Similarity Search Tool For Working Papers Utilizing BERT

Ethan Crouse, Eric Zou, Rahul Rengan Ramakrishnan
Capstone Coach: Hayden Ringer
Census Bureau Sponsor: Dr. Rae Ellis

Executive Summary

The U.S. Census Bureau maintains a repository of over 4,000 working papers aimed at sharing valuable research findings; however, ineffective search functionality makes locating relevant documents difficult and time-consuming. Researchers, both internal and external, face significant challenges due to the limited searchability since the only present ways to search are filtering by year and topic. Our team addressed these accessibility issues by developing a robust similarity search tool using advanced natural language processing (NLP) techniques. We automated the scraping of paper metadata from the Census Bureau website, stored this data efficiently, and implemented semantic search functionality using the pre-trained Sentence-BERT (SBERT) embeddings to identify relevant documents rapidly. Our user-friendly interface, developed using Flask and JavaScript, provides accurate and contextually meaningful search results. Preliminary tests indicate our model achieves high relevance (average precision of 0.92) and rapid query response times. Our solution significantly reduces the effort required to discover pertinent research, directly benefiting researchers and enhancing the value of the Census Bureau's research outputs. Despite limitations due to web scraping constraints and the reliance on abstract data, our tool demonstrably improves accessibility and usability of critical research resources.

1 Problem Statement

The core objective of our Capstone project was to enhance the accessibility and efficiency of searching the U.S. Census Bureau’s repository of working papers. Researchers currently struggle to locate relevant papers due to a rudimentary search interface that relies solely on basic keyword matches without contextual understanding. The project's priority was creating an effective, accurate, and easy-to-use similarity search system that significantly improves paper discovery. The resulting solution is crucial for researchers, policymakers, and academic institutions who depend on timely and accurate information dissemination from the Census Bureau studies. Improved search capabilities directly support efficient research practices, bolster external agency interactions, and amplify the societal and scientific impact of the Bureau's extensive research outputs. We received guidance from Census Bureau experts Ian Le, Maxwell Hope, and Rae Ellis, whose insights informed our project alignment with the Bureau’s practical research needs and methodological standards.

2 Ethical Considerations

2.1 Abstract-Length Bias

Our tool uses a natural language processing model to perform similarity searches based on working paper abstracts. Papers with longer and more detailed abstracts may receive higher similarity scores and are more likely to be the top search results. This introduces a bias that puts high-quality working papers with concise abstracts at potential limited visibility. This creates an uneven playing field in terms of exposure for working papers.

The ethical concern from the abstract length bias may unintentionally promote lengthy abstracts over concise ones. This could lead to high-quality papers with concise but detailed abstracts, not getting the exposure it could get if the NLP method looked at the entire paper.

The primary group impacted by the abstract-length bias would be researchers and authors who write concise yet detailed abstracts for their working papers, since their work will get less exposure through our search tool. Users of the search tool could also be affected by the bias since they aren't getting the best working papers for their search. To safeguard these ethical concerns, we are providing transparency about how the results are ranked.

2.2 Pre-Trained BERT

Our chosen NLP model, BERT, is pre-trained on a massive amount of text data. While this allowed the model to understand language context effectively, for example, being able to relate the words playground and slide, this training data may contain biases related to different demographic groups. Some terms or words that are trained to be related might be strongly associated with one group rather than another in the training data.

The ethical concern is the potential for our search tool to amplify the existing biases present in the pre-trained BERT model. This could lead to users being presented with search results that provide an incomplete view of the pool of working papers. If policymakers were relying on this tool, these biases could indirectly inform decisions and polices in a way that could put certain groups at a disadvantage. This could cause lasting negative effects on demographic groups that the bias is suppressing.

The groups that are most likely to be impacted by these ethical concerns are the demographic groups that are suppressed by the way the model was trained, with the possibility of bias in the training data. Another group that is affected is the general public that wants to get unbiased search results.

BERT, similar to other large language models, is trained on a massive amount of data and it proves to be difficult to even identify a bias to mitigate when it comes to these concerns. The best we can do is extensive testing on our model and being fully transparent that a bias in the training data is possible.

3 Literature Review

"A Comparison of Document Similarities Algorithms" (Gahman et al., 2023) evaluated five commonly used NLP techniques for similarity searches in academic papers: monolingual baseline (MT), cosine similarity between sentence embeddings (SEMB), word mover's distance (WMD), Sinkhorn (SNK), and Siamese long short-term memory (LSTM). This research directly informed our methodology by presenting the strengths and weaknesses of each approach across three distinct datasets. The study classified methods into statistical, neural network-based, and knowledge-based categories but notably omitted runtime comparisons, prompting us to investigate performance metrics independently.

Of the methods assessed, the Multi-Task Deep Neural Networks (MT-DNN) demonstrated superior accuracy. Further exploration into MT-DNN through "The Microsoft Toolkit of Multi-Task Deep Neural Networks for Natural Language Understanding" (Xiaodong et al., 2020) revealed its strengths in multi-task scenarios, particularly through shared contextual embeddings, which streamline simultaneous training on multiple NLP tasks. This made MT-DNN appealing for potentially improving accuracy and efficiency in similarity scoring and search.

Ultimately, we selected Sentence-BERT (SBERT) for implementation, based on insights from "Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks" (Reimers et al., 2019). SBERT significantly optimizes performance compared to traditional BERT by embedding sentences rather than words, reducing computation from millions of comparisons to a linear, vector-based comparison method. The dramatic improvement in processing speed, from over 65 hours to mere seconds in large-scale scenarios, confirmed SBERT as the ideal choice for efficiently computing paper similarity in our project.

4 Project Criteria

4.1 Data Collection

1. *Volume of Data*: We want to download all 4,431 papers from the website
2. *Fidelity*: We want each paper's complete title, paper number, abstract, and PDF.
3. *Speed*: The program should scrape all papers from the website in under an hour
4. *Ease of Use*: In case the data must be collected again, the scraping tool should be a single Python program. The sponsor should rate it at least 4 out of 5 on a scale for ease of use.
5. *Data Format:* The scraped data should come out in .csv format.

