Disease Dynamics: Web Application for Exploring Disease Mechanisms and Medicine Names through PubMed Analysis

BIIN 450-01

Esra Idil

May 10th, 2024

Abstract:

Drug repurposing is a tool that is gaining increasing attention in the pharmaceutical field as it takes advantage of existing drugs with known safety profiles which could potentially mean lower cost and time associated with clinical development stages. Several challenges still face drug repurposing such as the long research process that goes into building the background profile of the new drug. The web application-based tool, Disease Dynamics, aims to accelerate this process by optimizing search results for research. The tool works by accepting keywords related to the disease being researched such as techniques, genes, chromosomes, biological mechanisms, and other related topics. It then pulls the top 100 most relevant articles published in PubMed and provides a calculated relevancy score. The webpage also displays a list of medication names that had appeared in the returned articles which can be used as a starting point for drugs that can be further looked into by the user. The goal of Disease Dynamics is to provide the researcher with an easy-to-use webpage that starts off the research process for diseases that can be treated with repurposed drugs. Disease Dynamics is available at https://github.com/esraidil/BIIN-Project.

Introduction:

The Food and Drug Administration (FDA) has approved an average of 51 new drugs annually (cite 1). When this statistic is compared to the 148,204 registered clinical studies in the United States alone as of April 2024 (cite 2), the numbers seem disproprtionate. With the advanced technology available for use today it is easy to assume medicine discovery is quicker than ever but this field still faces several challenges that prevent it from progressing any faster. From finding funding to hiring scientists, drug discovery can be an expensive ordeal. Additionally, for the drug to be successful in its clinical trial phases and go out into production for use, it needs to satisfy the FDA's many standard requirements. This lengthy and intricate process is essential for the safety of the patient who will eventually take the medication but it also means that many patients who need the treatment may miss out while waiting for the medication to be approved.

One way scientists are tackling these issues in drug discovery is by repurposing drugs. This severely shortens development time as the medicine used for repurposing is already approved and consequently lowers the cost and associated risk of failure. A systematic approach to repurposing can further improve the outcomes of the application. Although this tool is not new, it has started gaining increased attention in the last decade and it has great growth potential.

My capstone project is a web application tool that speeds up the research aspect of drug repurposing. This is done by optimizing literature searching and text mining in the MEDLINE subcategory of the PubMed database. Natural language processing is the main technique that automates text mining and assists in searching, filtering, and understanding hundreds of articles to identify patterns and develop connections from the text at a much faster pace. The user can input biological mechanisms of the disease they wish to explore and get back a list of relevant articles from PubMed and identified patterns of common medications in these groups. This tool is intended to aid the research of an existing drug with relevant articles and applicable medication ideas that will cut back on the time that is needed to go through when starting research.

Materials and Methods:

<u>Data</u>

The data that is used for this project is primarily from the PubMed database, specifically its MEDLINE subset. Both public online databases are maintained by the National Center for Biotechnology Information (NCBI) which is located at the National Institutes of Health (NIH). The PubMed database contains more than 37 million abstracts of biomedical articles that are free and available for public use. While the article entry doesn't contain the full text, there are links available in the entry for the full text, usually through PubMed Central (PMC) or the publisher's own website. MEDLINE contains more than 31 million citations to biomedicine and life science journal articles. The reason the MEDLINE component on PubMed is used is because it requires a more specific selection process for articles compared to the rest of PubMed. The journal articles are also indexed with MeSH terms which allows users doing specific searches with more relevant results to the search. The MEDLINE citations are searched through the PubMed database.

<u>Programming</u>

Python:

Python is the programming language used to design the functionality of the tool. Python is a general-purpose programming language with a high-level yet easy-to-use syntax making it a go to language for many projects.

HTML:

HyperText Markup Language or HTML is a markup language used to design web pages. The hypertext defines the content and layout of the webpage. This language is generally used in addition to a scripting language to provide structure.

CSS:

Cascading Style Sheets or CSS is the stylesheet language used to design the webpages and add style. It is used to make a presentable webpage either in addition to HTML or independently of.

For this project, two HTML files with CSS styling were used to create the layouts and design of the two webpages of the project and one python file was used to create the main code of the web application. The following four functions were written in Python and they make up the main component of the tool:

1. *index Function*

The index function renders the homepage that is displayed when the application is run. The webpage is a jinja template made of HTML and CSS styling.

