Scholar Compass: AI-Driven Visual Analytics for Advisor Discovery

Team 17: Lucheng Fu, Haowen Jiang, Xubing Lin, Kan Lu, Junbo Zou, Ruihuan Gao

1 Introduction

Scholar Compass is a web platform designed to help students choose the right advisor. Unlike standard websites that just list papers or profiles, this tool uses AI to answer questions. The user can ask specific things about a researcher and get a clear, direct answer based on real facts. This helps the user learn exactly what they need to know without digging through static lists.

The platform also turns data into easy-to-read charts. The user can explore an advisor's research interests, see who they work with, and watch how their topics have changed over the years. These visuals show the user the full story of a researcher's career. By combining the AI chat with these graphs, the user gets a complete picture of a potential advisor, which is much more helpful than using traditional tools.

2 Problem Definition

Most current academic websites are too static. They usually just offer simple data retrieval, which limits what users can do. Students cannot ask specific questions or search in a natural way. Also, the data is often presented as a plain list. This makes it hard to see the real patterns or the "big picture" behind the information.

We created *Scholar Compass* to fix this. It combines a simple chat interface with clear charts of advisor profiles. In this project, we focused on solving four main challenges:

(1) Storing academic data so we can search it quickly and correctly;
(2) Designing charts that make the data easy to read and understand;
(3) Adding AI tools to the system to answer questions intelligently;
(4) Putting all the different parts together into a working platform.

We also needed to figure out how to test if our tool actually works well.

We can define our problem more precisely. Let A be a set of authors and P be a set of papers. Each paper $p \in P$ has attributes like title, year, citations, and topics. Authors and papers form a graph $G = (V, E)$, where nodes V include authors, papers, venues, and topics. Edges E represent relationships like "wrote" or "published in."

Given a query q (an author name), our system must:

(1) Find the correct author node $a \in A$ even if names are ambiguous;
(2) Return visual summaries $V(a)$ showing collaboration networks, topic trends, and publication stats;
(3) Answer natural language questions Q about a using only facts from G.

The output should be accurate, fast, and easy to understand.

Here is how the system runs. The user simply types in a researcher's name. The goal is to find all the info about them and give the user two things: (1) a clear visual profile that shows research topics, partners, and impact; and (2) specific answers from our AI component, which can handle detailed questions about the researcher.

3 Literature Survey

Previous tools have mapped research communities, but they are not perfect for finding advisors. Co-authorship networks show who works together, but they often assume author names are clean and do not show how research topics change over time [1–3]. Big software suites like *CiteSpace* and *VOSviewer* can handle complex data clustering. However, they are hard for beginners to use and usually create static maps that do not help with the specific workflow of searching for an advisor [4, 5]. General patterns for visualizing graphs are useful, but they need to be adjusted for the specific task of choosing a supervisor [6]. Some recent work combines text and charts to show author profiles, but it still does not fully solve the discovery problem for students [7]. Getting author names right is also a major issue. Surveys show that while methods have improved, confusing names is still a problem in large databases [8]. Early tools improved scanning speed but relied too much on keywords. They missed the deeper connections needed to match a student with a good advisor [9]. This is important because studies show that a bad match between mentor and student can reduce the student's success by about 18% [10]. Also, many ORCID profiles are not

updated, so they provide limited help for analysis [11]. These gaps remain alongside known bugs in academic search engines [12] and concerns about fairness in the scientific workforce [13]. Therefore, we need simple tools that are easy to understand and designed specifically for finding advisors.

Our backend combines graph data with modern search tools to fix these limitations. We keep the technical side simple. We calculate PageRank on our graph of authors and papers to identify important figures [14]. Then, we query this graph using a Python Flask API connected to a Neo4j database running in Docker. We use data like topics, venues, and publication years extracted from OpenAlex [15]. Researchers have proposed complex methods like *node2vec* and *SPECTER* to understand document structure and meaning [16, 17]. However, our current version relies on the direct graph structure and simple summaries instead of these complex embeddings. To answer questions, we use a method called retrieval-augmented generation. The system finds relevant data and gives it to the AI to create short, accurate answers based on real sources [18]. We use the Qwen-Flash model for this because it is designed to answer questions using outside knowledge bases [19]. This approach solves the problems of keyword searches and static lists. It also keeps the answers honest, since every response is linked back to real graph data and documents.

