ACQUISITION INNOVATION
RESEARCH CENTER

Bot Automation Using Large Language Models (LLMs) and Plugins

EXECUTIVE SUMMARY AND REPORT
JULY 2024

PROFESSOR
Naren Ramakrishnan, *Virginia Tech*

SENIOR RESEARCH ASSOCIATE
Patrick Butler, *Virginia Tech*

RESEARCH SCIENTIST
Brian Mayer, *Virginia Tech*

STUDENT RESEARCH ASSISTANT
Andrew Neeser, *Virginia Tech*

SPONSOR
Mr. Timothy Applegate, SES, Director, Acquisition Management and Integration Center (AMIC), Air Combat Command (ACC), U.S. Air Force

DISCLAIMER

The Acquisition Innovation Research Center (AIRC) is a multi-university partnership led and managed by the Stevens Institute of Technology and sponsored by the U.S. Department of Defense (DoD) through the Systems Engineering Research Center (SERC)—a DoD University-Affiliated Research Center (UARC).

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Office of the Under Secretary of Defense for Acquisition and Sustainment (OUSD(A&S)) and the Office of the Under Secretary of Defense for Research and Engineering (OUSD(R&E)) under Contract HQ0034-19-D-0003, TO#0285.

SYSTEMS ENGINEERING RESEARCH CENTER

AIRC ACQUISITION INNOVATION RESEARCH CENTER

STEVENS INSTITUTE OF TECHNOLOGY 1870

TABLE OF CONTENTS

LIST OF FIGURES

RESEARCH TEAM

Name	Organization	Labor Category
Naren Ramakrishnan	Virginia Tech	Professor
Patrick Butler	Virginia Tech	Senior Research Associate
Brian Mayer	Virginia Tech	Research Scientist
Andrew Neeser	Virginia Tech	Student Research Assistant

ACRONYMS AND ABBREVIATIONS

AI	Artificial Intelligence
AMIC	Acquisition Management and Integration Center
COR	Contracting Officer's Representative
CPARS	Contractor Performance Assessment Reporting System
DoD	Department of Defense
GUI	Graphical User Interface
LLM	Large Language Model
ML	Machine Learning

EXECUTIVE SUMMARY

The aim of this research study was to create tools that automate information extraction pipelines to support business processes in contract and procurement management. The research team was specifically asked to explore opportunities to use Large Language Models (LLMs) to accomplish this task. After reviewing the problem space and the potential solutions, the team designed and created a tool to generate reports on the status of entries from the Contractor Performance Assessment Reporting System (CPARS), broken down by contracting division. This tool automates the extraction of the Contracting Officer's Representative (COR) status information.

The team also explored methods for using LLM pipelines to automate other potential contractual management tasks and presented some demonstrations of possible uses. The research indicated that LLMs have significant potential to enhance contract and procurement management processes, e.g., automating field extraction from existing contracts, assisting contract generation and customization, rapid contract analysis, and streamlining routine document processing tasks. Based on demonstrations the sponsor agreed on their potential. Yet, while the potential benefits are substantial there are concerns with data privacy and security, accuracy and reliability, legal and compliance issues, and integration with existing systems. To mitigate these concerns and maximize benefits, the research team suggests focusing on local, open-source LLM solutions like LLaMA or Phi. These models can be deployed on-premises, ensuring data privacy and security while providing powerful LLM capabilities including customization and specialization.

BACKGROUND

CONTRACTOR PERFORMANCE ASSESSMENT REPORTING SYSTEM (CPARS) REPORT

The Contractor Performance Assessment Reporting System (CPARS) is used to document contractor and grantee performance information. Although not mandated by Acquisition Management and Integration Center (AMIC) policy, several divisions have requested monthly CPARS status reports due to infrequent system use and user inexperience.

The current process for generating these reports involves four main steps:

1. Extracting the Status Report from CPARS, including current, due, and overdue contract statuses;
2. Formatting the data in Excel for further processing;
3. Processing the data through a custom "CPARS Report Master" Excel file; and
4. Distributing division-specific reports via email.

This manual process, while functional, provides regular updates on contract statuses in CPARS, aligning with federal best practices for routine monitoring. However, its reliance on manual intervention and specific Excel files presents opportunities for automation to enhance efficiency and reduce potential errors.

