AC-2 Data Visualizer for Topic Models

Final Report

Course: 4805-03
Semester: Spring 2023
Professor Perry
Date 04/29/2023

Team
Project Owner: Arthur Choi

Team Members:
Kalp Patel, Aditya Shiroya, Michael Cooper, James Roll

Website: https://chillsterz3434.github.io/DaViToMo/

Table of Contents

1.0 Introduction

In today's information era, organizing and examining huge document collections has become an essential task in a variety of areas. Manual organizing, however, becomes less and less efficient as document collections continue to expand in size and complexity. In order to make content research easier, automated methods for detecting and displaying a collection's structure have become crucial. Use of topic models, which are statistical models that describe the latent subject structure of text, is one such method. Our project, Data Visualizer for Topic Models, aims to automatically identify the topics that are mentioned in Wikipedia articles using just the words that are found in such articles. Our system aims to evaluate, investigate, and display a topic model that has been learnt from a dataset, such as Wikipedia, and to integrate a recommendation engine into the interactive visualization. Using a similarity matrix or heat map, our recommendation engine will show viewers content that is related to the item they are presently reading. Without the requirement for manual organizing or specialized knowledge, this will enable users to explore and comprehend the hidden topics included in any given text collection. We think that our approach will be a crucial tool for scholars, analysts, and anybody else interested in understanding complex text databases. We think that by making massive document collections simpler to view and analyze, we might encourage new discoveries and insights in a wide variety of fields.

1.1 Overview

The aim of this project is to create a web-based system for organizing, summarizing, and displaying a large collection of documents, such as Wikipedia articles. The underlying thematic structure of the articles will be found using topic modeling techniques, which will then represent each text as a collection of topics. The system gives users a simple interface to navigate between high-level summaries (the "themes") and the specific articles themselves as they browse the corpus. The visualization will also include a recommendation engine that will propose articles with related themes and display how similar the suggested material is to the item being viewed right now. The system will ultimately help people explore and understand vast amounts of unstructured documents.

1.2 Abstract

Data Visualizer for topic models (DaViToMo) is a robust and user-friendly web-based system that enables users to quickly discover and navigate the topical structure of large text datasets like Wikipedia articles. Our system is built on Flask for our Python server and Express for our Node.js server to discover the subject organization of articles and represent each text as a collection of themes. We used React.js to visualize the structure of the data and create an interactive UI that allows the user to interact with the topic model. Companies, such as google, routinely use machine learning models such as topic models to analyze large bodies of unstructured text. The system also includes a recommendation engine that allows users to discover new articles based on the topical structure found by the model. Whether you're a researcher or a curious reader, Data Visualizer for Topic Models is the ideal tool for navigating and understanding massive quantities of unstructured data.

1.3 Project Goals

The main goal of this project is to create a web-based system that utilizes topic modeling to automatically identify and display the thematic organization of a sizable corpus of text, like Wikipedia articles. The system will offer a user-friendly interface that enables users to move between high-level found summaries (the "themes") and individual texts to explore and browse the corpus. The system will also have a recommendation engine that suggests articles with related themes and shows how similar different pieces of information are to one another. This research will make a contribution to the field of information retrieval and data mining by offering an effective tool for arranging and summarizing large amounts of unstructured data.

1.4 Definitions and Acronyms

Definitions:
Topic Model: A statistical model that represents the latent topical structure of text.
Wikipedia Dataset: A collection of text articles from the website Wikipedia.
Recommendation Engine: A software program that suggests relevant content to users based on their behavior and preferences.
Similarity Matrix: A matrix that measures the similarity between two or more items.
Corpus: a large collection of written or spoken texts, treated as a whole
Graph: A visual representation of data or information.
Heat Map: A visual representation of data that uses color to show different levels of intensity.
Web-based System: A server-based software system that is accessed via a web browser.
Acronyms:
API: Application Programming Interface.
AI: Artificial Intelligence
HTML: Hypertext Markup Language.
CSS: Cascading Style Sheets.
JS: JavaScript.
UI: User Interface.
UX: User Experience.
DB: Database.
HTTP: Hypertext Transfer Protocol.
HTTPS: Hypertext Transfer Protocol Secure.
SQL: Structured Query Language.
CSV: Comma-Separated Values.

