IAS Summer Internship 2021

Project Progress Report

Indian Academy of Science

8-Week Progress Report

Prepared by

Subham Das

Application Number: [redacted]

Guide: Dr. Areejit Samal, IMSc Chennai

July 10, 2021

TABLE OF CONTENTS

ACKNOWLEDGMENT

I would like to thank Indian Academy of Science for giving me the opportunity to participate in a project as big and impactful as this, it was truly an enriching experience.

I would like to express my deep and sincere gratitude to my, Dr. Areejit Samal, for giving me this opportunity and providing invaluable guidance throughout the project. He has taught me the methodology to carry out the project and to present it as clearly as possible. He has been a constant source of encouragement and has never failed to provide useful insight to progress the work without any problems. It was a great privilege and honor to work under his guidance.

Finally I would like to thank Mr. Ajay Subbaroyan and Mr. Rishabh, the team that has been constantly and wholeheartedly working behind this project tirelessly for the past 8 weeks.

Building a Database of Validated Logic Gates in Synthetic Biology

INTRODUCTION

- Synthetic biological circuits are biological parts inside a cell designed to perform logical functions mimicking those observed in electronic circuits. The applications range from simply inducing production to adding a measurable element, like GFP, to an existing natural biological circuit, to implementing completely new systems of many parts.

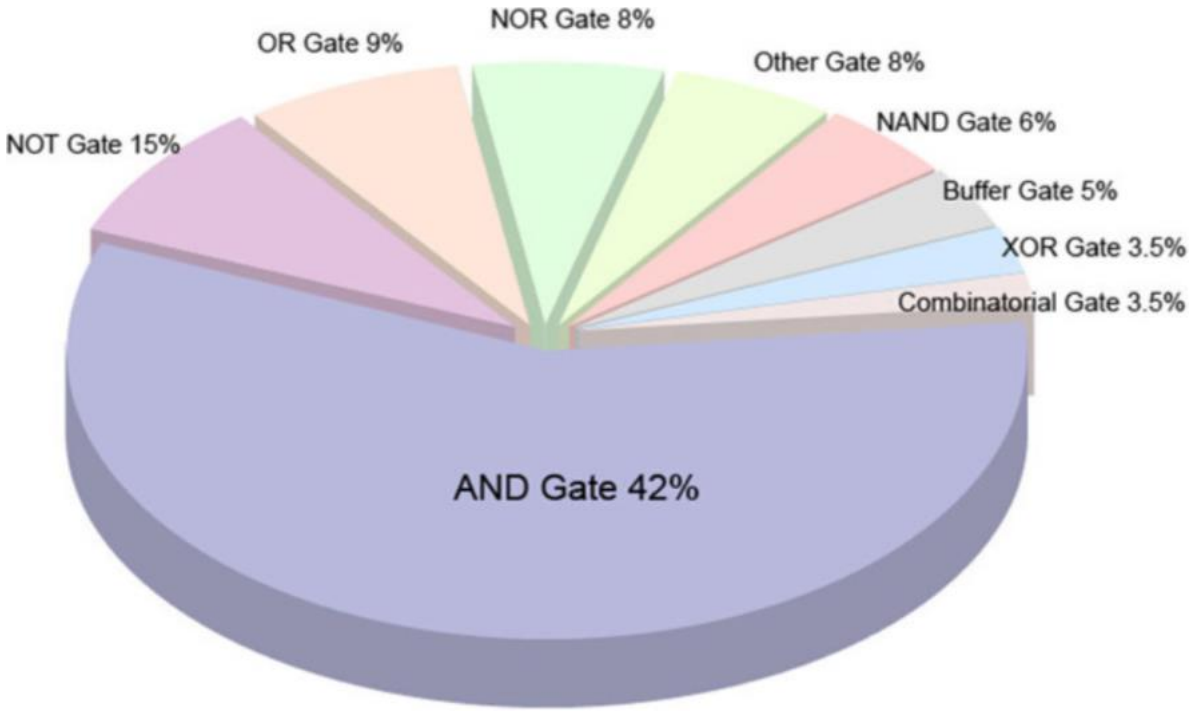
  The goal of synthetic biology is to generate an array of tunable and characterized parts, or modules, with which any desirable synthetic biological circuit can be easily designed and implemented. These circuits can serve as a method to modify cellular functions, create cellular responses to environmental conditions, or influence cellular development.