LARGE LANGUAGE MODELS (LLMS)

Large Language Models (LLMs) are advanced machine learning (ML) systems trained on extensive text data, capable of understanding and generating human-like text across various topics and tasks. Key abilities include:

1. Natural language processing and generation
2. Task adaptability
3. Broad knowledge base
4. Contextual understanding

In contract and procurement management, LLMs could potentially:

- Automate information extraction[1]
- Generate summaries[2]
- Assist with query resolution
- Enhance unstructured data analysis

There are many LLM based approaches, three of which were considered for this task and two are detailed in this report. The first approach is based upon the ReAct architecture[3] which allows for improved interaction between chain of thought prompting and action plan generation. ReAct achieves this through combining reasoning and acting related to a task in an interleaved manner. This allows the LLM to adjust generated plans in relation to user input. From a practical standpoint this involves using the LangChain[4] tool to build a LLM based bot that can use ReAct's prompt architecture to automate code generation for manipulating CPARS data.

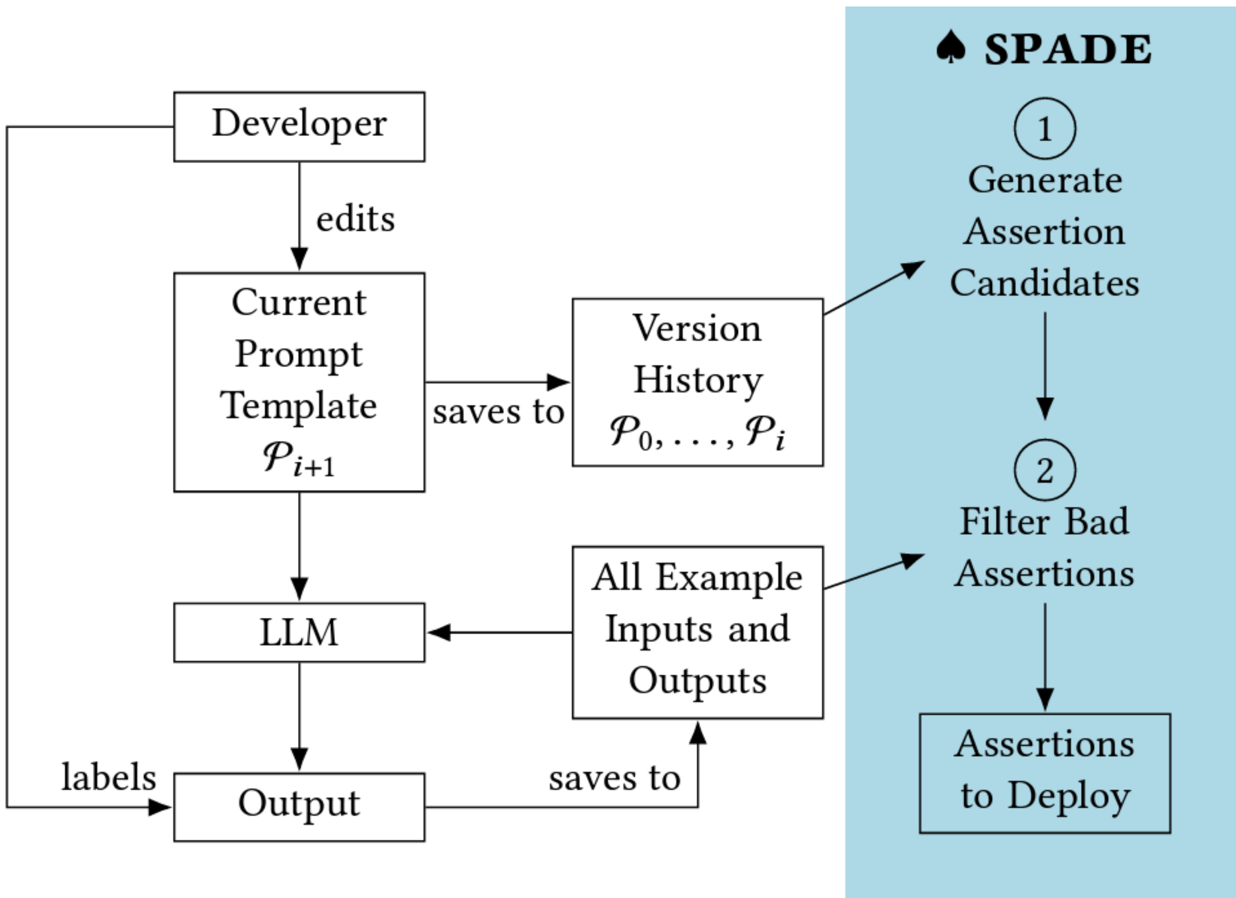

Figure 1. The SPADE Architecture, which analyzes the deltas (i.e., diffs) between consecutive prompt templates to generate assertions. Then, uses labeled pipeline inputs and outputs to filter out redundant and inaccurate assertions, while maintaining coverage of bad outputs.[5]