1.5 Assumptions

- The system will be designed and developed for use with English language text.
- Client/user has an active Internet Connection or has access to one to view the website.
- The topic model will be generated using Latent Dirichlet Allocation (LDA) algorithm.
- The user interacting with the system will have a basic understanding of topic modeling.
- The system will be used in a web-based environment.
- The system will be accessed through a standard web browser.
- The system will be developed using open-source technologies.
- The system will be tested and evaluated on Wikipedia dataset.

1.6 Challenges

One of our most difficult challenges we had was choosing the right framework to use for the backend. Although we initially intended to use Spring Boot, it became apparent from extensive testing and research that Node.js would be a better fit for our requirements. We had to redo a significant amount of our code as well as the backends' whole architecture because of this choice. But ultimately, this modification helped us create a system that was more effective and scalable.

We had trouble allowing access to the cloud database for testing reasons to every team member. We were unable to work concurrently on the same database as a result, which led to delays and communication problems. By developing a separate testing environment that all team members could access, we were able to finally overcome this problem.

Integration of the recommendation engine caused another challenge for us. Although the topic modeling and visualization parts of the system were simple, adding the recommendation engine required more investigation and development. To make sure that the recommendations were accurate and relevant to the user's needs, we had to test out numerous algorithms and methodologies.

It was challenging for us to create an interface that was simple and easy to use. In addition to making sure the interface was consistent and visually appealing, we had to consider several variables, including user preferences, needs, and accessibility considerations.

2.0 Architectural Design

A client-server architecture with a web-based interface is how the DaViToMo architecture is built. The four key parts of the system are collecting data, topic modeling, recommendation engine, and visualization.

2.1 Environment

The website was successfully created and tested on local PCs throughout the development period. A Kubernetes cluster that was hosted on a cloud-based server was used to deploy the system during the production phase after it had been containerized using Docker. This made system scalability and handling possible without any difficulty. We choose to continue utilizing MongoDB for database administration because of its adaptability and scalability. To provide high availability and data permanence, a cloud-based service provider hosted the MongoDB database. To avoid data loss in the case of a system breakdown, we also built routine database backups. Lastly, we used containerization and Git version control to make the system easy to maintain and upgrade. This made it possible to deploy upgrades and new features to the system quickly and without any downtime.

2.2 User Characteristics

This website is intended for students or researchers who need extra resources for their research projects. Using probabilistic topic modeling, the website will suggest related content in response to user search queries. We will serve an extensive range of users, including those who are new to the issue and those who already have a solid understanding, as the website intends to educate people about topic modeling. We will include suggestion boxes that describe how the results are created in order to make the website more user-friendly.

2.3 System

The four primary components that make up the system architecture are as follows:

Data collection: This component is in responsible for gathering text information from a dataset, like Wikipedia. The preprocessing of the acquired data includes tokenization, stop-word elimination, and stemming.

Topic Modeling: Using the Latent Dirichlet Allocation (LDA) technique, this component creates a topic model from the preprocessed text data. This component produces a list of topics and the keywords that go with each topic.

Recommendation Engine: Based on the user's current article view, this component creates suggestions. The engine employs a heat map or similarity matrix to display how closely the proposed material matches the article the user is currently reading.

Visualization: This component is in responsible for clearly and easily displaying the topic model and suggestions that were created. The component offers interactive visualizations including word clouds and heat maps.

3.0 Functional Requirements

- The system has the capability to create topic models from a given dataset.
- Users may explore and view the topic models using the system's interactive via the web interface.
- The system includes a recommendation engine that proposes articles with comparable topics and scores how similar the content is to the one being viewed.
- The system contains a number of visualizations to help with topic model understanding, including word clouds and heat maps.
- The system's search and visualization features are quick and effective, and it is built to manage huge data sets.
- Users with various levels of technical expertise should be able to utilize the system without any trouble.
- To allow further updates and improvements to the topic modeling method and web interface, the system is scalable.
- The system is fully functional and satisfies all requirements.

4.0 External Interface Requirements

4.1 User Interface Requirements

The primary objective of the user interface is to give users a simple, user-friendly website where they can search for articles and browse related articles using a topic model. Users can use keywords relating to their research topic or the article's name to search for it. To help viewers better understand how the connected articles relate to one another, the related articles is presented in a visually appealing fashion, like a graph or heat map. The user interface also seeks to educate users about topic models by offering details on the likelihood values and the justifications for why particular articles appear to be linked. The interface is made to be user-friendly, with easy-to-follow instructions and straightforward navigation.

4.2 Hardware Interface Requirements

- The system is web-based and usable from any device with an internet connection.
- Modern desktop and mobile browsers will be able to access the website with no problems.
- For accessing the system, no particular hardware requirements are necessary.