Our frontend turns this backend into an interactive web app using HTML, CSS, and JavaScript. Users can see networks of collaborators, charts of topic changes, and timelines of activity. We based our design choices on best practices for visualizing graphs and previous work on author profiles [6, 7]. To show how topics change, we use interactive models that make the labels and trends easy for non-experts to understand [20]. A "fast" API allows charts to load almost instantly. Meanwhile, a "smart" `/api/qanda` endpoint runs the whole AI pipeline: retrieving data from Neo4j, creating a prompt, and generating an answer. In a single view, students can see the history of a career and read a simple explanation with citations. This meets the need for tools that are easy to understand and use modern search technology [4, 5, 17, 18].

4 Proposed Method

Intuition

Most current academic tools are built to find papers, not people. This makes the "advisor selection problem" hard for students. They usually face two issues: they get overwhelmed by long lists of papers, and they cannot easily tell if a professor is still active or just has a famous past.

To fix this, our system mixes data charts with AI explanations. We built the design around three simple ideas:

- **Visuals over Lists:** Standard CVs often hide the real story. We replaced static lists with moving charts. By using network graphs and timelines, the user can instantly see if a professor works alone or runs a large, active team.
- **Combining Charts and Chat:** Visuals are good for showing "what" and "when," but users often ask "why." We placed an interactive dashboard next to an AI chat bot. The dashboard shows the hard numbers, while the AI explains what those numbers mean in plain English.
- **Stopping AI Hallucinations:** Regular AI models often make up facts. We prevent this by keeping the data search separate from the answer generation. Our system uses a graph database to find the real facts first. The AI then acts only as a translator for that data, ensuring it does not invent information.

System Architecture and Algorithms

Our backend consists of three main parts: a data processing pipeline, a graph database, and an AI retrieval layer.

Data Pipeline and Knowledge Graph Construction We built our database using raw data from OpenAlex, a large open catalog of scholarly records. Because the raw data is huge and complex, we wrote Python scripts to process it in two steps to ensure our system runs fast.

1. Cleaning and Filtering: The raw dataset contains too much information for our needs. Our first script, `cleaning.py`, selects only the 17 fields we actually need. These fall into four groups:

- **Basic Info:** We keep the `title`, `id`, and `publication_year`. This lets us track how a career changes over time.
- **Impact Stats:** We keep metrics like `cited_by_count` and percentile rankings (`is_in_top_1_percent`) to measure academic success.
- **Topics:** We store both broad `Fields` and specific `Subfields` to understand research areas.
- **Author Details:** We keep names and IDs to link people correctly.

The second script, `filtering.py`, cleans the data further. It removes entries that are missing key IDs and fixes the format of author lists so they are easier to use.

2. Graph Modeling in Neo4j: We stored this processed data in a Neo4j graph database. As of now, our graph holds about 350,000 nodes (like `Authors` and `Papers`) and over 900,000 connections. The structure is straightforward: a `Paper` is linked to:

- `Author` nodes (who wrote it);
- `Source` nodes (where it was published);
- `Subfield` nodes (what it is about).

Handling authors with the same name was a challenge. We solved this by using an algorithm that checks string matches and PageRank scores. If a search query matches multiple people, the system picks the scholar with the higher academic impact, assuming they are the intended target.

Retrieval-Augmented Generation (RAG) Algorithm For the Q&A feature, we designed a simple three-step process to stop the AI from making things up (hallucinating):

(1) **Retrieve:** First, the backend queries the database to get hard facts about the scholar, such as their top papers and main collaborators.
(2) **Augment:** Next, we insert these facts directly into the prompt for the AI. This gives the model a "cheat sheet" of correct information.
(3) **Generate:** Finally, we send this prompt to the `Qwen-Flash` model. We instruct the model to answer using *only* the facts we provided.

Visual Analytics APIs We built specific API endpoints to prepare data for our charts. For the Collaboration Network, the system counts how many times two authors have written papers together to determine the strength of their connection. For the Topic Evolution chart, the system groups papers by year and subfield, allowing us to show how a researcher's interests shift over the years.