The second approach streamlines automation tasks by leveraging the latest advancements in code generation technology. Users interact with artificial intelligence (AI) by providing detailed prompts that outline the specific requirements for the automation task. These prompts include the scope of the task, the desired input and output formats, and any processing steps that need automation. Along with the prompts, users specify the exact input and output formats required for the automation task. This specification helps in tailoring the generated code to match the expected data processing flow accurately. Users also provide sample data, which serves as a reference for the AI to understand the data structure and content expected to be handled by the automation. Based on the user's prompts and the specified input/output formats, the AI generates the necessary code. This process includes the installation of all necessary dependencies to ensure that the code runs smoothly in its intended environment. The aim of the code generation model is to produce a script that is optimized for the task, considering the provided sample data and the specified requirements.

The third and final approach makes use of the SPADE methodology[5] and LangChain to produce AI generated code. This method starts by converting the Excel spreadsheets to data frames using the Python Pandas[6] library. LangChain is then used to integrate spreadsheet data into the LLM context, enabling comprehension and processing. The user then creates a set of instructional prompts, tailored to replicate the current steps required for report generation. These prompts guide the LLM in processing the spreadsheet data effectively. Next, the LLM processes the prompts and the integrated spreadsheet data, generating results compatible with Pandas. Finally, the Pandas data frames are converted back into Excel for distribution and analysis.

Using the SPADE methodology, focusing on generating and filtering assertions based on prompt changes, plays a pivotal role. It involves analyzing prompt deltas to identify and mitigate potential LLM failures, ensuring the generated code is both effective and reliable. The process is designed to minimize failures and operational risks, thereby enhancing the LLM's utility in automating the CPARS report process.

METHODOLOGY

CPARS SOLUTION

After thorough review of the problem space and a meeting with the sponsor, it was determined that the needs of the sponsor for the CPARS solution were better fulfilled by a solution not requiring LLMs. Instead, a more traditional, straightforward tool was developed using Python and various libraries for spreadsheet manipulation.

The tool created is a desktop application with a graphical user interface (GUI) (Figure 2), designed to automate the process of generating CPARS reports. Key features of this tool include:

1. **File Selection**: Users can easily select the required input files (CPARS Report Master, CPARS Status Report, and CPARS Monthly Report) through a user-friendly interface.

2. **File Format Handling**: The tool can handle both .xls and .xlsx file formats, automatically converting .xls files to .xlsx when necessary.

3. **Data Preprocessing**: It performs necessary data cleaning and formatting, including date standardization and sorting.

4. **Report Generation**: The tool automates the transfer of specific data from input files to output reports, ensuring consistency and accuracy.

5. **Cross-Platform Compatibility**: The application is designed to work on both Windows and macOS operating systems.

6. **Executable Creation**: The tool can be compiled into a standalone executable file, making it easy to distribute and use without requiring Python installation.