2. *search Function*

The search function handles the searching and result mechanisms that are deployed when the user enters keywords in the web application. Specifically, search_terms splits the search entered by the user into separate words at every delimiter of a comma. This search is done through Entrez and it results in a list of articles pulled from PubMed. The search function renders the search_results page that will be displayed when the keywords are entered in the homepage by the user. The webpage is a jinja template made of HTML and CSS styling.

3. *calculate_relevance_score Function*

The calculate_relevance_score function generates a score for every output in the "Article Results" section of the web application. The score is calculated by counting the number of times any one of the keywords entered by the user shows up in the output "Title" and "Abstract" of the results.

4. *pubmed_records Function*

The pubmed_records function deploys a couple of different operations. Two lists collect elements throughout the function, table_results which stores the information that will be used to create the table for "Article Results" and common_med_names which stores the information that will be used to create the curated list for "Common Medication Names". These two functions are called within the search function as that is where the results are displayed in the web application.

Packages

Entrez from BioPython:

Entrez is a search tool developed and maintained by NCBI to assist with searching needs in most online biological databases. This tool can pull a variety of information such as the number of hits from a database provided the keywords to search for, records of articles from the database matching the criteria, and so much more. In this project, Entrez was used to search the PubMed database for disease mechanisms entered by the user and return the top 100 articles that match the criteria.

Xml.etree.ElementTree:

XML stores data in text format which makes it easy for storing, transporting, and working with data. PubMed only accepts electronic submissions that are formatted in their preferred xml tagged format which can be found in their website. This project uses the package Xml.etree.ElementTree as a way to work with the xml data as it implements an API to parse and create xml data.

Wordpunct_tokenize from NLTK Tokenizer:

This package returns tokens from a list of strings or words. It specializes in separating words within a string and can be used as the initial step before searching through the tokens. This project uses wordpunct_tokenize to separate the words in the returned list of abstracts and article titles as separate tokens.

Find_drugs from drug_name_entity_recognition:

The find_drugs function of the drug_name_entity_recognition package finds any drug names within a string. In this project, the find_drugs function works collaboratively with the NLTK package to split the text into separate words and then identify any drug names within the words.

Re:

Regular expression is a seqeunce of characters that sets the pattern being searched for. In this project, the results of common medication names returned from the find_drugs function comes in the form of the drug name followed by all of its synonyms and accession IDs for different databases. As this project only requires the name of the drug, the regular expression function was used to pull and display only the name of the drug and forgo the remaining result.

Web Framework

Flask Web Application:

Flask is a python web framework used in the backend for web application creation. It only needs a single Python file to deploy the application and it supports Python libraries and tools for a dynamic web application. In this project, Flask is used to set up a local server and to load HTML files with Jinja2 templating.

Jinja2:

Jinja2 is a user friendly templating language for Python. It is used in collaboration with HTML and Flask to create an easy to use template.

Initial Setup of Flask

There are a few steps that were completed only before the first time the Flask application was run.

1. The Python Package Manager “pip” was used to setup the virtual environment
    - pip install virtualenv

2. We then created the virtual environment in our project folder which is also where the Flask application will be stored
    - virtualenv venv
3. Once we are located within the same folder as the “venv” folder, we can activate the environment by using the following commands (this was done in Windows so different operating systems may require slightly different commands)
    - venv\Scripts\activate
4. We can install Flask in the virtual environment
    - pip install flask
5. The server is ready to be run after creating the .py file containing the Flask information and supporting HTML files for the design of the application
    - flask run

Results:

Design and Implementation:

Disease Dynamics architecture is composed by an interactive web browser for user input and a second web page for the visualization of the results.

Figure 1 shows the pipeline to produce results in the Disease Dynamics tool. The flask application is ran on the local server to open the website on http://127.0.0.1:5000. The HTML file that contains the design details for the initial homepage of the application is then rendered. This page provides a search box that allows the user to enter in the keywords they wish to search for. These keywords should be descriptive of the disease they aim to treat. Some examples of keywords include disease mechanisms, gene names, chromosome numbers, mutation types, laboratory techniques, medicine names, and similar areas. These keywords are sent through MEDLINE in PubMed to search for the highest matching reports. The results are filtered to only return back the top 100 reports. The search_results HTML file is then rendered to display the results. The titles and abstracts of the articles are displayed in a table format. The first column holds the title of the article, the second column the abstract of the article, and the third column has the generated relevancy score of the article to the user input. The title and abstract are then searched to find any drug names. The curated list of drug names is given above the table of article results. This page also returns the keywords entered as a header to serve as a reminder to the user.