4.3 Software Interface Requirements

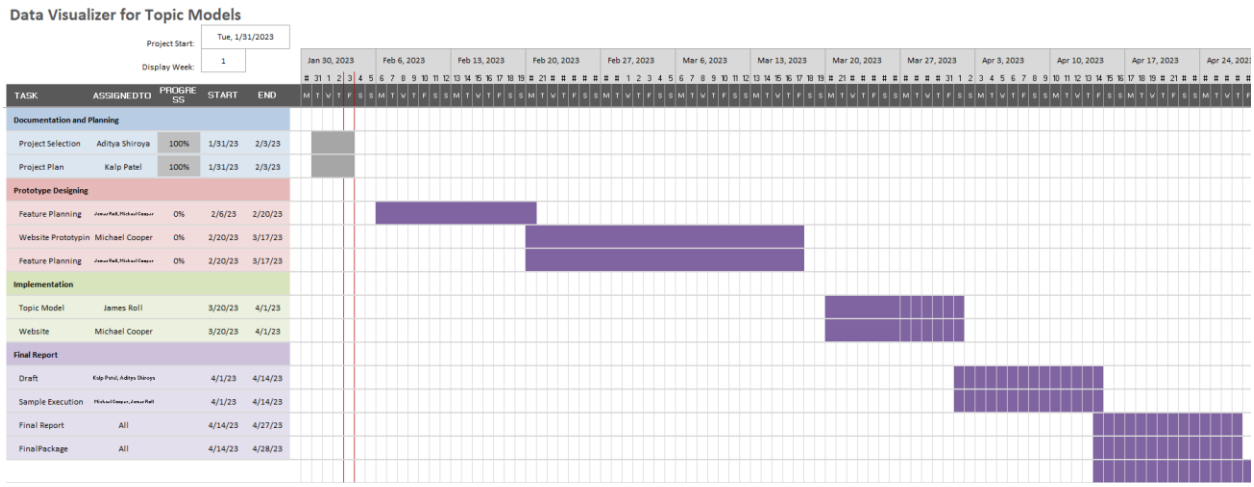
The system makes use of the Java Spring Framework, Maven, Lombok API, and Tomcat web services along with a REST API. These tools are used to build a connection between the back-end topic model system and the web application. In the future, other software interfaces could be implemented as necessary.

5.0 Project Plan

The project was completed in accordance with the following plan:

1. Planning and Research Phase:
    - Examined the best techniques and tools for creating a data visualizer for topic models.
    - Clearly stated project objectives and specifications.
    - A project schedule and resource allocation were developed.
2. Design and Development Phase:
    - Created the system's architecture.
    - React and Node.js were used to create the system's front end and back end, respectively.
    - Added search and filter capability as well as a recommendation engine.
    - Developed visualizations to help with topic model understanding, including word clouds and heat maps.
    - Performed thorough testing and debugging.
3. Deployment Phase:
    - Using Docker containers, the application was deployed on a Kubernetes cluster.
    - A separate testing environment was created for testing.
4. Maintenance and Support Phase:
    - Continued to monitor the system for errors and make required adjustments.
    - Continued to persist in keeping the system's scalability and update-compatibility.

6.0 Gantt Chart



7.0 Version Control Discussion

We utilized Git as our version control system for our project, and GitHub served as the hosting platform for our repository. The primary components of our version control strategy are as follows:

1. Repository Creation: At the beginning of the project, we set up the GitHub repository and enlisted each team member as a contributor. The repository owner, who manages access and incorporates code changes into the main branch, was chosen as one team member.
2. Branching: Each project feature or component was given its own branch, allowing each team member to focus on their individual job without interfering with the work of the others. We were also able to distinguish between issues and conflicts because to this method, which made it simpler to maintain the codebase.
3. Merging: We merged the code alterations from each branch back into the main branch after thorough testing and review. During the merge process, we resolved any conflicts that arose while making sure the code was functional and consistent.
4. Regular Updates: On a regular basis we committed code changes to the repository, and the commit messages noted significant revisions. This method gave us a thorough history of code modifications and made it easier for us to monitor the project's development.

8.0 Technical Development

We had several technical challenges when creating our project, and we added several features to improve its usability. We used React for the front end and Node.js for the back end. Data storage and retrieval are handled via an integrated MongoDB database in our system.

