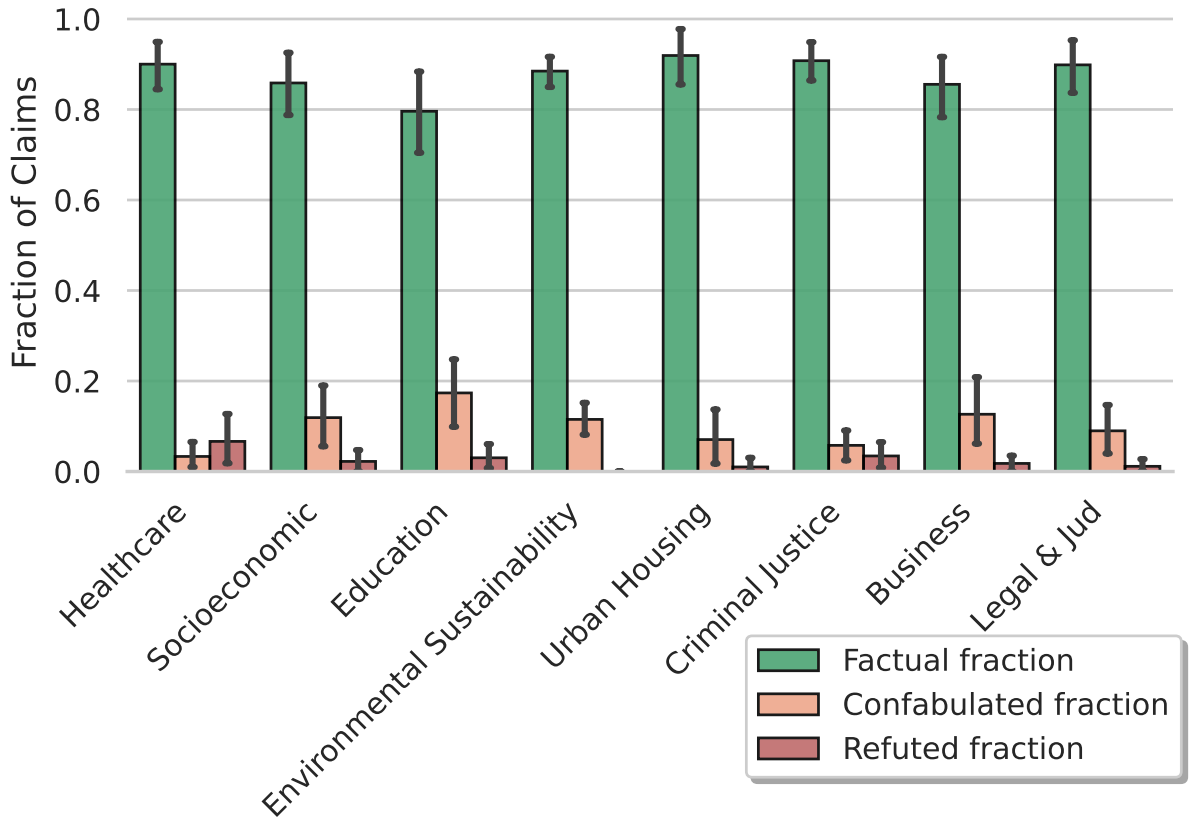


Figure 7: The fraction of claims which are factual, confabulated, and refuted for each of the 8 domains for Satyrn using natural language statements. These results include reports generated with all three LLMs: Mistral-7B, Mixtral 8x7B, and GPT-4.

K.2 Notes

Please note that many code interpreter reports needed to be generated several times before producing a report

L 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 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 indica-

tion that we should seek to present this definition directly to the model.

M Examples of Generated Documents

In this section, 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 100000. 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 we provided to the model and its outputs.

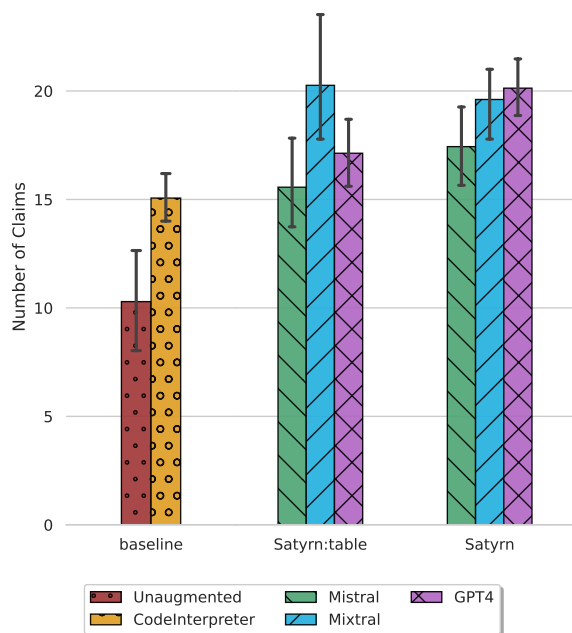


Figure 8: The average number of claims made for each configuration across all domains.

The first report, shown in Figure 10, was generated by Satyrn using GPT-4 and facts provided as natural language statements.

The second report, shown in Figure 11, was generated by GPT-4 under the No Augmentation Baseline condition. It was observed that No augmentation baseline reports had a placeholder for the values which can be used if correct data is provided.

The third report, shown in Figure 12, was generated by Code Interpreter using GPT-4 with access to the data. Notice in the report, that code interpreter is able to come up with values for the desired analysis but is not correct all the time. Report shared here with all the claims made, most of the claims were deemed as refuted when verified with the data.

The fourth report, shown in Figure 14, was generated by Satyrn using GPT-4 and facts provided as tables rather than natural language statements. GPT4 was able to perceive the table format the best out of the three models that we used. The analysis was well placed and the language was fluent. The prompt used to generate it is shown in Figure 13.

The fifth report, shown in Figure 15, was generated by Satyrn using Mistral-7B and facts provided as tables rather than natural language statements. With Mistral-7B, we observed that the table format was not always understood by the model and the language was sometimes confusing. Whereas

Mistral performed well in generating reports when prompts were given in a template format. For the two Mistral reports given observe that the language is comparatively better for the template report than the table report. The prompt used to generate it is shown in Figure 13.

The sixth report, shown in Figure 16, was generated by Satyrn using Mistral-7B and facts provided as natural language statements.

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.

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 10: 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.

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/less) 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 11: A Time over Time report generated with GPT-4 under the No Augmentation Baseline condition.

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.

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 12: A Time over Time report generated with Code Interpreter using GPT-4 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 13: 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 14: 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 15: 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 16: A Time over Time report generated with Satyrn using Mistral-7B and facts provided as natural language statements.