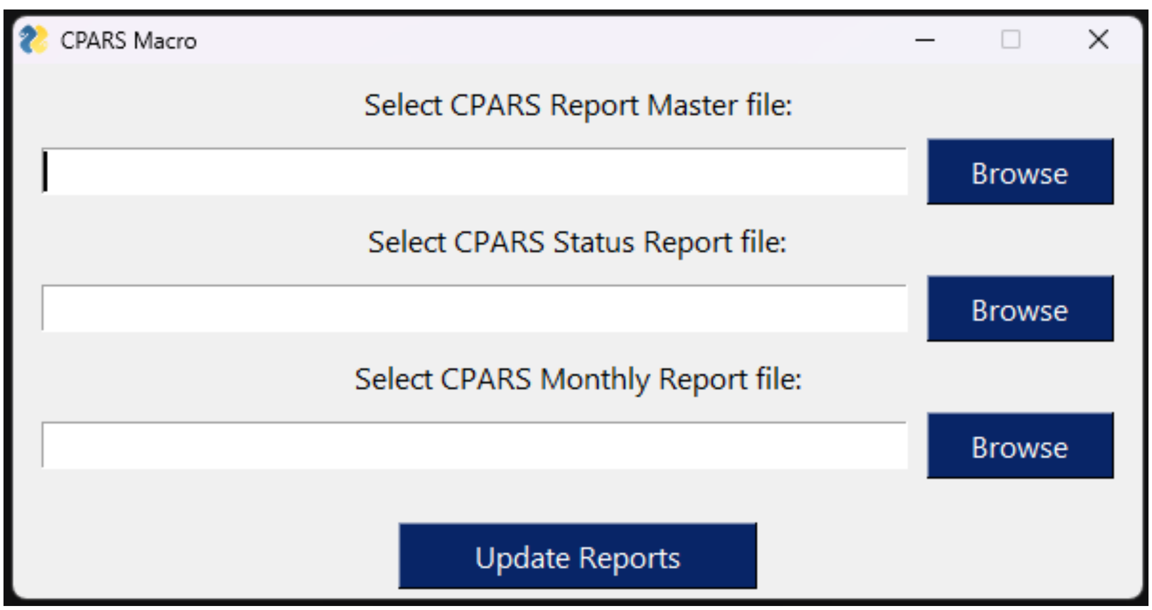

Figure 2. GUI of the CPARS report generation prototype

This approach was chosen over LLMs for several reasons:

1. **Consistency**: The task required exact matching and transfer of data between specific columns, which is more reliably achieved through direct programming.

2. **Efficiency**: For repetitive, structured tasks like this, a purpose-built tool can perform much faster than an LLM-based solution.

3. **Reliability**: The tool's behavior is predictable and consistent, which is crucial for handling sensitive contract information.

4. **Ease of Use**: The graphical interface makes the tool accessible to users without technical expertise.

5. **Data Privacy**: By processing data locally, the tool avoids potential privacy concerns associated with sending data to external LLM services.

In conclusion, while LLMs offer powerful capabilities for many tasks, this particular use case benefited more from a targeted, traditional programming approach. The resulting tool provides a streamlined, reliable solution for automating CPARS report generation, meeting the specific needs of the sponsor more effectively than an LLM-based alternative would have.

The compiled executable file (compatible with Windows and Mac) along with instructions for using it to create the CPARS report and a video demonstration are included as deliverables.

PROPOSED LLM SOLUTIONS

The research team developed demonstrations (demos) to showcase how LLMs can streamline contract management processes:

FIELD EXTRACTION DEMO

This tool demonstrates how AI can automatically extract important information from existing contract documents. Users can upload a contract and the system will identify and extract key details such as names, dates, and specific terms. This could significantly reduce the time spent manually reviewing contracts and minimize human error in data entry.