4.2 NLP/Similarity Searching

1. *Speed*: The inference should take less than 30 seconds to run on a M4 24-core MacBook Pro laptop.
2. *Language:* The NLP backend should be programmed in Python
3. *Accuracy*: Given a sample of 10 papers, we can find all 10 of them using the search engine. Similar papers provided should be in the same general category as the given paper.
4. *Cost:* The framework used must be free.

4.3 Interface

1. *Ease of Use:* Our sponsor should rate the ease of use at least 4 out of 5 on a scale for ease of use
2. *Cost:* The solution must be free to use
3. *Transferability:* The solution should be a downloadable repository and should not involve transferring any accounts.

5 Selected Solutions

To scrape papers from the Census website, we used Selenium to click through the pages of the Census website and gather links to the individual papers. We then used requests to retrieve the pages for each paper and used Beautiful Soup to gather the data. While we provided details on implementation and performance of four different NLP methods(TF-IDF, Word2Vec, Doc2Vec, SBERT), we chose to use SBERT for our implementation of the search engine as it provided the best groupings while still meeting all our requirements for speed and ease of use. To store the embeddings and metadata, we used FAISS vectorstore. For the interface, we chose to use a Python backend and a JavaScript frontend connected using Flask.

5.1 Web Scraping

The Census Working Paper repository spans 124 pages, each listing roughly 30 papers. Because the URL remains the same as you navigate between pages, we employed Selenium to simulate clicking through the pagination controls. Selenium is an open-source framework that exposes a WebDriver API for programmatic browser interaction. Once we had gathered all of the individual paper URLs, we used Python's Requests library, a simple HTTP client, to fetch each page's content. We then put the requested contents through Beautifulsoup, a Python library for consistently fetching elements of a page by turning websites into a tree structure.

5.2 BERT Architecture

To power our similarity search, we used SBERT, which is a version of BERT (Bidirectional Encoder Representations from Transformers) designed for comparing sentences efficiently. BERT is a language model that looks at all the words in a sentence at once (not just left to right or right to left), so it understands how each word relates to every other word in context.

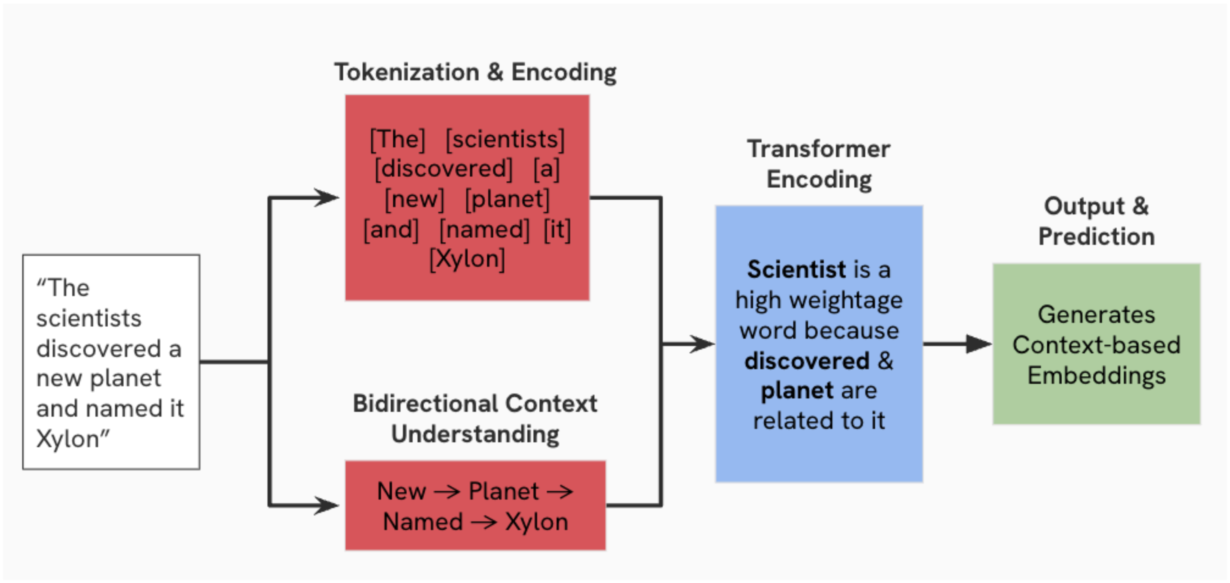

"The scientists discovered a new planet and named it Xylon"
Tokenization & Encoding
[The] [scientists] [discovered] [a] [new] [planet] [and] [named] [it] [Xylon]
Bidirectional Context Understanding
New → Planet → Named → Xylon
Transformer Encoding
Scientist is a high weightage word because discovered & planet are related to it
Output & Prediction
Generates Context-based Embeddings

Figure 1. *Visual of BERT applied to the sentence "The scientists discovered a new planet and named it Xylon"*

Figure 1 shows an example of how BERT works under the hood. We used the sentence *"The scientists discovered a new planet and named it Xylon"* to walk through the steps BERT takes when creating an embedding. First, the sentence is tokenized — this means splitting it into smaller units like words or subwords. Each token is then passed through BERT's transformer layers, which look at the entire sentence at once, rather than reading it left to right or right to left. BERT reads in both directions and builds a deep understanding of how each word relates to others. For example, even though "named" and "Xylon" are far apart from "planet," BERT still picks up on the fact that they're part of the same idea.

As it processes the sentence, BERT figures out which words carry the most meaning in context by assigning them higher weights. In this case, "scientists" ends up being especially important because it's tied to both "discovered" and "planet." This process results in what's called a **context-based embedding**, a numerical vector that represents the overall meaning of the sentence. These embeddings are designed so that semantically similar sentences will be closer together in vector space.

SBERT builds on this idea by making it easier to compare sentences directly. Regular BERT would require every sentence pair to be run through the model at query time, which doesn't scale well. SBERT instead lets us precompute embeddings for all our abstracts, store them using a vector index (we used FAISS), and then run fast cosine similarity comparisons between the query and the database. This lets our tool return results in under a second while still capturing the deeper meaning behind a user's query, making SBERT the best option for our use case.