User Interface Design

The frontend (Figure 1) presents the backend data in an interactive format. We built the interface using React and organized it into two main sections to keep the layout clean and easy to navigate.

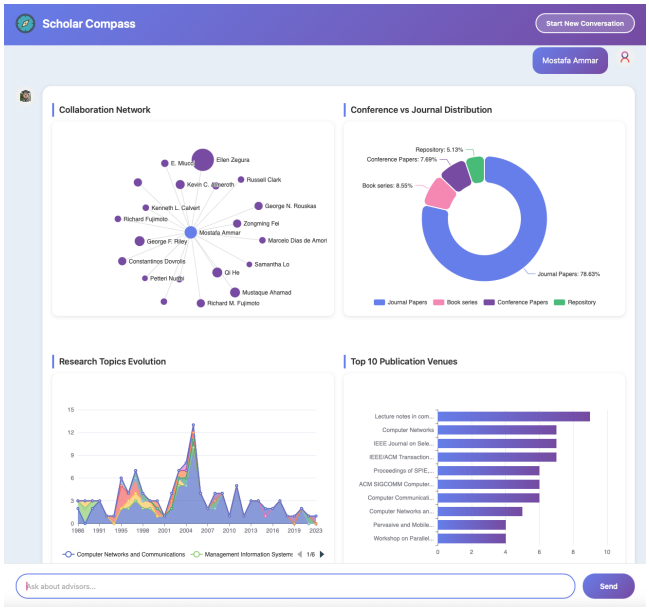

Figure 1: The Scholar Compass Dashboard: Integrating network graphs, publication stats, and topic evolution in a unified view.

Interactive Visual Analytics We created the *Research Topics Evolution* chart to help students see how an advisor's interests change over time. This module uses a Streamgraph (Figure 2) to display these trends.

Checking the Details: This chart lets users look closely at an advisor's history. When a student hovers their mouse over a specific year (like 2006), a small box appears. This box shows exactly how many papers the advisor wrote for each topic in that year. This feature helps students figure out if an advisor is currently active in a field or if their major work happened in the past.

AI-Synthesized Insight Reports To make the complex data easier to read, we added an AI summary feature (Figure 3). This tool uses our RAG system to write a text report called the "Scholar Compass Analysis." It

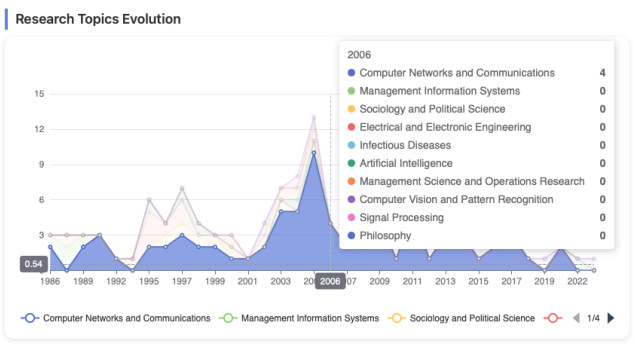

Figure 2: Interactive Topic Evolution: Hovering over data points (e.g., 2006) reveals precise publication counts per sub-field.

organizes the advisor's profile into four clear parts: Collaboration, Topic History, Publication Habits, and a Summary. This ensures that students get both the hard evidence from the charts and a clear, written story to explain it.