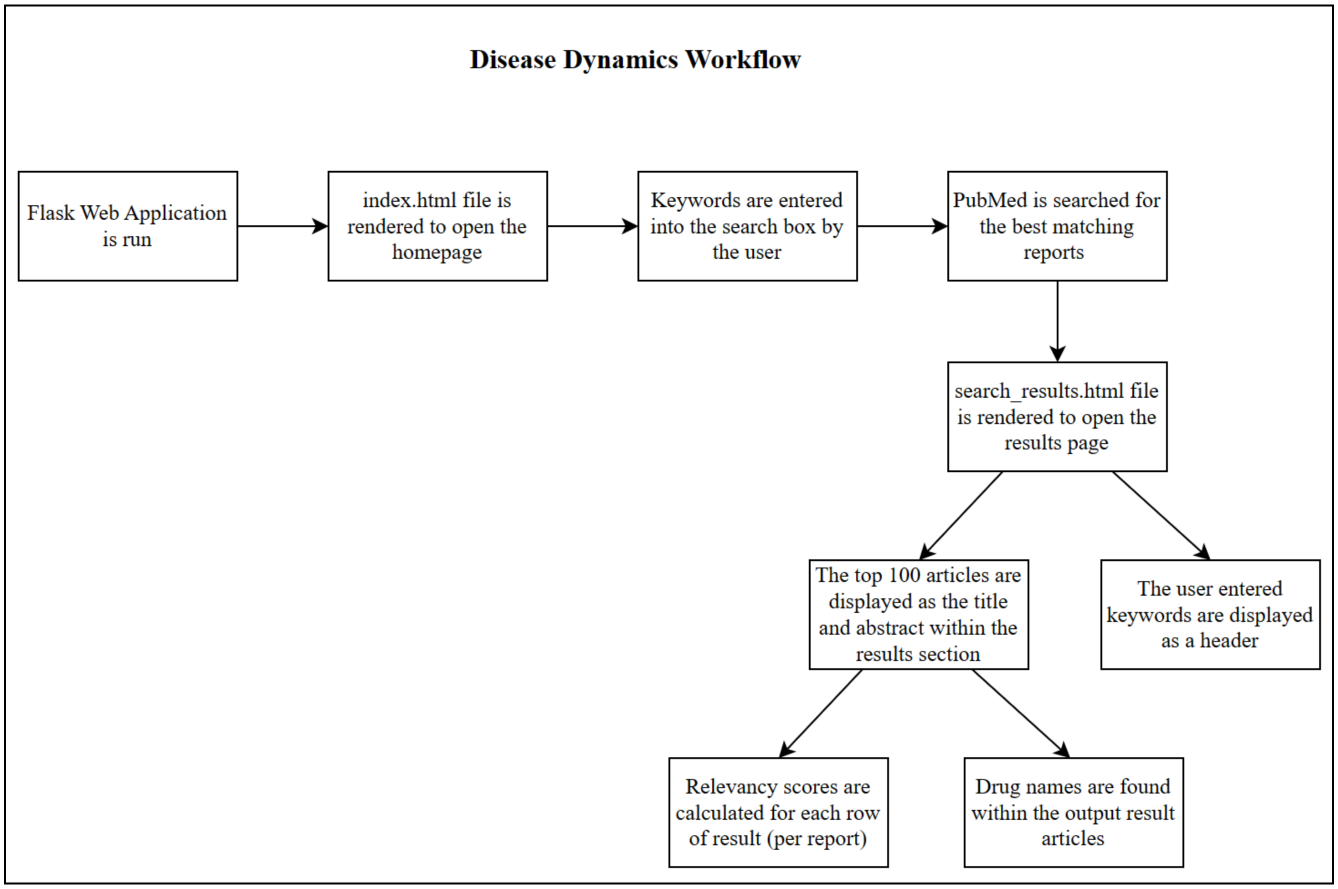

Disease Dynamics Workflow
Flask Web Application is run
index.html file is rendered to open the homepage
Keywords are entered into the search box by the user
PubMed is searched for the best matching reports
search_results.html file is rendered to open the results page
The top 100 articles are displayed as the title and abstract within the results section
The user entered keywords are displayed as a header
Relevancy scores are calculated for each row of result (per report)
Drug names are found within the output result articles

Fig. 1: Overview of the Disease Dynamics workflow.