5.3: User Interface

Our final user interface is built on Flask, a lightweight Python web framework that bridges our backend logic with an interactive frontend. Flask's intuitive URL routing, built-in templating engine , and robust request-handling capabilities let us quickly develop and maintain clear separation between server-side functionality and client-side presentation. Moreover, Flask's modular architecture and extensive plugin ecosystem, including extensions for database integration, user authentication, and real-time communication, give us the flexibility to tailor our application precisely to project requirements.

For the frontend, we opted to integrate a JavaScript-based framework rather than relying solely on static HTML. While HTML and CSS handle basic layout and styling, JavaScript enables dynamic content updates, responsive user interactions, and seamless communication with our Flask backend via asynchronous requests. This combination of a Python-driven server paired with a JavaScript frontend ensures both simple development cycles and a polished, user-friendly experience.

6 Results

6.1 Results - Data Scraping

For data scraping, we have a Python file that will collect information about all papers on the Census Working Papers website. It uses Selenium to crawl through the main page, clicks on the links for each paper and uses Requests and Beautifulsoup to collect the Author, Title, Abstract, Date Published, and files for each paper. The main problem we have is that due to the website's bot protection, we can only access a page every 15 seconds. This makes it so the script takes upwards of 18 hours to run. This is an issue we cannot get around, but our sponsors have let us know that it is not an issue for them, as they would be using internal tools. The dataset we are working with currently was scraped on April 7th. Out of 4,434 papers that were listed on the census website at the time, only 4,204 had working links. We scraped those, and of those we had to remove ~100 papers because they had empty abstracts. After that, besides a couple of papers that had the entire paper uploaded as an abstract, most abstracts were 500 to 1,500 words long.

Once we had cleaned the dataset, we performed exploratory data analysis. As a first step, we examined the most frequently occurring words to identify any potential issues with over-represented terms.

Table 1. Top 14 most frequent phrases in the abstracts.

Rank	Phrase	Frequency	Rank	Phrase	Frequency
1	data	6077	8	acs	1971
2	census	4510	9	income	1910
3	survey	3767	10	using	1849
4	bureau	2314	11	research	1724
5	estimates	2233	12	population	1633
6	paper	2217	13	results	1569
7	census bureau	2033	14	use	1379

Table 1 shows the 14 most frequent words in the dataset. The most noticeable thing is that most of the top frequent words are highly tied to the Census Bureau. It introduced the challenge of a high concentration of census-related terms compressing the semantic space, making it more difficult to distinguish between papers and reducing the diversity of recommendations, and potentially hiding papers that don't use words about data, census, or surveys.

To better understand this issue, we embedded the abstracts using all-MiniLM-L6-v2, a compact and widely used BERT-based model. We then used UMAP and T-SNE to reduce the 384-dimensional embeddings to 3 dimensions and visualized the result with Plotly. We provided the sponsor with a 3d interactive plot stored in an HTML file that can be spun around and shows the name of a paper when hovered over. Even when reduced from 384 to 3 dimensions, we were able to see solid groupings of papers that made sense to us. With that, we decided the dataset was good as is and we wouldn't need to do modifications like filtering out census-related words.

6.2 Results - NLP

Because one of our deliverables is a report on how well different methods work, we compared four different semantic search methods: TF-IDF, Word2Vec, Doc2Vec, and SBERT. To quickly summarize these methods TF-IDF compares how frequently words appear in a document, so that if two documents share words that appear very rarely in the whole dataset, then those two documents would be more similar to each other. Word2Vec turns words into vectors based on their meanings, allowing us to use mathematical concepts like cosine similarity on words. A very popular example of word2vec is that adding together the vectors for *king* and *woman* while subtracting the vector for *man* should result in a vector roughly equal to *queen*. Doc2vec expands on this by assigning a vector for the meaning of an entire document rather than just the individual words. SBERT takes the idea of vectorizing words and makes the vectors more accurate by including context.

In order to compare these methods, we used precision@k, then timed the searches and embedding of the dataset. Precision@k is simply having the algorithm return the top k results, then manually labeling them as on-topic or off-topic. The number of on-topic results divided by k is the precision@k.

Table 2. Precision@5 for TF-IDF, Word2Vec, Doc2Vec and SBERT

	Income Disparity	Small Business	Administrative Records
TF-IDF	0.20	0.40	0.40
Word2Vec	0.60	0.0	0.0
Doc2Vec	0.40	0.0	0.20
SBERT	0.80	0.60	0.80

Table 2 shows the precision@k of the four methods on three different searches with k = 5. These searches Table 1 shows that SBERT outperforms the other models in semantic search accuracy. In particular, the MPNet version of SBERT returned the most consistently relevant papers across all queries. For example, when we searched *"racial disparities in income distribution"*, the top results included papers on racial income gaps, intergenerational mobility, and wage inequality — even without keyword overlap. In contrast, TF-IDF required exact wording, and Word2Vec and Doc2Vec failed to handle multi-word queries effectively.