- Over the past 8-weeks, we have analyzed and gone through a wide range of papers and prepared a database that is a collection of experimentally validated logic gates.
  We have documented AND gates, Buffer gates, Combinatorial gates, NAND gates, NOR gates, NOT gates, OR gates, XOR gates, and other gates across many species (Human, Escherichia coli and Bacillus clausii, Yeast, etc.). Currently, we have a total of **389** such logic gates.

- The database provides information regarding the input, output, and species originated from along with the paper referenced and its author. The database will be of particular interest to the life sciences community and facilitate the application of rational engineering principles to the design of biological systems based on genetic logic gates, thereby combining electronic engineering with cell biology. The database will provide a global view of genetic logic gates in synthetic biology. Researchers can follow these genetic logic gates to explore how the input and output of genes, proteins, and promoters are organized.

- The team plans to further increase the range and content of the database and subsequently present the information collected in a simple and elegant manner to the entire science community. It would also provide them with a chance to add on to the database ensuring its validity remains for a substantial period.

- Finally we aim to provide a simple PUBMED search query which would enable people to read these exact papers and explore other relevant ones. It would also help researchers find more papers in the future by inputting the same search query.



METHODS

(i) Data Collection – A positive dataset

The project started with the collection of a positive dataset, i.e., collection of papers which we knew contained synthetic logic gates. We manually analyzed hundreds of papers, all the way from 2002 to 2021, and eventually ended up with 389 logic gates across 68 research papers.

We gave each logic gate an ID and noted its input, output, species and PMID.

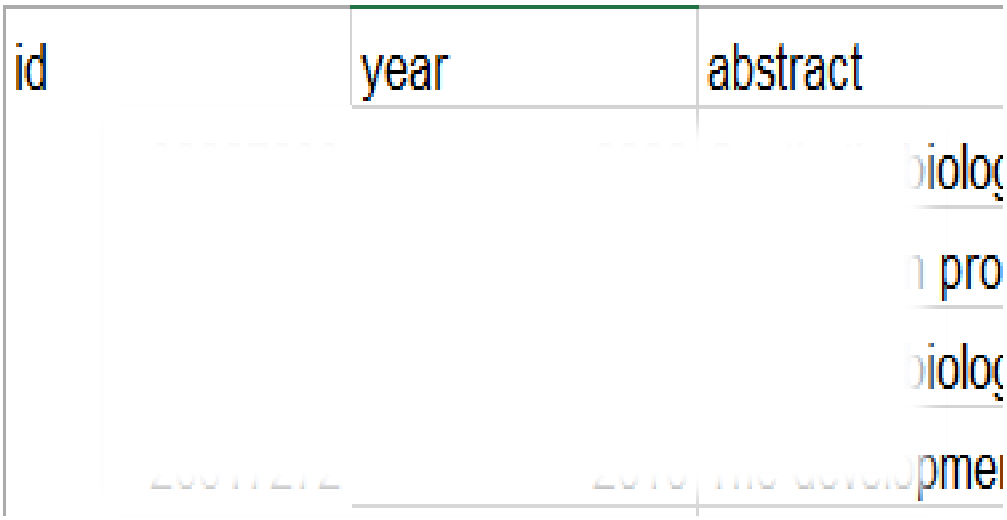
GATE ID	Category	INPUT 1	INPUT 2	INPUT 3	INPUT 4	OUTPUT	Species	PMID
LF001	AND	pLuxI*	pTet*			pcysG	Pseudomonas	2304
LF002								304
LF003								304
LF004								150
LF005								150
LF006	NOT	pRhl	-			Yellow Fluoroscent Protein (YFP)	E. Coli	21150

(ii) Preparing the dataset for analyzing

Before we could move on to finding more papers of this type using a more efficient method, we had to prepare a dataset to import into our NLP project.