Running the Web Application

Below are the steps to run the web application from your local server:

- Download the three files included in this repository
- Open the Command Prompt
- Set the directory to where the three files are now saved on your pc
- Type the command "python BIIN_FINAL_PROJECT.py" in the Command Prompt and click the enter button
- Wait a few seconds until you get the following message and then open the url on your browser

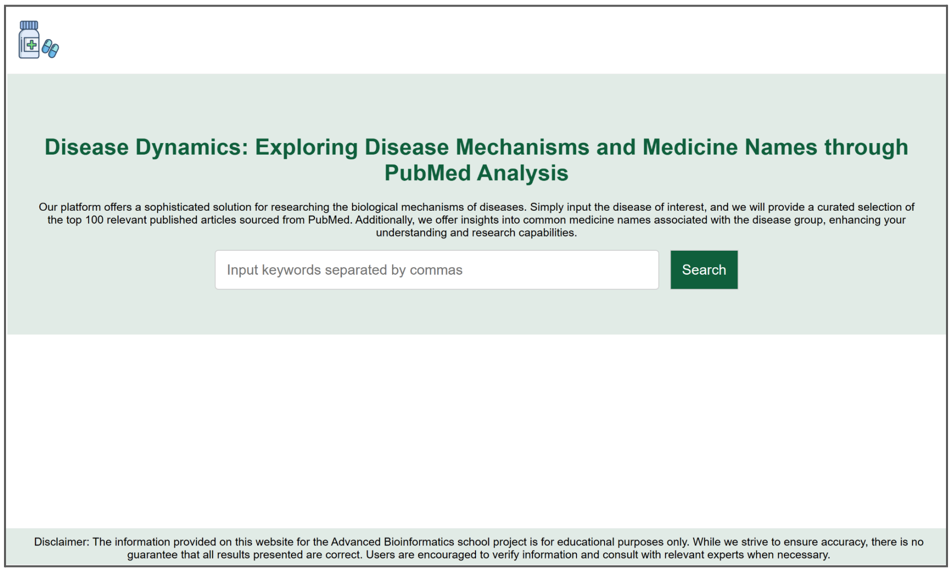
```
Serving Flask app 'BIIN_FINAL_PROJECT'
Debug mode: on WARNING: This is a development server. Do not use it in a production
deployment. Use a production WSGI server instead.
Running on http://127.0.0.1:5000 Press CTRL+C to quit
Restarting with stat
Debugger is active!
Debugger PIN: 135-337-769
```

You should now be on the homepage of the flask app. You can enter a couple of keywords about the techniques used to find drugs, mechanisms of a disease, genes affected by a disease, and so on to explore relevant articles that will help start the accelerated research process of drug discovery. When you press the "Search" button you might have to wait a minute or two before getting any results.

The results page will display a few things. The user-inputted keywords will be displayed in the header, a list of medications found throughout the relevant articles, and the titles and abstract as well their relevancy scores of the 100 most relevant articles.

<u>Output Description:</u>

Figure 2 displays the homepage of the application when first ran. This page greets the user with an informative description of the tool and a quick instruction on how to use the program.



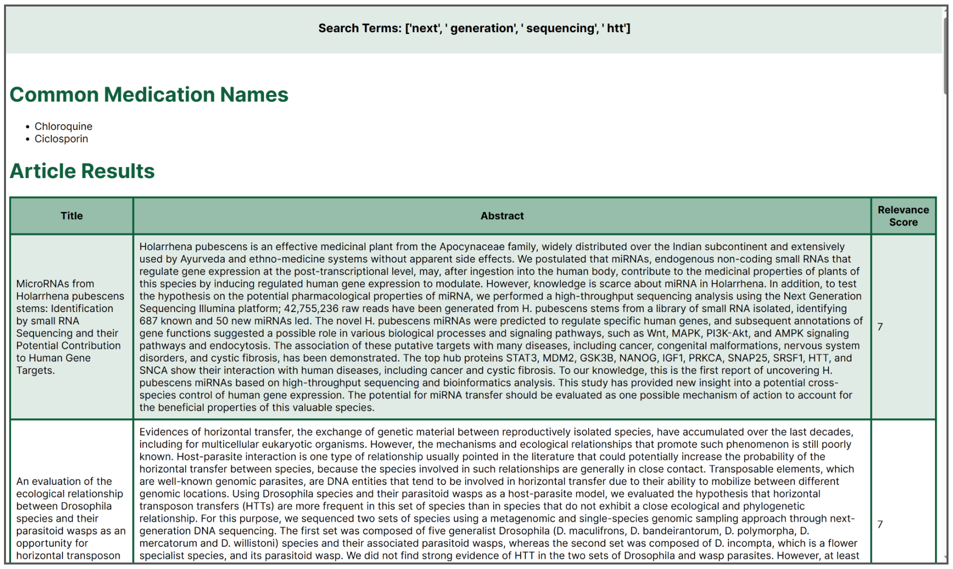
Disease Dynamics: Exploring Disease Mechanisms and Medicine Names through PubMed Analysis

Our platform offers a sophisticated solution for researching the biological mechanisms of diseases. Simply input the disease of interest, and we will provide a curated selection of the top 100 relevant published articles sourced from PubMed. Additionally, we offer insights into common medicine names associated with the disease group, enhancing your understanding and research capabilities.