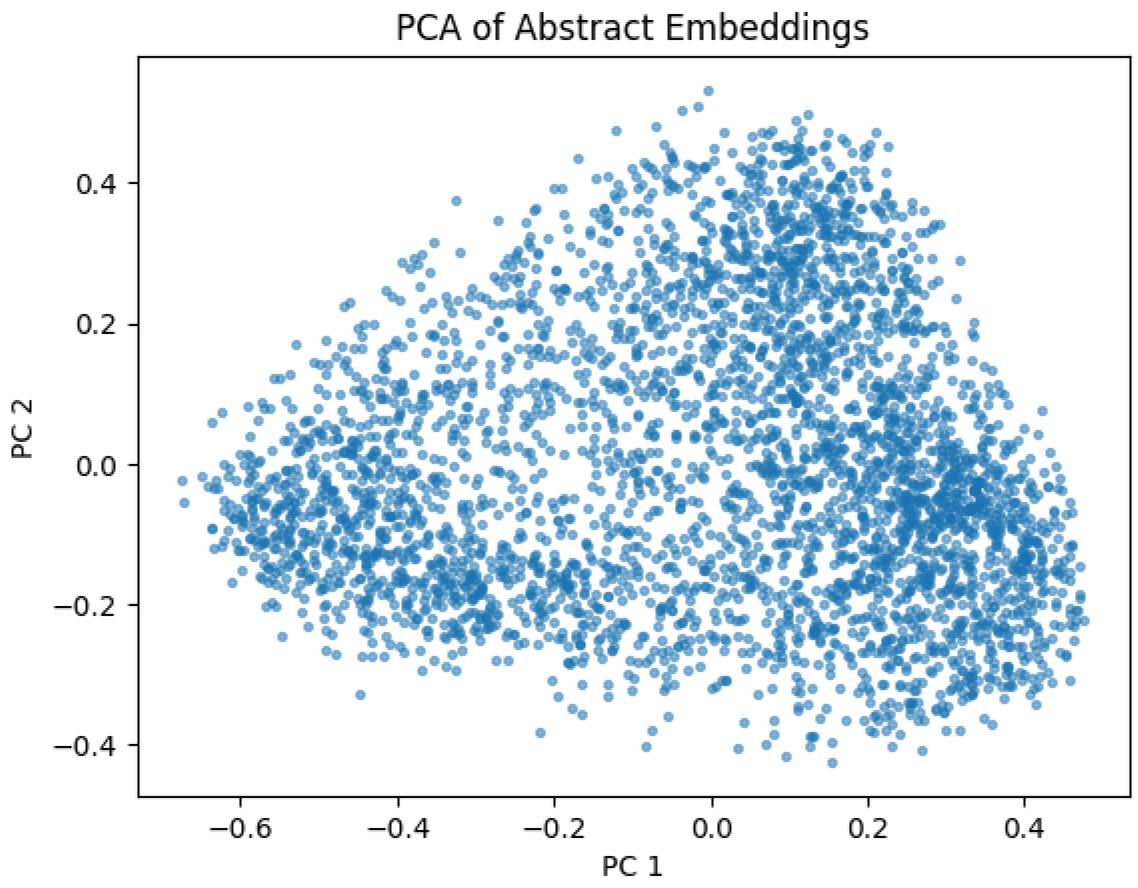


Fig 2: PCA clustering analysis of embeddings

To better understand the structure of the embeddings, we visualized them using PCA. Figure 2 shows a clear cluster formation, confirming that SBERT is grouping related topics together in the embedding space. We also computed cosine distance from the mean embedding vector and found several outliers, papers that diverged from typical Census themes. These included technical or niche topics, like autocovariance estimation or conversations with statisticians, demonstrating SBERT's ability to preserve uniqueness without relying on frequent terms.

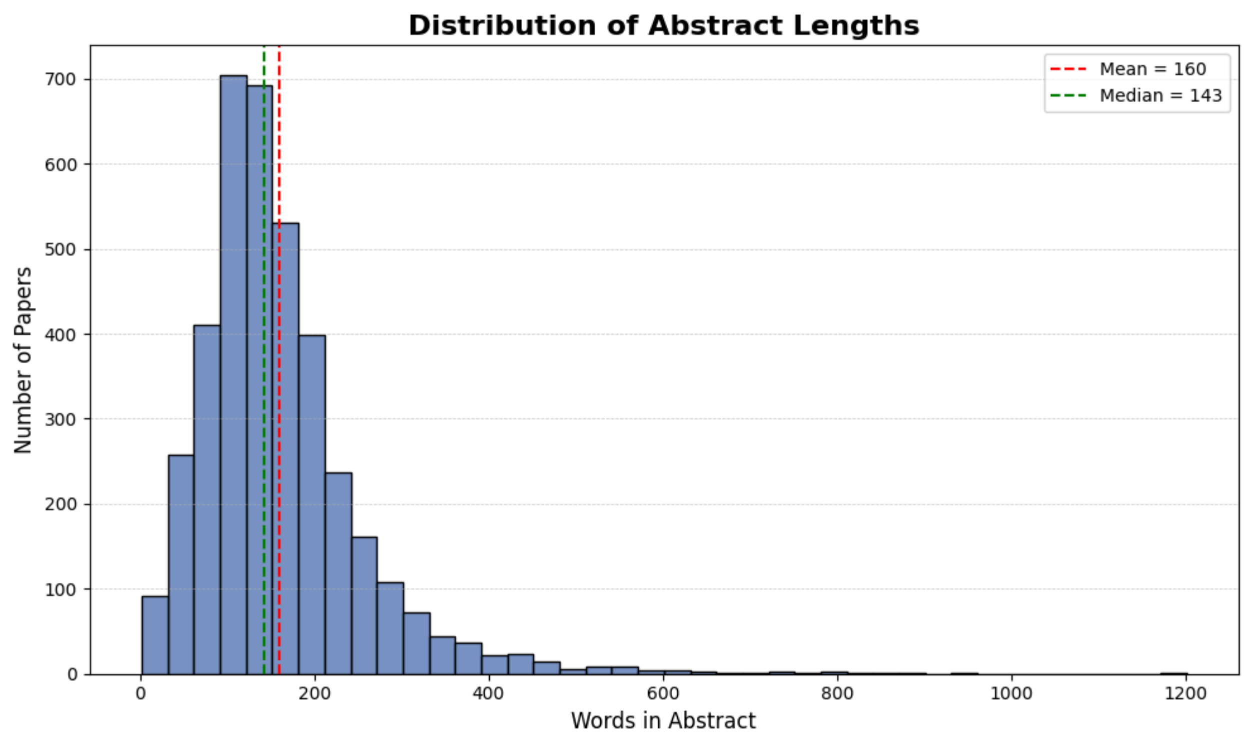


Fig 3: Abstract word length distribution

We examined the impact of abstract length. Figure 3 shows that most abstracts are between 100–250 words, with a long tail of longer entries. Since longer abstracts provide more context, this may bias SBERT's output by giving those papers an edge in similarity scoring, validating the ethical concern we raised in Section 2.