Using the provided dataset, we applied the Latent Dirichlet Allocation (LDA) method to generate topic models. We were able to assign probability distributions to each word in the text corpus and identify topics within the dataset using this technique. We also included a recommendation engine that proposes articles that are similar to the one being seen. To determine how similar the suggested articles are to the one being viewed, we employed cosine similarity.

Designing a simple and user-friendly interface was one of the biggest technological challenges we encountered. To make sure that the interface was simple to use for users with different levels of technical skill, we carried out a thorough investigation into user needs, preferences, and accessibility requirements. Additionally, we had to make sure that the interface was aesthetically pleasing and consistent.

We adhered to a version management plan that involved setting up branches for each project feature or component and a GitHub repository. As a result, each team member was able to fulfill their assigned tasks without interfering with anybody else. Modifications from each branch were then carefully tested and evaluated before being merged back into the main branch. We regularly pushed updates to the repository and noted important changes in the commit reports.

Overall, the technological development of the project required significant teamwork and communication. To make sure the system was operating successfully and efficiently, we needed to reconsider our decisions frequently and make changes. We are proud with what we were able to accomplish despite a few challenges along the way.

9.0 Project Development Narrative

Topic models are a form of statistical modeling that is used to discover hidden topics or themes within large collections of text data. They are widely used in natural language processing, information retrieval, and machine learning applications.

The goal of a topic model is to identify the underlying topics that are present in a corpus of documents, and to represent each document as a distribution over these topics. In other words, a topic model can be thought of as a way of organizing a large collection of text data into meaningful categories.

There are several different types of topic models, but one of the most popular is the Latent Dirichlet Allocation (LDA) model. The LDA model is a generative probabilistic model that assumes that each document is a mixture of a small number of topics, and that each word in the document is generated from one of these topics.

Topic models have many applications in a variety of fields. In the field of natural language processing, topic models can be used for text classification, sentiment analysis, and summarization. In the field of information retrieval, they can be used for document clustering and search. In the field of machine learning, they can be used for feature selection and dimensionality reduction.

Our topic model is learned from Wikipedia's large data set of articles. The user inputs an article, and our topic model searches through that article and downloads the linked articles in the original article. For example, if a user is reading an article about Artificial Intelligence, our model will recommend similar articles related to AI, such as Machine Learning, Computer Vision, and Natural Language Processing, then the user can pick a topic to go to that topic's page. On that page, the top 20 words in that article will be shown, as well as the rest of the topics. The main section of that page shows the top 10 related articles. The user can select an article to go to the next page that shows the article's topics and related articles as well as that document's text.

Our team began the development process by thoroughly researching the top technologies and tools for our data visualization project. To choose the best frameworks, libraries, and programming languages for our project, we reviewed a variety of options over several weeks. We ultimately chose to use the React framework for our front-end development since it allowed us to create responsive and engaging user interfaces. We chose Node.js for our back end since it is a scalable and effective framework for developing server-side applications.

Selecting the right database management system for our project was one of the most difficult decisions we had to make. After much consideration, we settled on MongoDB because it is a NoSQL database that is adaptable and scalable and can manage significant volumes of unstructured data. To produce topic models, we used the LDA algorithm.

We created a recommendation engine employing a similarity metric that determined the cosine similarity of articles based on their textual content in addition to topic modeling. The cosine similarity computation was implemented using the NPM package 'natural'.

10.0 Detailed Use Cases

The key features of the data visualizer for topic models are covered in detail in this section through use cases.

1. Topic Model Generation

User goal: Create a topic model for a certain article.

Scenario: The user needs to create a topic model for a certain article. They access the data visualizer's main page and click the "Get Started" button. They type the Wikipedia article's title into the input box and select "Run Script." The system then executes the Python script to create the article's subject model, and the output is saved in the cloud Mongo database. The subject model visualization page is reached when the user is redirected.

2. Topic Model Visualization

User goal: Examine and display the topic model that was created for a specific article.

Scenario: The user is interested in exploring and viewing the subject model created for a certain article. When they access the topic model visualization page, a heat map showing the likelihood that the top 70 terms are in a subject is displayed. Hovering over each one will reveal the actual likelihood, which has been increased by 10,000 to make it easier to read. A word cloud of the article's most important terms is also available to users, with the size of each word representing its significance. The visuals can be interacted with by the user to further investigate the subjects.

3. Topic-Based Article Recommendations

User goal: Get recommendations for articles on a particular topic.