Disclaimer: The information provided on this website for the Advanced Bioinformatics school project is for educational purposes only. While we strive to ensure accuracy, there is no guarantee that all results presented are correct. Users are encouraged to verify information and consult with relevant experts when necessary.

Fig. 2: User Interface of the Index of the web application.

Figure 3 displays the results returned after the user enters keywords in the search box of the homepage. For this specific example, the user entered “next”, “generation”, “sequencing”, “htt” which would indicate that the researcher wanted to learn more about the applications of the next generation sequencing technique on the htt gene. This could possibly mean that the researcher is searching for a treatment option using next generation sequencing as a cure for Huntington’s disease.


Search Terms: ['next', ' generation', ' sequencing', ' htt']

Common Medication Names

- Chloroquine
- Ciclosporin

Article Results

Title	Abstract	Relevance Score
MicroRNAs from Holarrhena pubescens stems: Identification by small RNA Sequencing and their Potential Contribution to Human Gene Targets.	Holarrhena pubescens is an effective medicinal plant from the Apocynaceae family, widely distributed over the Indian subcontinent and extensively used by Ayurveda and ethno-medicine systems without apparent side effects. We postulated that miRNAs, endogenous non-coding small RNAs that regulate gene expression at the post-transcriptional level, may, after ingestion into the human body, contribute to the medicinal properties of plants of this species by inducing regulated human gene expression to modulate. However, knowledge is scarce about miRNA in Holarrhena. In addition, to test the hypothesis on the potential pharmacological properties of miRNA, we performed a high-throughput sequencing analysis using the Next Generation Sequencing Illumina platform; 42,755,236 raw reads have been generated from H. pubescens stems from a library of small RNA isolated, identifying 687 known and 50 new miRNAs led. The novel H. pubescens miRNAs were predicted to regulate specific human genes, and subsequent annotations of gene functions suggested a possible role in various biological processes and signaling pathways, such as Wnt, MAPK, PI3K-Akt, and AMPK signaling pathways and endocytosis. The association of these putative targets with many diseases, including cancer, congenital malformations, nervous system disorders, and cystic fibrosis, has been demonstrated. The top hub proteins STAT3, MDM2, GSK3B, NANOG, IGF1, PRKCA, SNAP25, SRSF1, HTT, and SNCA show their interaction with human diseases, including cancer and cystic fibrosis. To our knowledge, this is the first report of uncovering H. pubescens miRNAs based on high-throughput sequencing and bioinformatics analysis. This study has provided new insight into a potential cross-species control of human gene expression. The potential for miRNA transfer should be evaluated as one possible mechanism of action to account for the beneficial properties of this valuable species.	7
An evaluation of the ecological relationship between Drosophila species and their parasitoid wasps as an opportunity for horizontal transposon	Evidences of horizontal transfer, the exchange of genetic material between reproductively isolated species, have accumulated over the last decades, including for multicellular eukaryotic organisms. However, the mechanisms and ecological relationships that promote such phenomenon is still poorly known. Host-parasite interaction is one type of relationship usually pointed in the literature that could potentially increase the probability of the horizontal transfer between species, because the species involved in such relationships are generally in close contact. Transposable elements, which are well-known genomic parasites, are DNA entities that tend to be involved in horizontal transfer due to their ability to mobilize between different genomic locations. Using Drosophila species and their parasitoid wasps as a host-parasite model, we evaluated the hypothesis that horizontal transposon transfers (HTTs) are more frequent in this set of species than in species that do not exhibit a close ecological and phylogenetic relationship. For this purpose, we sequenced two sets of species using a metagenomic and single-species genomic sampling approach through next-generation DNA sequencing. The first set was composed of five generalist Drosophila (D. maculifrons, D. bandeirantorum, D. polymorpha, D. mercatorum and D. willistoni) species and their associated parasitoid wasps, whereas the second set was composed of D. incompta, which is a flower specialist species, and its parasitoid wasp. We did not find strong evidence of HTT in the two sets of Drosophila and wasp parasites. However, at least	7

Fig. 3: User Interface of the Search Results page of the web application.