Table 3. Models and Times to Embed Vectors on Google Colab basic CPU instance

Model	Time to Embed (Seconds)
Word2Vec	1.42
Doc2Vec	26.17
SBERT	370.47

When doing the Precision@k measurements, we also timed the models in embed time and search time. Search time for all models was well below one second, but there was a large difference in the times to embed. While SBERT took the longest, we still chose to recommend SBERT because embeddings are stored and not computed with each search, so the embedding time is only an issue when re-embedding the whole dataset.

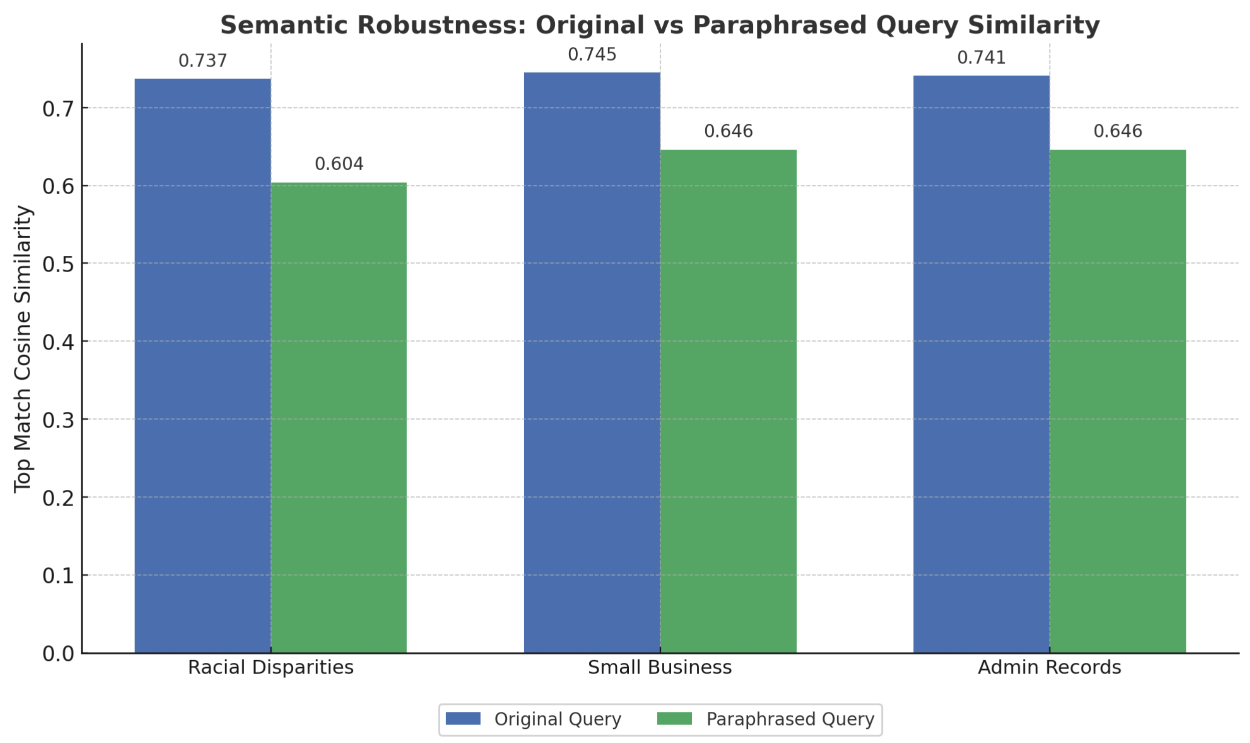


Fig 4: Top cosine similarity score for original queries vs paraphrased versions.

A key evaluation metric we also wanted to test was if changing key words in a sentence, but retaining its meaning, would have a meaningful impact on the results. To test the semantic robustness of our model, we evaluated how well it handled queries that were rephrased using different wording but retained the same core meaning. For each of our three benchmark topics, we created a paraphrased version of the original query and compared the top match cosine similarity scores returned by SBERT. As shown in Figure 4, the top results for paraphrased queries remained closely aligned with those of the original queries, with only moderate drops in similarity scores (typically 0.09 to 0.13 points). For example, the query *"racial disparities in income distribution"* returned a top score of 0.737, while its paraphrased counterpart *"ethnic differences in tax brackets"* returned 0.604. In all cases, the returned papers were still topically relevant, confirming that SBERT captures the broader meaning behind user queries rather than relying on exact keyword matches. These results demonstrate that our search system is resilient to natural variation in phrasing, a key requirement for real-world usability.