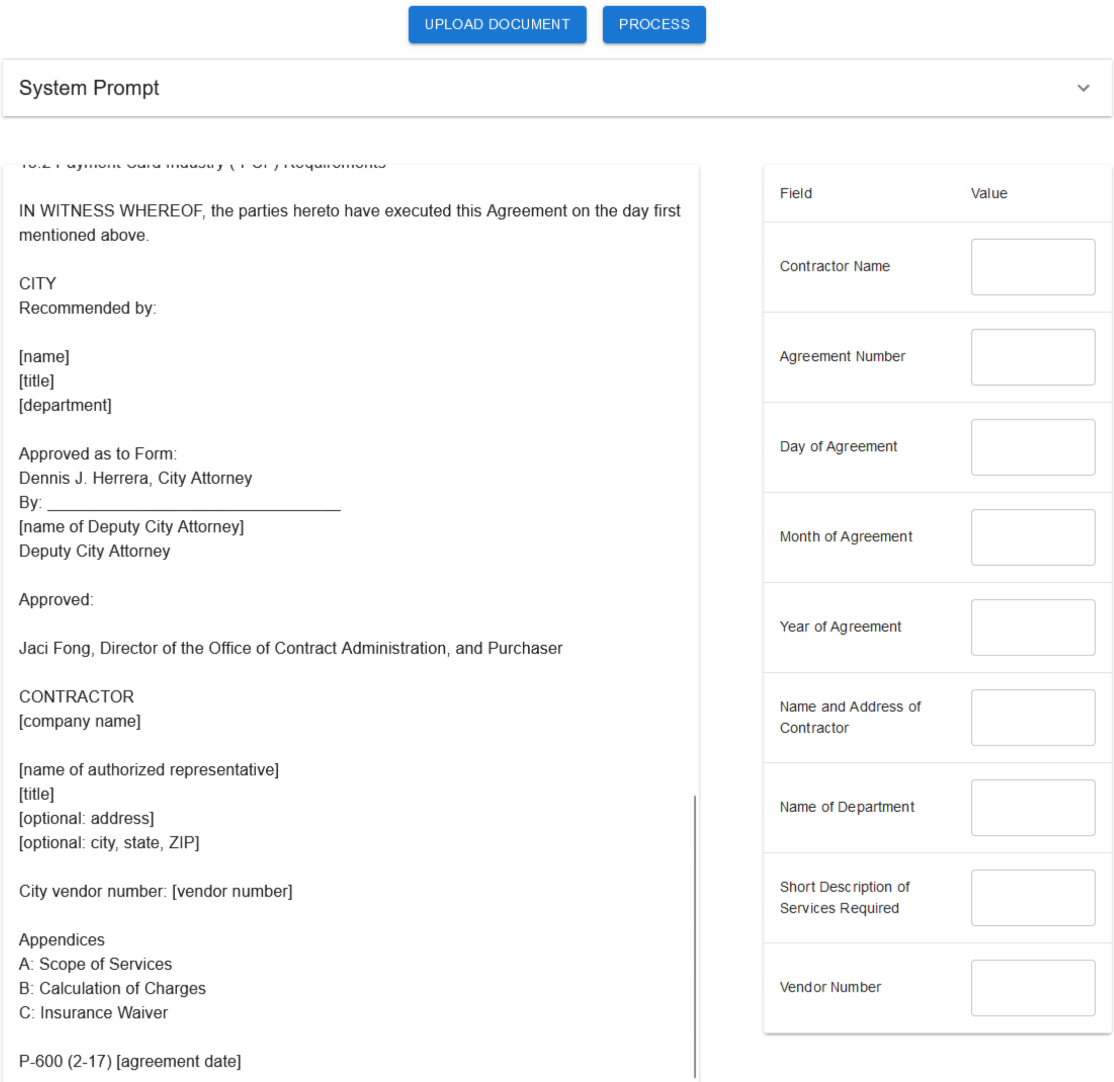

Figure 3. Screenshot of demonstration tool extracting structured information (i.e., known fields) from a contract

CONTRACT ASSISTANT DEMO AI

This demo shows how AI can help create customized contracts quickly. Users can select a contract type, input basic information, and choose specific clauses to include. The system then generates a complete, professionally worded contract based on these inputs. This could speed up the contract creation process while ensuring consistency across documents.

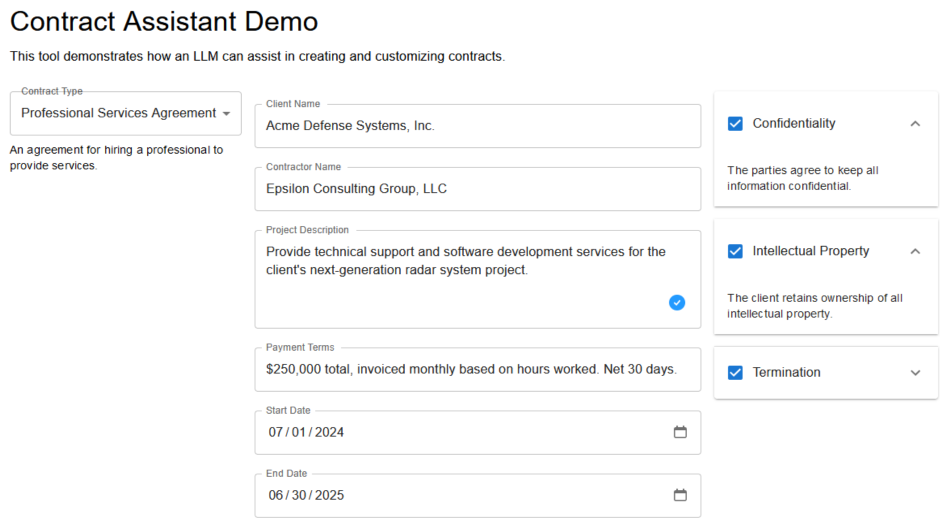

Figure 4. Screenshot of demonstration tool providing options of information and clauses to include in a contract with the assistance of AI.