Scenario: The user is looking for content on a particular topic. They access the topic model visualization page and select a relevant topic. The system displays a list of the top 10 articles that are relevant to that subject, along with a summary of each. Any article can be clicked to be read and related topics explored by the user.

4. Document Search and Exploration

User goal: Browse and research articles and the topics they are linked to.

Scenario: The user wants to look for a certain article and research topics that are relevant to it. They go to the documents page and use the search field to enter a search term. The algorithm displays a list of articles that match the query along with their titles and summaries. Any article can be clicked to be read and related topics explored by the user. The system runs the topic model again for the chosen article and displays the top 10 topics and articles that are most relevant to each topic. The user may interact with the visualizations to go further into the topics and articles.

11.0 Conclusion

In conclusion, our effort to create a data visualizer for topic models has been a huge success. We were successful in providing a user-friendly, scalable, and effective application while also fulfilling all of our functional needs. Our team put a lot of effort into making sure the program could manage massive datasets and swiftly deliver correct results. We had multiple challenges during the development process, but with determination and teamwork, we were able to get beyond them. To make sure that everyone was on the same page and that each member's knowledge was used to its full potential, our team closely collaborated. To ensure that the program was created easily and successfully, we also used version control properly.

APPENDICES

Appendix A: Screen Mockups

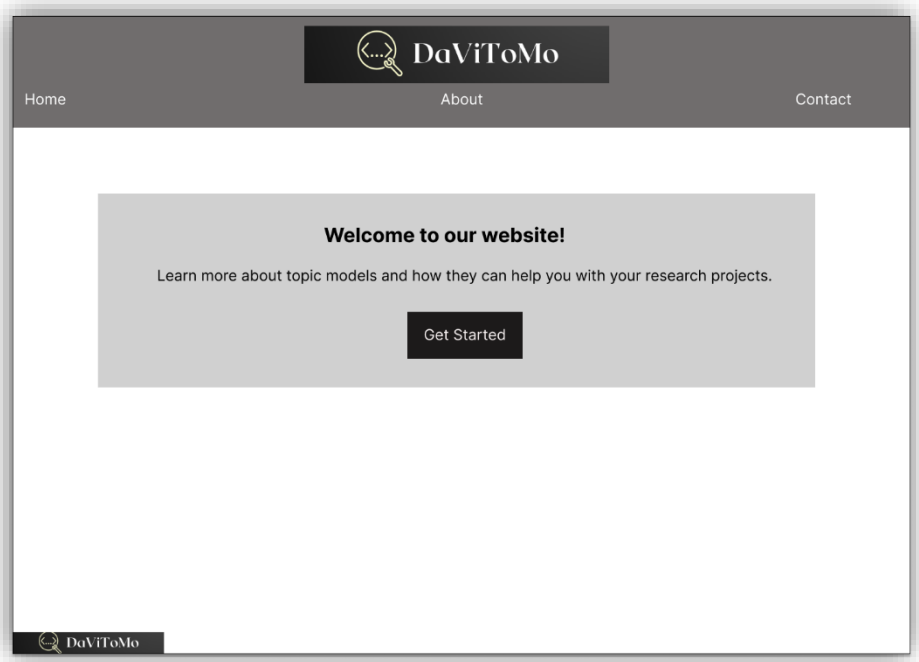


Figure 1: Homepage Mockup

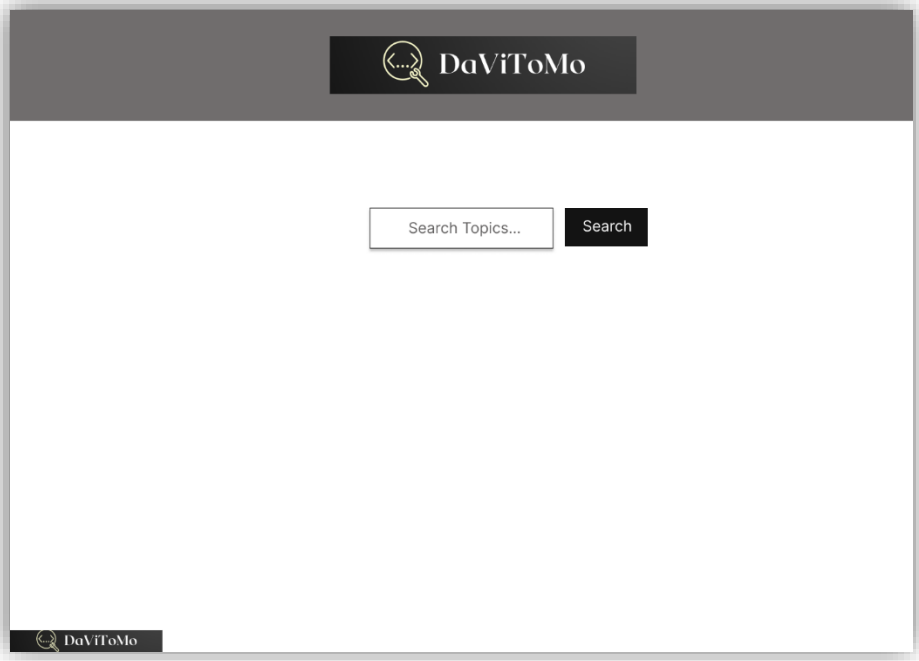


Figure 2: Topic Selection Mockup

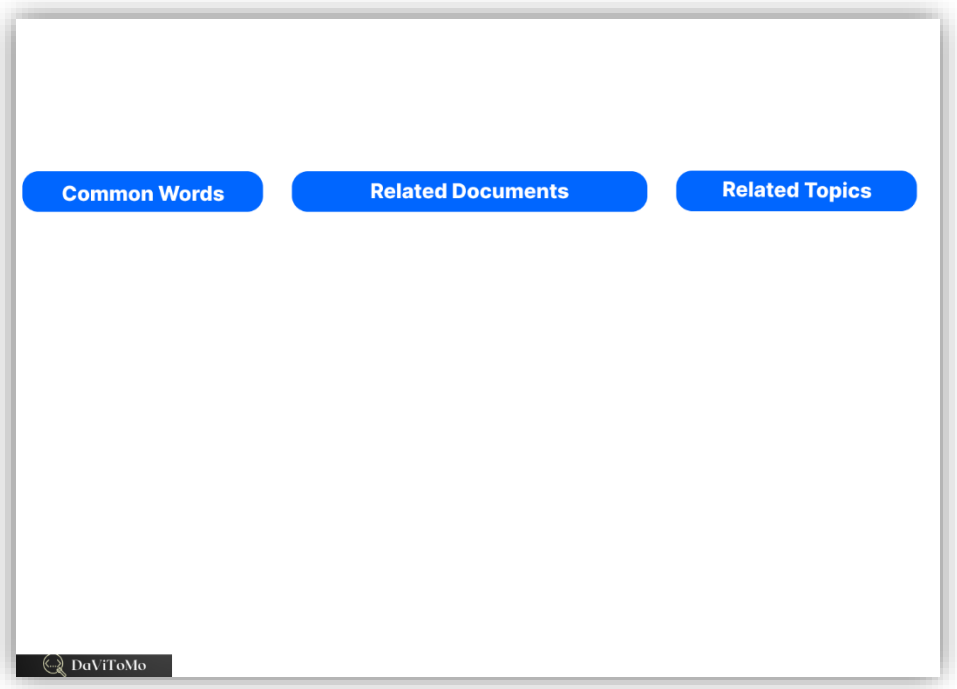


Figure 3: Topic Page Mockup

Figure 4: Document Page Mockup

Appendix B: Sample Output

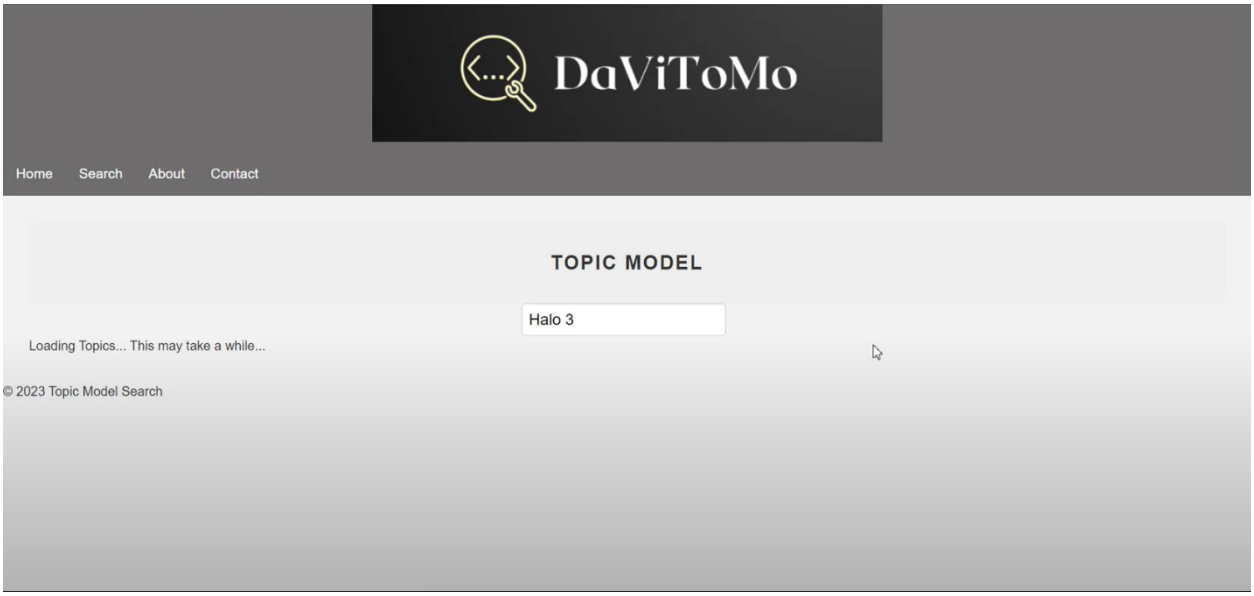


Figure 5: Topic Search

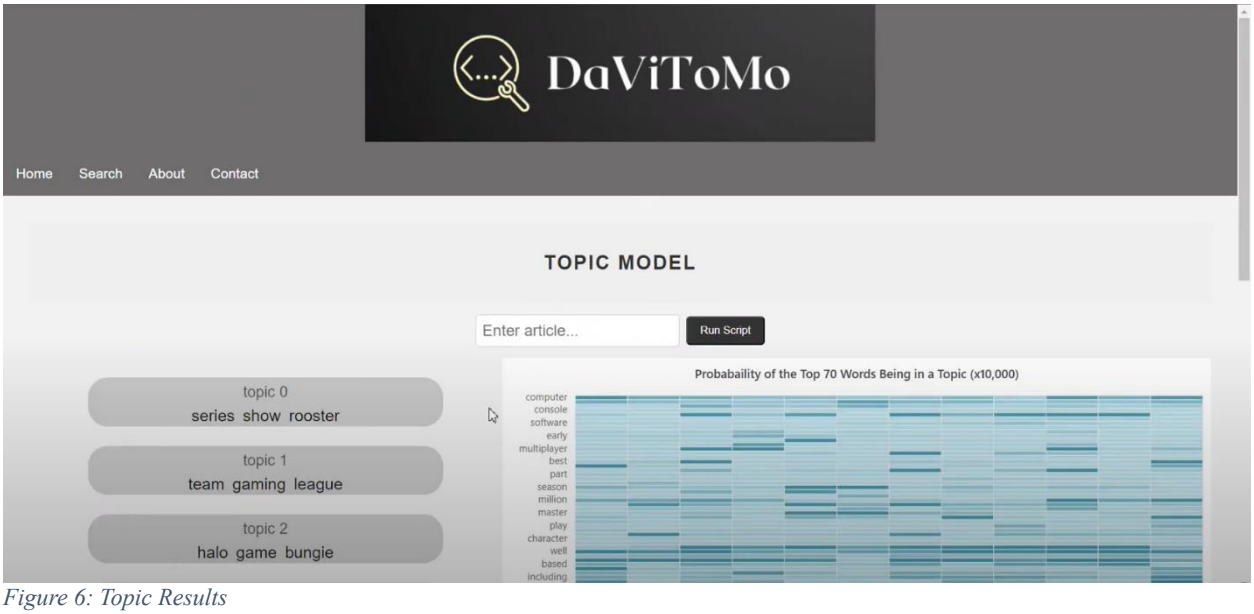


Figure 6: Topic Results

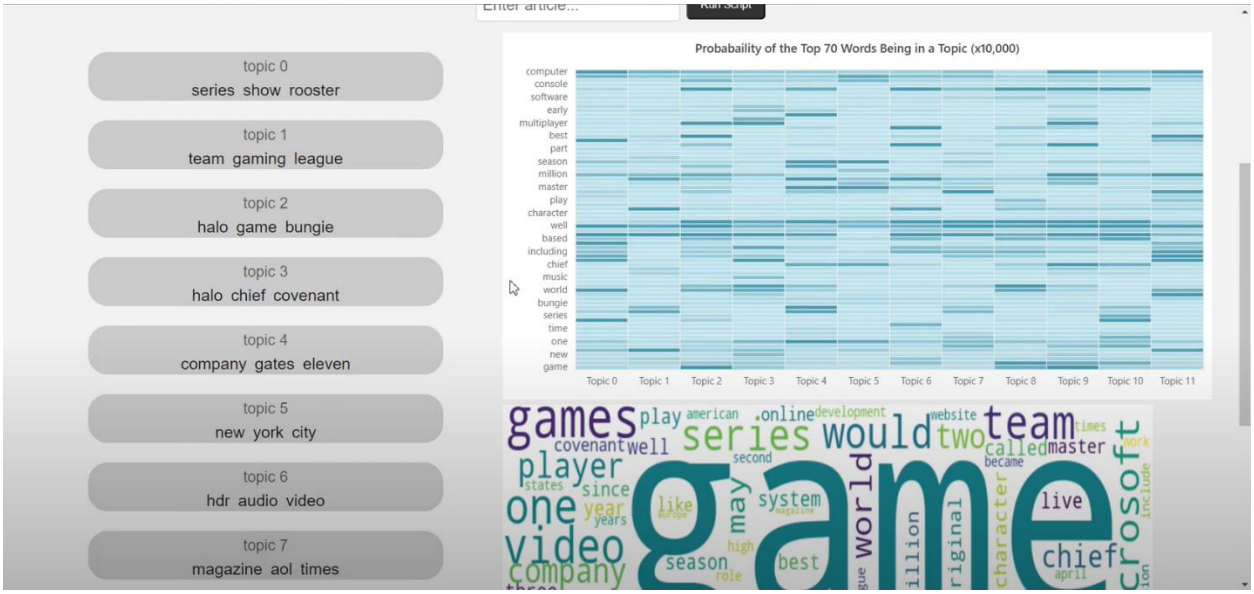


Figure 7: Heatmap

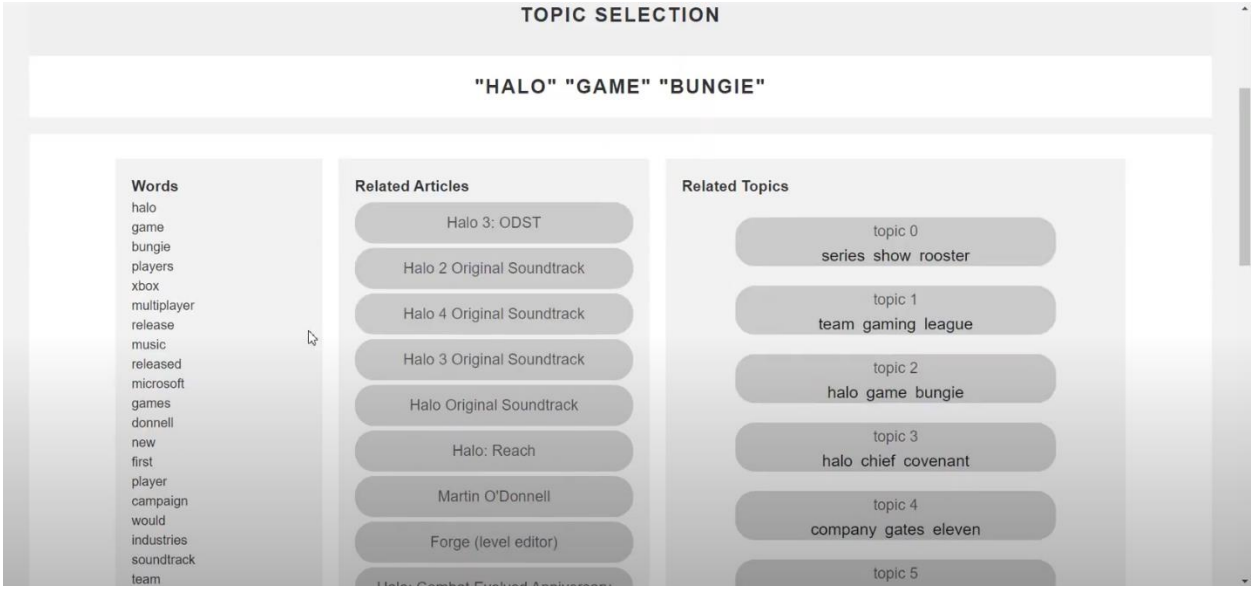


Figure 8: Related Articles