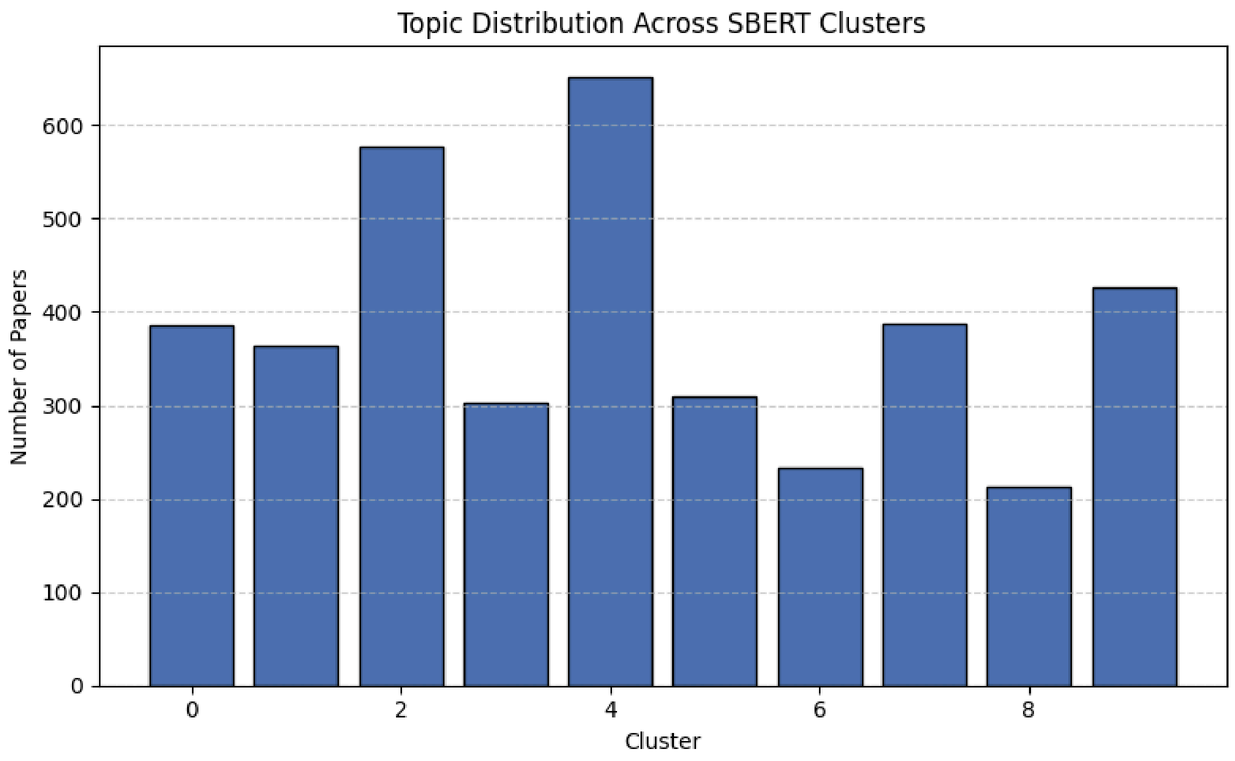


Fig 5: Topic Distribution Across SBERT Clusters

The final, and arguably the most important metric we needed to test was the quality of data that SBERT would make decisions on. To assess the diversity of topics captured by our SBERT embeddings, we applied KMeans clustering to the 384-dimensional abstract vectors using $k = 10$. As shown in Figure 5, the resulting clusters are moderately well balanced, with the largest cluster containing ~650 papers and the smallest just over 200. This suggests that while certain research areas dominate (e.g., ACS data, economic metrics), the corpus still spans a wide range of topics.

We also calculated the silhouette score, a measure of clustering separation. Our result of 0.057 indicates modest overlap between clusters, which is expected in a domain where many abstracts share Census-related vocabulary. Nonetheless, the presence of coherent clusters confirms that SBERT captures enough semantic variation to support features like topic-based navigation and diversity filtering in the future.

6.3 Results - GUI

We developed two graphical user interfaces (GUIs): an initial proof-of-concept version and a more refined final implementation. The prototype was built using Streamlit, a Python framework designed for easily creating data science applications. This allowed us to quickly put together a functional interface with minimal development overhead.

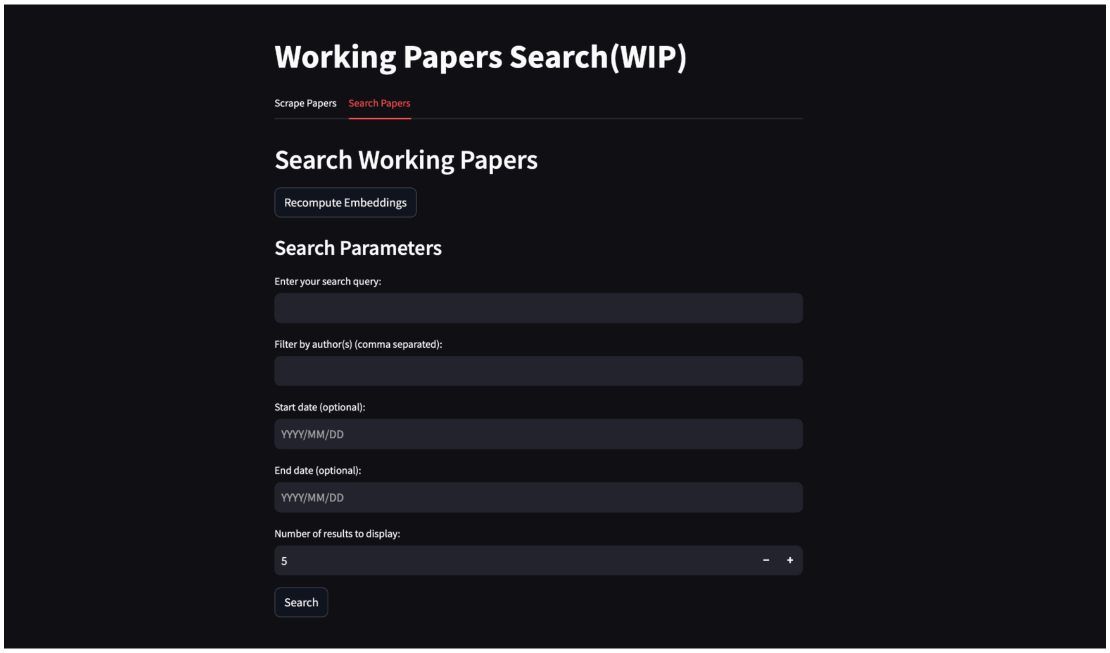
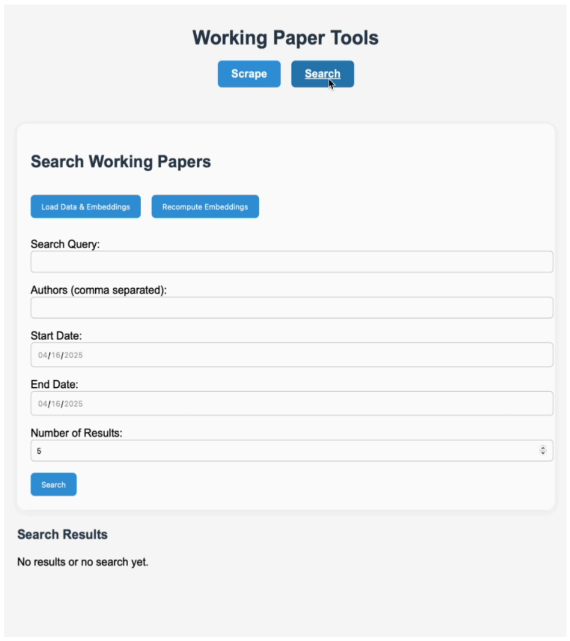


Fig. 6 Home page of Streamlit application

Figure 6 is a screenshot of the home page for this application. It provided a page to scrape pages and a page to embed, then search through papers. It was able to get very usable search results, so we displayed this application in the midterm presentation, but it did not provide the freedom to customize as we wanted, and deploying the actual application on a production scale would cost money from Streamlit. This was a problem as our sponsor wanted a solution that wouldn't cost anything extra outside of the hosting costs on their servers. For this reason, we decided to make a second implementation using Flask and JavaScript. It runs as a Python backend being called to do the processing for a JavaScript frontend. This approach gave us full control over the interface and allowed for more extensible and scalable features.



