# MedEasyne: An Accessible Medical Search Engine

John Lay
University of Illinois Urbana-Champaign
Champaign, USA
johnlay2@illinois.edu

Moksh Shah
University of Illinois Urbana-Champaign
Champaign, USA
moksh2@illinois.edu

Vivek Patel
University of Illinois at Urbana-Champaign
Champaign, USA
vpate70@illinois.edu

## ABSTRACT

MedEasyne is an accessible search engine, acting as both a search engine across an aggregated collection of medical literature, and as an adaptive interpreter, leveraging language models to provide experience-appropriate explanations of the resulting literature.

**ACM Reference Format:**
John Lay, Moksh Shah, and Vivek Patel. 2018. MedEasyne: An Accessible Medical Search Engine. *ACM Trans. Graph.* 37, 4, Article 111 (August 2018), 6 pages. https://doi.org/XXXXXXX.XXXXXXX

## 1 INTRODUCTION

Accessing and understanding medical knowledge is vital for professionals and students alike. However, existing medical literature search tools often pose barriers for those with limited expertise, as they are less able to navigate complex medical literature.

Introducing MedEasyne, a medical literature search engine aimed at simplifying medical information. Our goal is to provide an easy to use medical literature search engine and enhance medical literacy for users of all levels.

In this paper, we outline our approach to developing MedEasyne, leveraging text retrieval methods to create a medical literature search engine, a connection to a Large Language Model (LLM) for further explanation, and a user-friendly interface.

## 2 MOTIVATION

Accessing and comprehending medical information is a crucial skill for both professionals and students in the field. However, existing search engines and aggregators predominantly cater to experienced individuals, leaving novices intimidated by the complexity of the results. In response to this challenge, we developed MedEasyne. MedEasyne will make medical literature searching easier for people of all skill levels as it caters to the user.

Authors' Contact Information: John Lay, University of Illinois Urbana-Champaign, Champaign, USA, johnlay2@illinois.edu; Moksh Shah, University of Illinois Urbana-Champaign, Champaign, USA, moksh2@illinois.edu; Vivek Patel, University of Illinois at Urbana-Champaign, Champaign, USA, vpate70@illinois.edu.


ACM 1557-7368/2018/8-ART111
https://doi.org/XXXXXXX.XXXXXXX

## 3 INTENDED USERS

MedEasyne is intended for all people looking to find simpler and easier to understand medical literature search results. For example, a medical professional could make use of MedEasyne for more information in their work or school. However, MedEasyne would be most effective for specifically novice medical students as they would have the least experience reading complex papers and could greater benefit from the more comprehensible results.

## 4 MAJOR FUNCTIONS

MedEasyne's major functions are (1) a medical literature search engine and (2) customizable LLM-based interpretation and term explanation of the search results. When the user opens the website, these two features are encoded in two text input boxes. The first asks the user to describe themselves, such as "I am a middle school student" and the second is a standard search box where one would input a medical topic they're interested in, such as "heart disease". There is a single search button, and it returns the top 3 results from our document database. It currently features only PubMed articles but can be arbitrarily expanded to other sources given their document collection. The titles and abstracts are displayed, and next to each result is another button "View Details". This uses the previously given user-description and the corresponding search result to prompt an LLM to provide a user-suitable explanation of the abstract as well as define any "difficult" words. For example, if the user claims they are a middle school student, the explanation and definitions will be much simpler than if the user claims they are a licensed doctor.

## 5 IMPLEMENTATION DETAILS

The code is hosted at https://github.com/vpate70/MedEasyne

### 5.1 Technologies

- **Frontend:** React
- **Backend:** Python, Uvicorn (server), FastAPI (REST API), Whoosh (search library)
- **LLM:** Meta-Llama (local) or OpenAI GPT-3.5 (web API)

### 5.2 Program Flow

The frontend of MedEasyne is implemented as a React app. The user inputs text into these corresponding components, and upon pressing "search", a REST call is made to the backend `/search` endpoint via FastAPI, a Python library for handling REST requests.

The backend is implemented in Python and for search uses Whoosh, a search engine library where we use the Okapi BM25 ranking function. The search is performed only on the title and abstract to keep results prompt. The documents are collected in a

```
"i am a doctor" avg time: 17.117537331581115
"i am medical student" avg time: 19.37762131690979
"i am a middle schooler" avg time: 16.821037101745606
total avg 17.772065250078835
```

**Figure 1: Time to extract and define for LLama3 for different contexts, given the same abstracts.**

searchable index, where this index is created by downloading the PubMed article database via FTP. We use a one-time Python script to download and parse each set of articles, creating a Whoosh index that can be searched quickly by the library using the query text contained in the REST call.