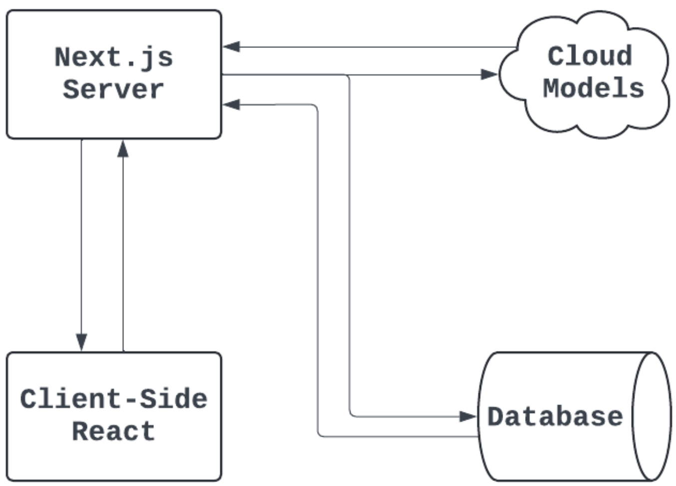

Figure 5. LLM demo architecture

These two demonstrations highlight the potential of AI to:

- Reduce time spent on repetitive tasks in contract management;
- Improve accuracy in contract creation and data extraction;
- Provide a user-friendly interface for complex legal processes; and
- Offer flexibility in contract customization while maintaining standardization.

A video recording of each demonstration has also been included in the deliverables.

CONCLUSIONS

The research team completed a thorough review of the problem space and feasible solutions to automate the reporting process. As the CPARS solution did not require the use of any LLMs, the team developed an executable, simple and straightforward interface to achieve the sponsor's goals and simplify the workflow. The executable file generates reports on the status of entries in CPARS broken down by contract division.

The deliverable is in the form of a compiled executable file that was written in the Python programming language. This program performs both the manual and automated steps of the sponsor's workflow and is more user-friendly, providing a GUI to automate the entire process.

Additionally, the research team provided two demonstrations using LLMs that we thought would be useful in the day-to-day of an acquisition professional.

RECOMMENDATIONS

The research indicates that LLMs have significant potential to enhance contract and procurement management processes. The team recommends exploring LLM applications in areas such as automated field extraction from existing contracts, assisted contract generation and customization, rapid contract analysis, and streamlining routine document processing tasks.

While the potential benefits are substantial, it's crucial to address key concerns including data privacy and security, accuracy and reliability, legal and compliance issues, and integration with existing systems. To mitigate these concerns and maximize benefits, the team suggests focusing on local, open-source LLM solutions like LLaMA or Phi. These models can be deployed on-premises, ensuring data privacy and security while providing powerful LLM capabilities.

By leveraging local, open-source models, organizations can benefit from advanced LLM capabilities while maintaining control over their data and computing environment. This approach not only addresses security and privacy concerns associated with cloud-based LLM services but also opens up new possibilities for customization and specialization.

APPENDIX A. LIST OF DELIVERABLES

1. CPARS Automation: the compiled executable file (compatible with Windows and Mac)
2. Updated instructions for creating the CPARS report using the new executable file
3. Video recording of the CPARS demonstration
4. CPARS Report Presentation
5. CPARS Monthly Report Example
6. Video recording of the Field Extraction demonstration
7. Video recording of the Contract Assistant demonstration

REFERENCES

[1] 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. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS '20), December 06, 2020. Curran Associates Inc., Red Hook, NY, USA, 9459–9474.

[2] Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B. Hashimoto. 2024. Benchmarking Large Language Models for News Summarization. Transactions of the Association for Computational Linguistics 12, (January 2024), 39–57. https://doi.org/10.1162/tacl_a_00632

[3] Yao, Shunyu, et al. "React: Synergizing reasoning and acting in language models." arXiv preprint arXiv:2210.03629 (2022).

[4] langchain-ai/langchain: Build context-aware reasoning applications. Retrieved July 19, 2024 from https://github.com/langchain-ai/langchain

[5] Shankar, S., Li, H., Asawa, P., Hulsebos, M., Lin, Y., Zamfirescu-Pereira, J.D., Chase, H., Fu-Hinthorn, W., Parameswaran, A.G., & Wu, E. (2024). SPADE: Synthesizing Assertions for Large Language Model Pipelines. arXiv. https://doi.org/10.48550/arXiv.2401.03038

[6] The pandas development team. "Pandas-Dev/Pandas: Pandas." Zenodo, December 8, 2023. https://doi.org/10.5281/zenodo.10304236.