Working Paper Tools

Search Working Papers

Search Query:

Authors (comma separated):

Start Date: 04/16/2025

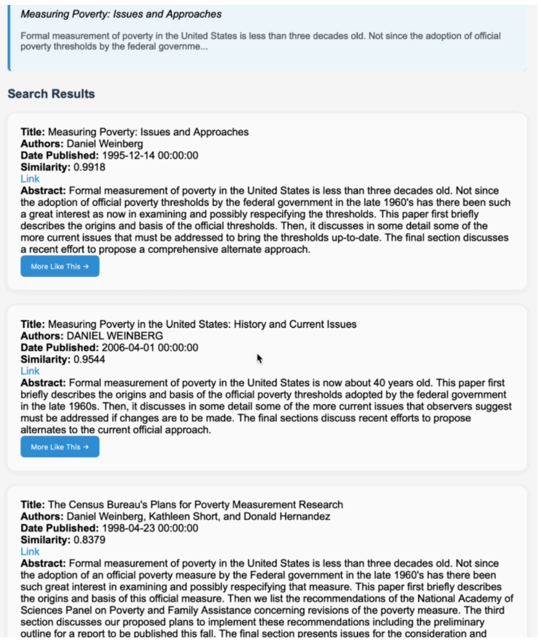
End Date: 04/16/2025

Number of Results: 5

Search Results

No results or no search yet.

Fig. 7 Homepage of Flask application



Measuring Poverty: Issues and Approaches

Formal measurement of poverty in the United States is less than three decades old. Not since the adoption of official poverty thresholds by the federal governme...

Search Results

Title: Measuring Poverty: Issues and Approaches
Authors: Daniel Weinberg
Date Published: 1995-12-14 00:00:00
Similarity: 0.9918
Link
Abstract: Formal measurement of poverty in the United States is less than three decades old. Not since the adoption of official poverty thresholds by the federal government in the late 1960's has there been such a great interest as now in examining and possibly respecifying the thresholds. This paper first briefly describes the origins and basis of the official thresholds. Then, it discusses in some detail some of the more current issues that must be addressed to bring the thresholds up-to-date. The final section discusses a recent effort to propose a comprehensive alternate approach.

Title: Measuring Poverty in the United States: History and Current Issues
Authors: DANIEL WEINBERG
Date Published: 2006-04-01 00:00:00
Similarity: 0.9544
Link
Abstract: Formal measurement of poverty in the United States is now about 40 years old. This paper first briefly describes the origins and basis of the official poverty thresholds adopted by the federal government in the late 1960s. Then, it discusses in some detail some of the more current issues that observers suggest must be addressed if changes are to be made. The final sections discuss recent efforts to propose alternates to the current official approach.

Title: The Census Bureau's Plans for Poverty Measurement Research
Authors: Daniel Weinberg, Kathleen Short, and Donald Hernandez
Date Published: 1998-04-23 00:00:00
Similarity: 0.8379
Link
Abstract: Formal measurement of poverty in the United States is less than three decades old. Not since the adoption of an official poverty measure by the Federal government in the late 1960's has there been such great interest in examining and possibly respecifying that measure. This paper first briefly describes the origins and basis of this official measure. Then we list the recommendations of the National Academy of Sciences Panel on Poverty and Family Assistance concerning revisions of the poverty measure. The third section discusses our proposed plans to implement these recommendations including the preliminary outline for a report to be published this fall. The final section presents issues for the consideration and

Fig. 8 Search result of Flask application

Figures 7 and 8 display the homepage and an example search result from the Flask-based application.. While it looks similar on the surface to the Streamlit app, coding it from the ground up gave us the freedom to implement a "more like this button easily," and we can add more pages. One of the future goals is to add a page looking at the specific paper to show insights about the specific paper, like where in the paper is the most similar, and what files are available. This Flask and JavaScript solution meets all the criteria outlined in the sponsor's requests and allows us to implement new features easily.

7 Limitations

While our semantic search tool greatly improves on the existing keyword-based system, it comes with several limitations related to data quality, model architecture, and performance constraints.

7.1 Abstract-Only Indexing

One of the biggest limitations is that we only index paper abstracts, not full texts. This decision was made due to scraping constraints, since most working papers are uploaded as PDFs that are not readily accessible in plain text. As a result, important details in the main body of the papers are not searchable, and our system may fail to retrieve relevant documents when the abstract is vague or underdescriptive. Additionally, some abstracts were missing or malformed, which reduced our final dataset from 4,434 papers to around 4,100 usable entries.

7.2 Abstract Length Bias

As noted in Section 2, our use of SBERT introduces a subtle but important bias toward longer abstracts. Longer abstracts contain more tokens and semantic information, which increases the likelihood that SBERT will find overlapping context with a given query. This could result in shorter, more concise (but high-quality) abstracts being unfairly ranked lower, even if they are topically appropriate.

7.3 Pre-Trained Model Bias

The SBERT model we used was pre-trained on large generic corpora like Wikipedia and web data. While this enables high performance in general semantic tasks, it means that certain Census-specific terminology or statistical jargon may not be captured as effectively. This may reduce accuracy for queries involving uncommon or domain-specific phrases (e.g., “synthetic estimates,” “ACS-PUMS imputation,” or “noise-injection disclosure avoidance”).