Discussion and Conclusion:

In conclusion, Disease Dynamics is a research optimization tool for users who wish to start researching about potential durg repurposing opportunities. It was developed to allow for a faster and more efficient start to research. The article title and abstract outputs can be used to develop an idea of what direction the researcher wants to head in and can be used as reference for further deeper research. Most importantly, Disease Dynamics automatically curates a list of medication names found within the pre-determined relevant articles. This list could be useful as a starting point for possible drugs that can be repurposed. Further directiosn of this tools would include allowing the user to specify additional filters to their article results such as the type of article or how old the article is. Another version of this tool could replace the PubMed database with PubMed Central which would be a great option for student researchers who could use that table of full text results in one page rather than the citations. These improvements would prove useful to researchers as well and would further simplify data exploration.

References:

1. About - PubMed. 2023. PubMed. https://pubmed.ncbi.nlm.nih.gov/about/.
2. Drug to Target/Gene/Protein Interaction Databases. 2012 Feb 14. Biostarsorg. https://www.biostars.org/p/17392/.
3. drug-named-entity-recognition. 2024 Apr 14. PyPI. https://pypi.org/project/drug-named-entity-recognition/.
4. Elsevier. 2021. wwwelseviercom. https://www.elsevier.com/industry/drug-repurposing.
5. GeeksforGeeks. 2019 Jun 6. Python NLTK tokenize.WordPunctTokenizer(). GeeksforGeeks. https://www.geeksforgeeks.org/python-nltk-tokenize-wordpuncttokenizer/.
6. GeeksforGeeks. 2020 Apr 28. Reading and Writing XML Files in Python. GeeksforGeeks. https://www.geeksforgeeks.org/reading-and-writing-xml-files-in-python/.
7. GeeksforGeeks. 2022 Dec 28. Templating With Jinja2 in Flask. GeeksforGeeks. https://www.geeksforgeeks.org/templating-with-jinja2-in-flask/.
8. GeeksforGeeks. 2022 Dec 28. Templating With Jinja2 in Flask. GeeksforGeeks. https://www.geeksforgeeks.org/templating-with-jinja2-in-flask/.
9. GeeksforGeeks. 2023 Nov 24. HTML and CSS. GeeksforGeeks. https://www.geeksforgeeks.org/html-css/.
10. HTML Images. 2024. W3schoolscom. https://www.w3schools.com/html/html_images.asp.
11. HTML Styles CSS. 2024. W3schoolscom. https://www.w3schools.com/html/html_css.asp.

12. HTML Tutorial. 2024. W3schoolscom. https://www.w3schools.com/html/default.asp.

13. Kulkarni VS, V. Alagarsamy, Solomon VR, Jose PA, Murugesan S. 2023. Drug Repurposing: An Effective Tool in Modern Drug Discovery. Russian Journal of Bioorganic Chemistry. 49(2):157–166. doi:https://doi.org/10.1134/s1068162023020139. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9945820/.

14. MEDLINE Overview. 2024. Nihgov. [accessed 2024 May 9]. https://www.nlm.nih.gov/medline/medline_overview.html.

15. Mullard A. 2022. 2021 FDA approvals. Nature reviews Drug discover/Nature reviews Drug discovery. 21(2):83–88. doi:https://doi.org/10.1038/d41573-022-00001-9. https://www.nature.com/articles/d41573-022-00001-9.

16. National Center for Biotechnology Information. 2024. Nihgov. https://www.ncbi.nlm.nih.gov/.

17. National Institutes of Health (NIH). 2019. National Institutes of Health (NIH). https://www.nih.gov/.

18. NLTK :: Natural Language Toolkit. 2023. Nltkorg. [accessed 2024 May 8]. https://www.nltk.org/.

19. re — Regular expression operations. 2024. Python documentation. https://docs.python.org/3/library/re.html.

20. Real Python. 2021 Feb. Python Web Applications: Deploy Your Script as a Flask App. Realpythoncom. https://realpython.com/python-web-applications/.

21. Real Python. 2021 May 5. Natural Language Processing With Python's NLTK Package. Realpythoncom. https://realpython.com/nltk-nlp-python/.

22. Topic: Clinical trials. 2021. Statista. https://www.statista.com/topics/6756/clinical-trials/#statisticChapter.

23. Vineela Parvathaneni, Kulkarni NS, Muth A, Gupta V. 2019. Drug repurposing: a promising tool to accelerate the drug discovery process. Drug discovery today. 24(10):2076–2085. doi:https://doi.org/10.1016/j.drudis.2019.06.014. https://pubmed.ncbi.nlm.nih.gov/31238113/#:~:text=Drug%20repurposing%20has%20therefore%20become%20a%20productive%20approach,drugs%20for%20new%20use%20but%20encounters%20several%20challenges.