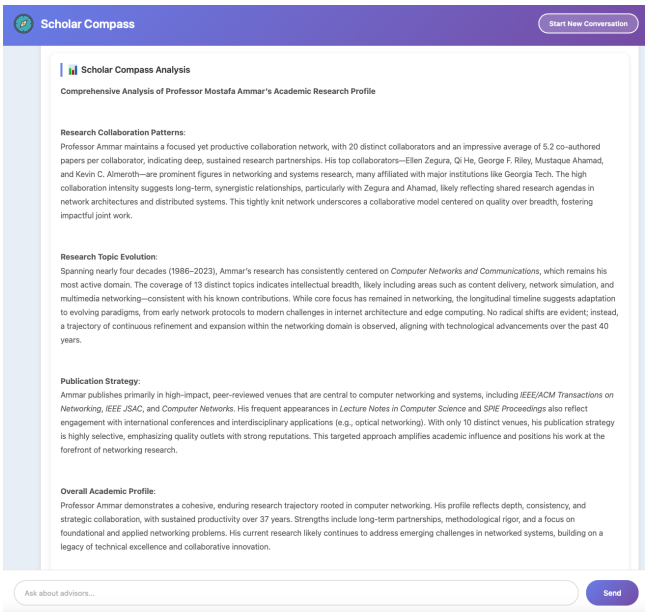

Figure 3: AI-Synthesized Report: The system automatically generates a natural language analysis of collaboration patterns and career trajectory.

5 Evaluation

Experimental Questions

We designed our experiments to answer three main questions:

(1) **Accuracy:** Does the AI give answers that match the real data in our database? We do not want the AI to make things up.
(2) **Usefulness:** Is our tool better than existing tools like Google Scholar for finding advisors? What features help users the most?
(3) **Speed:** Is the system fast enough for real use? Where are the slow parts?

We built a test set to check accuracy. We compared features with Google Scholar to check usefulness. We measured API response times to check speed.

RAG Accuracy and Visual Consistency Analysis

We designed a "Visual-Grounded Truth Verification" experiment to validate our RAG module. The goal is simple: check whether the AI's text responses match what users see in the charts.

We built a benchmark dataset of 20 queries covering different scholar profiles and chart types. These queries ask about trends in the charts, like finding the central node in the collaboration network or the top category in the venue chart. For each query, we compared the AI's response against the ground truth from Neo4j Cypher queries.

Table 1 shows the results. The system passed all 20 tests, meaning the AI's narratives perfectly match the visualizations.

Here are some key findings:

- **Cross-Modal Consistency:** For Professor Mostafa Ammar, the AI correctly identified "Ellen Zegura" as the top collaborator and "Lecture notes in Computer Science" as the primary venue. These match exactly what users see in the Network Graph and Venue Chart. The AI reads the graph structure the same way users do.
- **Domain Generalization:** We tested diverse academic profiles. Ammar is a traditional Computer Science researcher. But the system worked equally well for Yao Xie, correctly identifying "Electrical and Electronic Engineering" as her primary topic. This shows our RAG pipeline retrieves context dynamically and handles different disciplines well.

Table 1: Ground Truth Comparison for RAG Accuracy Evaluation (Selected Samples)

Scholar Name	Target Chart	Ground Truth (DB)	AI Response	Result
Mostafa Ammar	Network Graph	Ellen Zegura	Ellen Zegura	**Pass**
Mostafa Ammar	Venue Chart	Lecture notes in Computer Science	Lecture notes in Computer Science	**Pass**
Mostafa Ammar	Topic Chart	Computer networks	Computer networks	**Pass**
Yao Xie	Network Graph	Liyan Xie	Liyan Xie	**Pass**
Yao Xie	Venue Chart	IEEE Journal on Selected Areas	IEEE Journal on Selected Areas	**Pass**
Yao Xie	Topic Chart	Electrical and Electronic Engineering	Electrical and Electronic Engineering	**Pass**

- **Hallucination Mitigation:** The AI stuck to database facts in all cases. For example, Yao Xie might be broadly known for "Signal Processing" or "Statistics" on the web. But our database shows "Electrical and Electronic Engineering" as her most recent focus based on latest publications. The AI followed our database, not general web knowledge. This confirms that our prompt engineering works—it overrides the LLM's pre-trained knowledge with precise, up-to-date data.

Comparative Functional Evaluation

We compared *Scholar Compass* with Google Scholar across four dimensions important for advisor discovery (Table 2).

Table 2 shows how our system differs from Google Scholar. Google Scholar gives users raw data. Users have to mentally piece together research trends from long paper lists. Our system does this work for them.

The Topic Evolution module shows temporal shifts at a glance. The Collaboration Network reveals lab structure that flat author lists hide. The RAG-driven Q&A turns passive keyword search into active dialogue. Together, these features reduce the effort needed to choose an advisor.