7.4 Computational Bottlenecks

Embedding 4,000+ abstracts using SBERT (MPNet) takes time — up to two minutes on a standard laptop CPU. While this is only a one-time cost (since embeddings are stored), it could become a bottleneck if the dataset is updated frequently or if full-paper indexing is introduced in the future. Running queries is fast (<1 second), but embedding or re-embedding the entire corpus remains expensive without GPU acceleration.

8 Interpretation of Results

Our similarity search tool provides the U.S. Census Bureau with a meaningful upgrade to the current working paper search experience by introducing NLP-driven search functionality. Unlike the existing system, which only filters by year and topic, our tool allows users to enter a custom query and receive papers that are highly relevant even when exact keywords aren't used. This significantly improves the discoverability of research done at the U.S. Census Bureau for both internal researchers and external stakeholders, such as policymakers who rely on Census data for decision-making.

From our testing, the system consistently returned contextually appropriate papers within a second of submitting a search query. This is made possible by Sentence-BERT (SBERT), a model that understands relationships between ideas rather than just matching words. For example, a search like "racial income inequality" correctly surfaces papers on intergenerational mobility and wage gaps—topics that are topically similar but don't always share exact phrasing. These capabilities directly support the Bureau's mission to disseminate research efficiently and equitably.

The results also point to new directions the Bureau might consider. Our analysis suggests that indexing the entire text of papers, rather than just abstracts, would improve accuracy—especially for papers with brief or vague abstracts. We also identified edge cases where Census-specific jargon wasn't well understood by the pre-trained model, indicating an opportunity to improve the system further by fine-tuning the model on internal datasets or terminology. Additionally, our embedding visualizations showed strong clustering, which opens the door to implementing topic-based browsing or automatic categorization features in the future. We also noted that longer abstracts tended to rank higher in search results, which suggests a slight bias that could be addressed in future scoring adjustments.

9 Individual Contributions

Ethan
Technical Contributions: Wrote code to use Selenium to collect metadata from working papers. Beautification of the user interface, developed a webscraping demo for the tools and techniques presentation, and assisted in fine-tuning the model to maximize searchability accuracy.

Non-technical Contributions: Presented at midterm, presented a demo at Tools & Techniques, and presenting at the final presentation, large contributions to all technical memos.

Eric
Technical Contributions: Coded Streamlit prototype application presented at the midterm presentation. Coded all backend and frontend except for final page design for the final product. Did data analysis on the dataset as well as precision@k and runtime measurements for Word2Vec, Doc2Vec, and TF-IDF

Non-Technical Contributions: Presented at midterm and will present in final presentation, presented at Tools & Techniques. Contributed to all tech memos. Wrote final report for sponsor

Rahul
Technical Contributions: Fully implemented the SBERT-based semantic search pipeline. This included preprocessing the dataset, selecting and integrating the all-mpnet-base-v2 model and the all-MiniLM-L6-v2, generating embeddings, and building the full search logic. Also handled the development of evaluation benchmarks, including precision@k, paraphrased query robustness, semantic outlier detection, and clustering analysis. Designed and executed all testing for model performance and interpretability, including the PCA plots, abstract length bias analysis, and cosine similarity diagnostics used throughout the Results section.

Non-Technical Contributions: Presented at the Elevator Pitch, Midterm Presentation, will present at the final presentation. Contributed to all tech memos and write-ups.

10 Conclusions and Future Work

10.1: Conclusions and Summary

Our goal was to fix the problem of poor searchability in the Census Bureau's working paper archive. The only way to filter papers before was by topic and year, which made it hard to actually find anything specific. We built a search engine that uses SBERT to understand meaning, not just exact words, and return papers that are topically relevant even if the phrasing is different. The end result is a tool that makes it way easier to explore and discover research in the Census paper ecosystem.

From start to finish, the system performed how we wanted. The scraper collected over 4,000 usable abstracts despite rate limits, and the SBERT model consistently returned strong results across multiple topics. It even handled paraphrased queries well, showing that the model can understand the actual intent behind a search. The frontend we built with Flask is simple but flexible and works fast. Overall, our tool is a big improvement over the old system and actually makes the paper archive usable.

10.2: Future Work

There are a few clear directions this project could go next. One is expanding beyond abstracts to include full paper text. Right now, if something important isn't in the abstract, it can't be searched. Getting access to the full PDFs and embedding them would make the system a lot more powerful and reduce the chance of missing key papers.

Another idea is fine-tuning the SBERT model on Census-specific data. The current model works well, but it was trained on general text like Wikipedia. If we trained it on actual Census papers using examples of similar and different documents, we could boost accuracy even more. We'd also like to add new features to the frontend, like filtering by topic clusters, tagging results with confidence scores, or showing users what part of the text matched their query. All of these would make the search experience more helpful and transparent.

References & Acknowledgements

Gahman, Nicholas, and Vinayak Elangovan. ‘A Comparison of Document Similarity Algorithms’. *arXiv [Cs.CL]*, 2023, http://arxiv.org/abs/2304.013 30. arXiv.

Liu, Xiaodong, et al. ‘The Microsoft Toolkit of Multi-Task Deep Neural Networks for Natural Language Understanding’. *arXiv [Cs.CL]*, 2020, http://arxiv.org/abs/2002.07972. arXiv.

Reimers, Nils, and Iryna Gurevych. ‘Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks’. *arXiv [Cs.CL]*, 2019, http://arxiv.org/abs/1908.10084. arXiv.

Reimers, Nils, and Iryna Gurevych. 2019. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.” In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, November. Association for Computational Linguistics. https://arxiv.org/abs/1908.10084.

A special shoutout to out client mentors, Ian Minh Chieu Le and Maxwell Hope. Without their guidance and advice, our project would most definitely be in shambles with little direction.

Also a huge shoutout to the entire teaching staff throughout this Capstone project: Dr. Angela Patterson, Dr. Alan Lattimer, Alicia Arneson, and Hayden Ringer.

“We have neither given nor received unauthorized assistance on this assignment.”

[illegible] Ethan [illegible]