The idea was to find the most frequently occurring words in the abstracts of all papers and prepare a PUBMED query search such that we not only get our positive dataset as a such result but also additional new papers that can be added.

The PUBMED query search would also help researchers find new gates in the future (the ones discovered after the completion of this project). Hence, a .csv, and .tsv file was created having three columns – id, year, and abstract, that could be imported in the project and easily analyzed.



(iii) Applying NLP

The main part of the project involved analyzing all the papers collected and come up with an efficient PUBMED query search that would not only help find more relevant papers but also serve as the basis for future searches allowing people to access exactly the information related to logic gates.

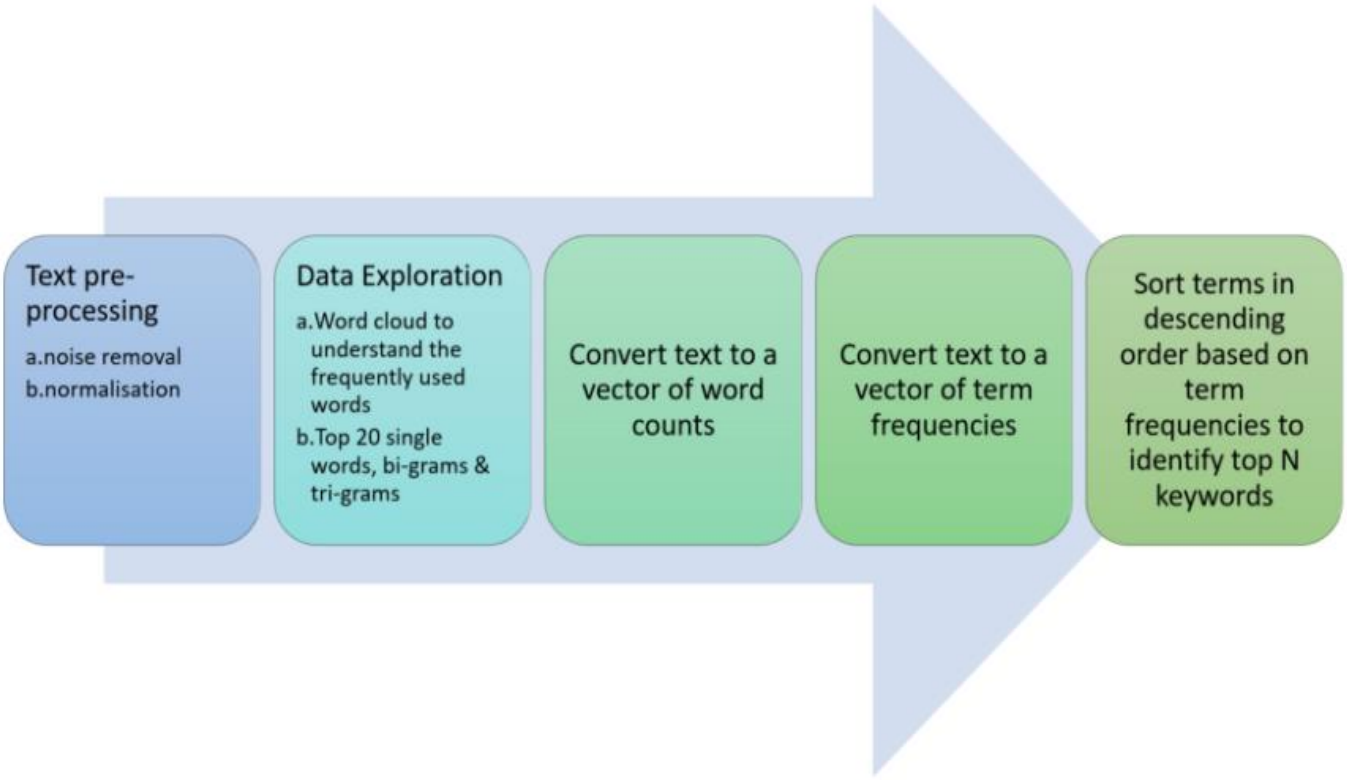
The .csv file prepared before was imported and analyzed using different techniques to get the relevant keyword search.