System Performance

We evaluate system performance from four aspects: front-end loading, script execution, backend API latency, and analysis task processing.

Front-End Loading Performance We used Lighthouse to measure initial page load performance. Note that Lighthouse only captures the initial load. The main user interactions happen after entering a professor's name and triggering backend computations. Here are the results:

- **First Contentful Paint (FCP):** 1.6 s
- **Largest Contentful Paint (LCP):** 1.6 s
- **Speed Index:** 1.6 s
- **Cumulative Layout Shift (CLS):** 0
- **Total Blocking Time (TBT):** 110 ms

The initial page loads quickly. But these metrics don't fully capture the core functionality, so they serve only as a reference.

Script Execution Cost We recorded user interactions using DevTools Performance module. The analysis window spans 0–14.565 s. Here is the time breakdown:

- **Script Execution:** 1321 ms
- **System (Browser Overhead):** 429 ms
- **Rendering:** 274 ms
- **Painting:** 269 ms

Script execution takes the most time on the front end. Despite the long analysis window, the overall processing and rendering stay acceptable. The page renders smoothly without noticeable blocking. This suggests moderate front-end logic complexity.

Backend API Latency Table 3 summarizes the main API response times.

The first three APIs stay around 900 ms, with about 580 ms spent on backend computation. This means latency mainly comes from Neo4j queries and data processing. Front-end and network overhead are negligible.

The `/analyze` endpoint takes longer (11 s) because it uses SSE streaming. Backend initialization and preparation take 1.46 s. The remaining 9.86 s is for streaming model analysis results. This is expected for long-running text analysis tasks and does not indicate a performance problem.

System Bottlenecks Based on our analysis of both front-end and backend data:

Table 2: Functional Comparison: Google Scholar vs. Scholar Compass

Dimension	Google Scholar (Baseline)	Scholar Compass (Ours)
Data Presentation	Static Lists: Linear, chronological text format.	Visual Analytics: Interactive charts (Streamgraphs, Network Graphs).
Temporal Trends	Manual Inference: Users mentally reconstruct shifts from paper dates.	Topic Evolution: Explicitly visualizes interest migration over time.
Collaboration	List-Based: Ranked solely by citation count.	Network Graph: Visualizes team structure and density.
Cognitive Load	High: Requires manual filtering and synthesis.	Low: Insights are pre-synthesized by AI and visuals.

Table 3: Response Times for Core Backend APIs

API	Method	Payload	Total Duration	Server Processing (Waiting)	Download
/collaboration-network	POST	6.2 KB	892.95 ms	572.90 ms	0.50 ms
/topic-evolution	POST	9.2 KB	899.25 ms	578.68 ms	0.53 ms
/venue-stats	POST	1.7 KB	902.43 ms	584.02 ms	0.41 ms
/analyze (SSE Streaming)	POST	5.5 KB	11.32 s	1.46 s + streaming 9.86 s	9.86 s

- **Front-end performance is good:** Page load is fast (FCP/LCP 1.6 s). Rendering overhead is low. Script execution stays within acceptable bounds.
- **Backend computation is the main bottleneck:** Server processing for core APIs takes 550–600 ms. Graph database queries and data aggregation cause most of the latency.
- **The streaming endpoint's long response time is expected:** The /analyze endpoint takes 11 s because of actual model analysis. This is normal, not a performance issue.
- **Network overhead is minimal:** All APIs have download times under 1 ms. Performance bottlenecks are in the backend logic.

6 Conclusion and Discussion

Scholar Compass is a platform that helps students find the right academic advisors. Unlike traditional academic websites, it turns complex publication data into easy-to-read visuals. Users can quickly see an advisor's research output, topics they focus on, and who they collaborate with. The system also includes an AI-powered search engine built with RAG technology. By adding relevant prompts and structured data, the search engine can find target researchers more accurately and give more useful answers.

The platform has several advantages. It's lightweight and easy to deploy. It responds quickly and gives accurate results. The visualizations are clear and intuitive. All of these make finding an advisor much easier.