We then send the Whoosh results back to our React frontend via another REST call, where the results are displayed to the user. From the frontend, if the user requests details regarding aspecific result, a REST call is again made to the backend, but this time to the `/llm` endpoint, containing the corresponding abstract and user-provided self-description. We then prompt an LLM, either Meta-Llama or OpenAI GPT-3.5, the former hosted locally on the backend and the latter using a simple API call to OpenAI's servers, with the prompt

"

```
You are extracting and defining difficult healthcare words
from the input passage. Tailor the response to someone
fitting the following description: {user_desc}

{abstract}"
```

The LLM output is then sent back to the React frontend and displayed to the user next to the requested search result.

## 6 EVALUATIONS

The evaluations for our system will include the speed at which responses are generated and the usefulness of what the local Meta-LLama has generated for extraction and definition. The speed of the time it takes the endpoint to respond was tested for both the Search and LLM. Using simple search queries like 'Telemedicine','Mental Health', 'Precision Medicine', and 'Healthcare Disparities', the search over 100 endpoint hits took an average of 1.18 seconds to respond with results. The speed for local LLama3-8b instruct LLMs gave results too slowly with times ranging from 2-4 minutes based on the response size. The default Llama3-8b did not fit in the VRAM of a 3060 with 12GB of VRAM. Thus, it was outsourcing to the CPU making the time it took for inferencing slow. As a result of the slow response, we ended up needing to use a quantized version of the LLama-8b model. With 12GB of VRAM and needing to run the frontend, Whoosh search, and the backend endpoints, we ended up using a 4-bit and 1-sign-bit quantized version of the model from SanctumAI [2]. With the help of Llama C++ [1], this would run within only about 9GB of VRAM. Within Fig. 1 we can see the average time in seconds it took for LLama3 to extract and define those words given the same abstracts. With a total average time of about 18 seconds for inferencing, this quantized model was the correct choice. Fig. 3 shows the breakdown of the time it took for the different parts of the model.

For how useful the model is at extracting, defining, and tailoring to the user, we will compare an example.The responses from Figures 5, 6, 7 are generated from an abstract from a paper about calculating the linear transfer coefficients. From those responses, varying knowledge, you can see the amount of phrases/words defined increase as the education level of the user decreases. In the response for all three of the contexts for users, the model gives a small blurb about the user's experience, defines some phrases, and then finally has a conclusion. From these empirical results, the model does well to extract, define, and tailor the responses.

```
llama_print_timings:        load time =    4186.51 ms
llama_print_timings:      sample time =     249.35 ms /   404 runs   (    0.62 ms per token,  1620.24 tokens per second)
llama_print_timings: prompt eval time =     282.35 ms /   124 tokens (    2.28 ms per token,   439.17 tokens per second)
llama_print_timings:        eval time =   19341.30 ms /   403 runs   (   47.99 ms per token,    20.84 tokens per second)
llama_print_timings:       total time =   22169.38 ms /   527 tokens
```

**Figure 2: Example of time it takes for 3 to generate tokens.**

## 7 USAGE GUIDE

### 7.1 As a User

The search engine is quite intuitive as the design is simple from a user perspective. As seen in Fig. 4, there is an initial text box for entering the context about the user, and below the user can enter the search query. When a search is executed, there will be boxes like Fig 2. that appear for each hit. The user can use the button within each result entry to "Show details" to generate a personalized response to further understand the abstract of the research paper.

### 7.2 As a Developer

To host the website, you'll need to start up the Uvicorn server by running from the project directory:

```
python ./backend./main.py
```

And then the frontend is accessible by starting as any react app, first cding to the React app, then running `npm start`:

```
cd ./frontend/react/my-app && npm start
```

For the search to work, you need an appropriate Whoosh `index` folder. To create the index we use in the paper, which is a subset of the PubMed database, run

```
python ./backend./indexmaker.py
```

This will create an index folder `./backend/index`, which is where our code expects it to be. If you wish to use your own database, hence create your own index, refer to the Whoosh documentation regarding that: https://whoosh.readthedocs.io/en/latest/indexing.html

## 8 REFERENCES

(1) https://github.com/abetlen/llama-cpp-python
(2) https://huggingface.co/SanctumAI/Meta-Llama-3-8B-Instruct-GGUF
(3) https://whoosh.readthedocs.io/en/latest/index.html
(4) https://www.uvicorn.org/
(5) https://fastapi.tiangolo.com/

## 9 FIGURES

The following pages are figures used in the paper or for illustrative reasons, too large to fit in the columns:

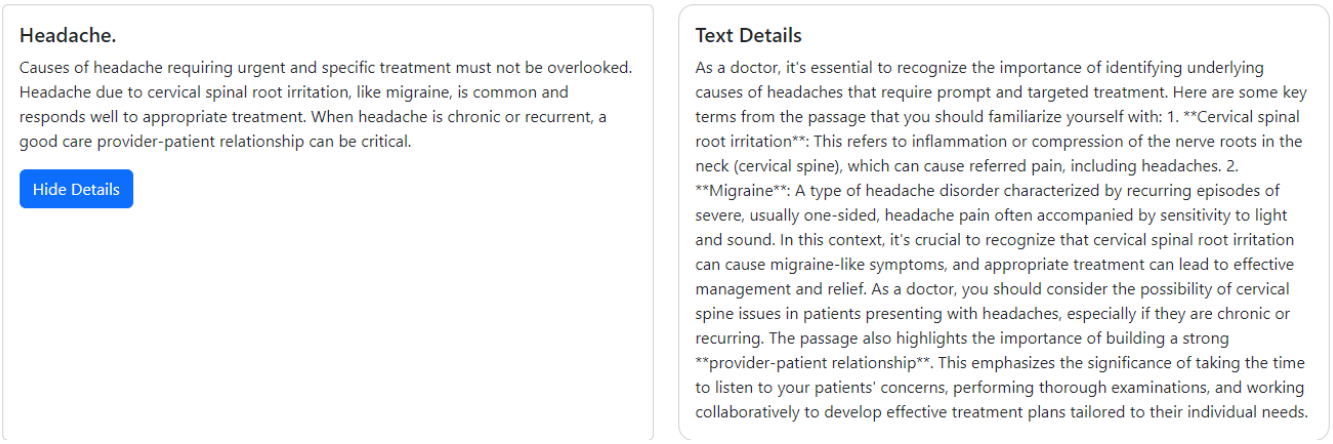


**Figure 3: Response for the search query "headache" and "I'm a doctor" as a profession context.**

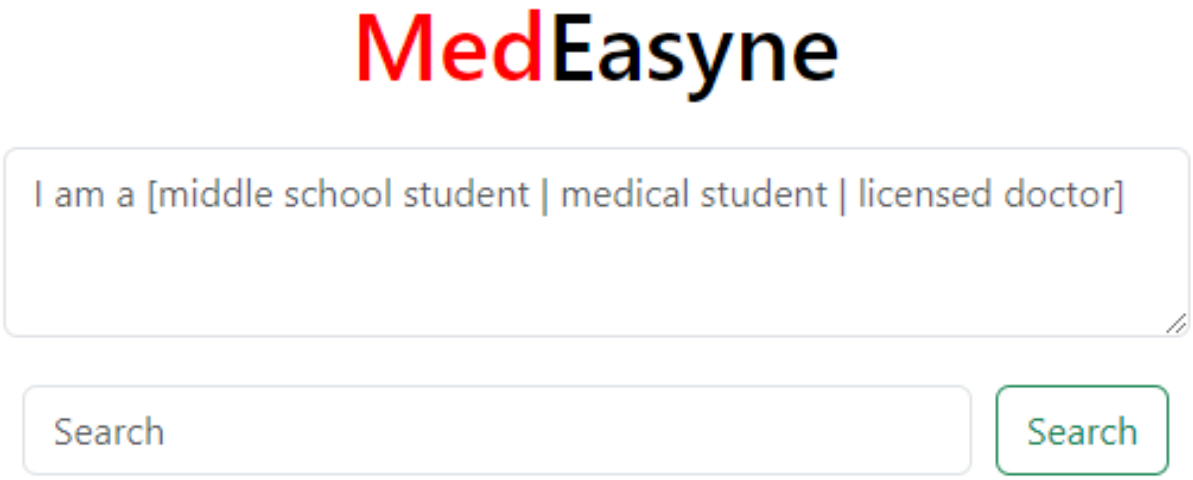


**Figure 4: Landing page of MedEasyne**

As a doctor, you'll appreciate the technical language used in this passage, which describes a computer program for analyzing data in nuclear medicine.

Let's break down some challenging healthcare terms:

1. **Compartment model**: In medical imaging, a compartment model represents a system as a series of interconnected compartments or regions that exchange substances. This model helps researchers understand how substances move within the body and can be used to diagnose and treat diseases.
2. **Transfer coefficients**: These are mathematical values that describe the rate at which substances flow between compartments in the body. In the context of nuclear medicine, transfer coefficients are crucial for understanding the movement of radioactive tracers through the body.
3. **Least-squares fit**: This is a statistical method used to find the best-fit line or curve that minimizes the sum of the squared differences between observed data points and the predicted values.
4. **Weight factors**: In statistics, weight factors are used to adjust the relative importance of each data point when performing a least-squares fit. In this context, the program uses the specified precision of the observed values as weight factors to ensure that more precise measurements have a greater impact on the final results.
5. **Uncertainty assessment**: This refers to the process of estimating the range of possible values within which the true value of a measurement or calculation lies. In the context of transfer coefficients, uncertainty assessment helps researchers understand the reliability of their results and make informed decisions.