24. Vineela Parvathaneni, Kulkarni NS, Muth A, Gupta V. 2019. Drug repurposing: a promising tool to accelerate the drug discovery process. Drug discovery today. 24(10):2076–2085. doi:https://doi.org/10.1016/j.drudis.2019.06.014. https://www.sciencedirect.com/science/article/abs/pii/S1359644618302599#:~:text=A%20comparison%20of%20traditional%20drug%20discovery%20process%20versus,abolishes%20all%20the%20steps%20needed%20for%20FDA%20approval.

25. xml.etree.ElementTree — The ElementTree XML API. 2024. Python documentation. https://docs.python.org/3/library/xml.etree.elementtree.html.

Appendix:

BIIN_FINAL_PROJECT.py

```
from flask import Flask, render_template, request
from Bio import Entrez
import xml.etree.ElementTree as ET
from nltk.tokenize import wordpunct_tokenize
from drug_named_entity_recognition import find_drugs
import re

app = Flask(__name__)

Entrez.email = 'eidil@ramapo.edu'

#This section runs the index.html file which holds the home page of the
web app. It is also where the user input is entered.
@app.route('/', methods=['GET'])
def index():
    return render_template('index.html')

#This section runs the search_results.html file which holds the results
page of the web app. It is also where the results returned from the user
input is displayed.
@app.route('/search', methods=['POST'])
def search():

    #User enters keywords that are then split by the delimiter comma and
stored separately as search_terms then joined to form search_query.
    search_terms = request.form.get('search_terms', '').split(',')
    search_query = ' AND '.join(search_terms)

    #Entrez is used to search the PubMed database for the search_query
words and returns the 100 most relevant articles.
    handle = Entrez.esearch(db="pubmed", term=search_query, retmax=100)
    record = Entrez.read(handle)

    #This section runs the function pubmed_records for the two main types
of results, records and common_med_names.
    ID_List = record['IdList']
    records, common_med_names = pubmed_records(ID_List, search_terms)
    records.sort(key=lambda x: x['relevance_score'], reverse=True) #Sorts
the column of the table named relevance_score in descending order.
    common_med_names = sorted(common_med_names)

    #Finally, the search_results.html file is rendered and is given
information from the search_terms, records, and common_med_names.
    return render_template('search_results.html',
search_terms=search_terms, records=records,
common_med_names=common_med_names)

#The relevance_score is calculated for each row in the table by counting
the number of occurrences of all search terms in the title and abstract.
def calculate_relevance_score(search_terms, title, abstract):
    score = 0
    for term in search_terms:
        score += title.lower().count(term.lower())
        score += abstract.lower().count(term.lower())
    return score

#The function used to pull pubmed articles from the database and further
pull information such as title and abstract from these records.
def pubmed_records(ID_List, search_terms):
    table_results = [] #This list holds the title, abstract, and
rlevance_score information found from pubmed articles.
    common_med_names = [] #This list holds the common medication names
that show up more than once in the title and abstract returned by
table_results.

    #The for loop goes through each pubmed article retrieved from pubmed.
    for pubmed_ID in ID_List:
        handle = Entrez.efetch(db="pubmed", id=pubmed_ID, retmode="xml")
        xml_data = handle.read()
        root = ET.fromstring(xml_data)

        #The for loop connects to the three web pages and retrieves the
data as article, title, and abstract.
        for article in
root.findall('.//PubmedArticle/MedlineCitation/Article'):
```

```
            title = article.find('.//ArticleTitle').text if
article.find('.//ArticleTitle') is not None else 'N/A'
            abstract = article.find('.//AbstractText').text if
article.find('.//AbstractText') is not None else 'N/A'

            #Make sure that null data is replaced with 'N/A' to avoid
errors.
            if title is None:
                title = 'N/A'
            if abstract is None:
                abstract = 'N/A'

            #The wordpunct_tokenize function from the NLTK tokenizer
package separates words in a sentence so that each word can be analyzed
separately.
            title_tokens = wordpunct_tokenize(title)
            abstract_tokens = wordpunct_tokenize(abstract)

            #The find_drugs function from the
drug_named_entity_recognition package compares the list of words from the
tokens output against the list of available drugs in their database and
flags matches.
            title_drugs = find_drugs(title_tokens, is_ignore_case=True)
            abstract_drugs = find_drugs(abstract_tokens,
is_ignore_case=True)
            all_drugs = title_drugs + abstract_drugs

            pattern = r"'name':\s*'([^']+)'" #The raw data obtained from
the find_drugs function is in the form of the drug name and all of its
synonyms so the pattern specifies that the program is only interested in
the drug name.

            #This for loop uses the re module to form a match between the
pattern and string of drugs found, if there is a match, it will place it
in the medicine_names list.
            for drug in all_drugs:
                match = re.search(pattern, str(drug))
                if match:
                    medicine_name = match.group(1)
                    common_med_names.append(medicine_name)
            common_med_names = list(set(common_med_names))

            #The relevance_score function is used to assign a score for
the combination of the title and abstract.
            relevance_score = calculate_relevance_score(search_terms,
title, abstract)

            #The table is created for the results from articles.
            table_results.append({
                'title': title,
                'abstract': abstract,
                'relevance_score': relevance_score
            })

    return table_results, common_med_names

if __name__ == '__main__':
    app.run(debug=True)
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Disease Dynamics: Exploring Disease Mechanisms and Medicine
Names through PubMed Analysis</title>
    <style>

        body {
            font-family: 'Inter', sans-serif;
            margin-bottom: 100px;
        }

        .main {
            background-color: #e5ebe8;
            padding: 4rem 2rem;
            text-align: center;
        }

        #logo {
            display: inline-block;
            margin: 15px;
            height: 60px;
            width: auto;
        }

        .search-form {
            display: flex;
            justify-content: center;
            align-items: center;
            gap: 1rem;
        }

        .search-input {
            width: 100%;
            max-width: 600px;
            padding: 1rem;
            border: 1px solid #ccc;
            border-radius: 5px;
            font-size: 1.25rem;
        }

        .search-button {
            background-color: #11633f;
            padding: 1rem;
            border: 1px solid #ccc;
            color: #fff;
            font-size: 1.25rem;
        }

        footer {
            position: fixed;
            bottom: 0;
            left: 0;
            width: 99%;
            background-color: #e5ebe8;
```

```
            padding: 10px;
        }
    </style>
</head>
<body>
    <a href="#"><img id="logo"
src="https://allspinesurgerycenter.com/wp-content/uploads/2021/05/drugs.pn
g"></a>
      <div class="main">
        <h1 style="color: #11633f;">Disease Dynamics: Exploring Disease
Mechanisms and Medicine Names through PubMed Analysis</h1>
        <p>
            Our platform offers a sophisticated solution for researching
the biological mechanisms of diseases.
            Simply input the disease of interest, and we will provide a
curated selection of the top 100 relevant published articles sourced from
PubMed.
            Additionally, we offer insights into common medicine names
associated with the disease group, enhancing your understanding and
research capabilities.
        </p>
    <form action="/search" method="POST" class="search-form">
        <input type="text" id="search_terms" name="search_terms"
class="search-input" placeholder="Input keywords separated by commas" />
        <button type="submit" class="search-button">Search</button>
    </form>
    <footer> Disclaimer:
        The information provided on this website for the Advanced
Bioinformatics school project is for educational purposes only. While we
strive to ensure accuracy,
        there is no guarantee that all results presented are correct.
Users are encouraged to verify information and consult with relevant
experts when necessary.
    </footer>
</body>
</html>
```