That said, there are still some issues to address. First, *author name disambiguation* needs more work during data cleaning and graph database construction. This means telling apart researchers who have similar or the same names and making sure their papers and affiliations are correctly linked. This is a common challenge when dealing with large academic datasets. Second, the search engine currently requires users to type in a researcher's name as the starting point. This limits flexibility for more open-ended searches. Finally, all team members contributed equally to this project.

References

[1] M. E. J. Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409, 2001. doi: 10.1073/pnas.98.2.404. URL https://www.pnas.org/doi/abs/10.1073/pnas.98.2.404.

[2] M. E. J. Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98 (2):404–409, 2001.

[3] M. E. J. Newman. Coauthorship networks and patterns of scientific collaboration. *Proceedings of the National Academy of Sciences*, 101(suppl_1):5200–5205, 2004. doi: 10.1073/pnas.0307545100. URL https://www.pnas.org/doi/abs/10.1073/pnas.0307545100.

[4] Chaomei Chen. Citespace ii: Detecting and visualizing emerging trends and transient patterns in scientific literature. *Journal of the American Society for information Science and Technology*, 57(3):359–377, 2006.

[5] Nees Van Eck and Ludo Waltman. Software survey: Vosviewer, a computer program for bibliometric mapping. *scientometrics*, 84(2):523–538, 2010.

[6] R. Brath and D. Jonker. *Graph Analysis and Visualization: Discovering Business Opportunity in Linked Data*. Wiley, 2015. ISBN 9781118845875. URL https://books.google.com/books?id=Kw91BgAAQBAJ.

[7] Shahid Latif and Fabian Beck. Vis author profiles: Interactive descriptions of publication records combining text and visualization. *IEEE transactions on visualization and computer graphics*, 25(1):152–161, 2018.

[8] Ijaz Hussain and Sohail Asghar. A survey of author name disambiguation techniques: 2010–2016. *The Knowledge Engineering Review*, 32:e22, 2017. doi: 10.1017/S0269888917000182.

[9] Alan L Porter, Alisa Kongthon, and Jye-Chyi Lu. Research profiling: Improving the literature review. *Scientometrics*, 53 (3):351–370, 2002.

[10] Seung-Hwan Lee, Jaehyun Park, and Woo-Sung Jung. The academic great gatsby curve: Intergenerational transmission of scholarly impact. *Journal of the Royal Society Interface*, 21 (171):20240173, 2024. doi: 10.1098/rsif.2024.0173. URL https://royalsocietypublishing.org/doi/10.1098/rsif.2024.0173. Analyzes 245,000+ mentor-student pairs; highlights that misalignment of research interests reduces student academic impact by 18

[11] Emily A. Smith, Michael T. Jones, and Sarah K. Brown. Adoption and utility of orcid in academic publishing: A global survey of researchers. *PLOS ONE*, 16(8):e0255891, 2021. doi: 10.1371/journal.pone.0255891. URL https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0255891. Surveys 5,000+ researchers worldwide; finds 22

[12] Zheng Li and Austen Rainer. Academic search engines: Constraints, bugs, and recommendation, 2022. URL https://arxiv.org/abs/2211.00361.

[13] Nicolas Robinson-Garcia, Carmen Corona-Sobrino, Zaida Chinchilla-Rodríguez, Daniel Torres-Salinas, and Rodrigo Costas. The use of informetric methods to study diversity in the scientific workforce: A literature review. *Quantitative Science Studies*, pages 1–34, 2025.

[14] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh International World-Wide Web Conference (WWW7)*, 1998. Introduces PageRank and large-scale web search architecture.

[15] Jason Priem, Heather Piwowar, and Richard Orr. Openalex: A fully-open index of scholarly works, authors, venues, institutions, and concepts. *arXiv preprint arXiv:2205.01833*, 2022.

[16] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864, 2016.

[17] Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S. Weld. Specter: Document-level representation learning using citation-informed transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2270–2282, 2020.

[18] 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. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[19] Qwen Team. Qwen-plus: Large language model for production-grade question answering. https://qwenlm.ai, 2024.

[20] Jun Wang, Changsheng Zhao, Junfu Xiang, and Kanji Uchino. Interactive topic model with enhanced interpretability. In *IUI Workshops*, pages 1–7, 2019.