The passage also mentions nuclear medicine as an application area for this computer program. Nuclear medicine uses small amounts of radioactive material to diagnose and treat diseases, such as cancer. The program described in the passage can help researchers analyze data from these studies by estimating transfer coefficients and assessing uncertainty in those estimates.

**Figure 5: Response tailored to a doctor.**

As a medical student, you'll likely come across complex terms related to radiopharmaceuticals and pharmacokinetics in your studies. Let's break down some of these difficult healthcare words from the passage:

1. **Linear transfer coefficients**: These refer to the rate at which substances move between compartments or regions within a system. In the context of nuclear medicine, this might involve understanding how a radiotracer distributes throughout the body.
2. **Compartment model**: This is a mathematical representation of a biological system, divided into smaller, hypothetical compartments that interact with each other. Compartment models are used to simplify complex physiological processes and predict how substances will move through the body.
3. **Time-compartment content curves**: These are graphical representations of the amount of substance present in each compartment over time. In nuclear medicine, these curves might show how a radiotracer accumulates in different parts of the body after injection.
4. **Least-squares fit**: This is a statistical technique used to find the best possible curve that fits a set of data points. In this case, the program uses a least-squares fit to match the experimental time-compartment content curves with the predicted values from the compartment model.
5. **Weight factors**: These are used in the least-squares fit process to adjust the importance or "weight" given to each individual data point when calculating the best-fit curve. In this context, the weight factors are based on the precision of the observed values.
6. **Uncertainty**: This refers to the range of possible values within which a calculated value is likely to lie. In this case, the program can estimate the uncertainty in the calculated transfer coefficients, providing an idea of the reliability of the results.

As you delve deeper into nuclear medicine and pharmacokinetics, you'll encounter more complex terminology related to compartment models, radiotracers, and statistical analysis. Understanding these concepts will help you better comprehend the underlying principles of radiopharmaceuticals and their applications in medical imaging and therapy.

**Figure 6: Response tailored to a medical student.**

Let's break down some tricky healthcare words from this passage into simpler terms, tailored for a middle schooler:

* **Transport/Exchange of matter**: This means moving or exchanging substances from one place to another. For example, when you eat food, the nutrients are absorbed into your body and then transported to different parts like your muscles and brain.
* **Natural system**: A natural system refers to living things like our bodies, as well as natural environments like forests and oceans.
* **Functional studies**: These are scientific experiments that try to understand how something works or functions. In this case, it's studying how substances move through the body.
* **Nuclear medicine**: This is a type of medical treatment that uses special kinds of rays (called radiation) to help diagnose or treat diseases like cancer.
* **Compartment model**: Imagine your body as a series of connected containers (compartment 1, compartment 2, etc.). In this case, the program helps scientists understand how substances move between these "containers" and get to their final destination in the body.
* **Time-compartment content curves**: This is a fancy way of saying "a graph that shows how much substance is present in each container at different times". Think of it like tracking how your favorite snack changes amount over time (e.g., from 10 cookies at 3 pm to 5 cookies at 6 pm).
* **Least-square fit**: This means the program tries to find the best match between the experiment results and the model. It's like trying to find the perfect puzzle piece that fits exactly.
* **Precision**: In this context, it refers to how closely the calculated transfer coefficients (more on that later) match the actual values.
* **Weight factors**: These are special numbers used by the program to make sure the most important data points get more attention when calculating the transfer coefficients.
* **Uncertainty**: This is a measure of how reliable or accurate the results are. Think of it like trying to predict your grades on a test – there's always some uncertainty about whether you'll get an A, B, or C.

Some other tricky words that might be useful to know:

* **Transfer coefficients**: These are numbers that describe how substances move between different compartments in the body.
* **Compartment model parameters**: These are special values used to create the compartment model and understand how substances move within it.

I hope this helps you better understand some of these complex healthcare words!

**Figure 7: Response tailored to a middle school student.**

## CONTRIBUTIONS

### John Lay - johnlay2

- Prototype website frontend
- FastAPI endpoints
- Whoosh (search engine library) backend
- Search index creation/tuning
- Wrote sections 4-5 (Major Functions and Implementation Details)
- README (and section Usage Guide: As a Developer)
- Recorded presentation

### Moksh Shah - moksh2

- Downloaded and Extracted PubMed Data
- Assisted with development of backend search functionality
- Assisted with development of backend ranking and index creation
- Wrote sections 1-3 (Introduction, Motivation, Intended Users)
- Created Presentation

### Vivek Patel - vpate70

- LLM/local Llama code
- React frontend
- Screenshots of example usage
- Wrote Evaluations and User Guide: As a User
- Demo Recording