search_results.html

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Search Results</title>
    <style>

        body {
            font-family: 'Inter', sans-serif;
        }

        table {
            width: 100%;
            border-collapse: collapse;
            margin-bottom: 20px;
        }
        th, td {
            border: 3px solid #11633f;
            padding: 8px;
        }
        th {
            background-color: #97c1ae;
        }
        tr:nth-child(even) {
            background-color: #e5ebe8;
        }

        header {
            position: fixed;
            top: 0;
            left: 0;
            width: 99%;
            background-color: #e5ebe8;
            padding: 10px;
            text-align: center;
        }

    </style>
</head>
<body>
    <header>
    <h3>Search Terms: {{ search_terms }}</h3>
    </header>
    </div>
    <h1 style="color: #11633f; margin-top: 125px;">Common Medication
Names</h1>
    <ul>
        {% for name in common_med_names %}
            <li>{{ name }}</li>
        {% endfor %}
    </ul>
    <h1 style="color: #11633f;">Article Results</h1>
    <table>
        <tr>
            <th>Title</th>
            <th>Abstract</th>
            <th>Relevance Score</th>
        </tr>
        {% for record in records %}
        <tr>
            <td>{{ record.title }}</td>
            <td>{{ record.abstract }}</td>
            <td>{{ record.relevance_score }}</td>
        </tr>
        {% endfor %}
    </table>
</body>
</html>
```