Keywords form an important component since they provide a concise representation of the article's content. Keywords also play a crucial role in locating the article from information retrieval systems, bibliographic databases and for search engine optimization. Keywords also help to categorize the article into the relevant subject or discipline.

A wide range of free Python NLP libraries offer some relatively easy-to-deploy tools that can help us uncover key features of large datasets. The tools we have used in this project allowed us to create single and multi-term keyword lists (called n-grams) from our data set. This lists will help us frame a basic high level understanding of the dataset's contents.

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, semantic reasoning, and wrappers for industrial-strength NLP libraries.

The general flow employed in the program to obtain data is as follows:



- Keywords themselves can be useful, particularly in formulating a question.
- Keywords can help focus in on smaller sets of individual records in order to learn more about them and begin to answer particular questions about user needs and goals
- Keywords in combination with analysis of smaller sets of individual records can help us identify gaps in understanding of users that can help focus subsequent research efforts.

Using the full article text could be computationally intensive, so only abstracts have been used for NLP modelling. The same code block can be used on the full article text to get a better and enhanced keyword extraction, which would be the next phase of the project.

Before we proceed with any text pre-processing, we will explore the dataset in terms of word counts, most common and most uncommon words.

	id	year	abstract
0	[redacted]	2020	Synthetic biology is primarily an emerging res...
1		2012	Information processing and decision-making is ...
2		2012	Synthetic biology has advanced the design of s...
3		2010	The development of a highly parallel enzyme lo...
4		2019	Controlling gene expression with sophisticated...

VIEWING THE DATASET

	abstract	word_count
0	Synthetic biology is primarily an emerging res...	187
1	Information processing and decision-making is ...	201
2	Synthetic biology has advanced the design of s...	176
3	The development of a highly parallel enzyme lo...	272
4	Controlling gene expression with sophisticated...	118
...	...	...
63	The endonuclease scission of magnetic particle...	163
64	The engineering of biological systems is antic...	126
65	Gene regulation in biological systems is impac...	233
66	The construction of artificial networks of tra...	142
67	Cells are able to navigate environments, commu...	145

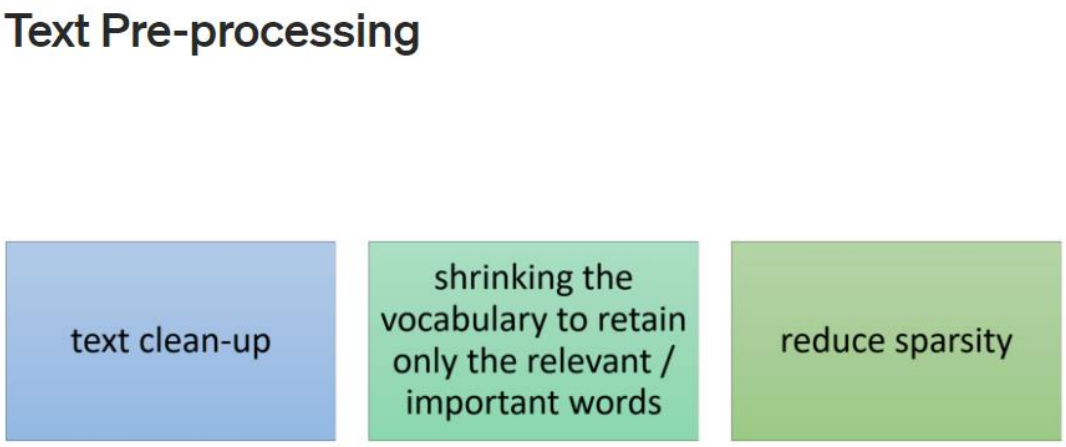
68 rows × 2 columns

CHECKING THE WORD COUNT

```
count     68.000000
mean     167.573529
std       56.957779
min       16.000000
25%      142.000000
50%      153.500000
75%      206.000000
max      326.000000
Name: word_count, dtype: float64
```

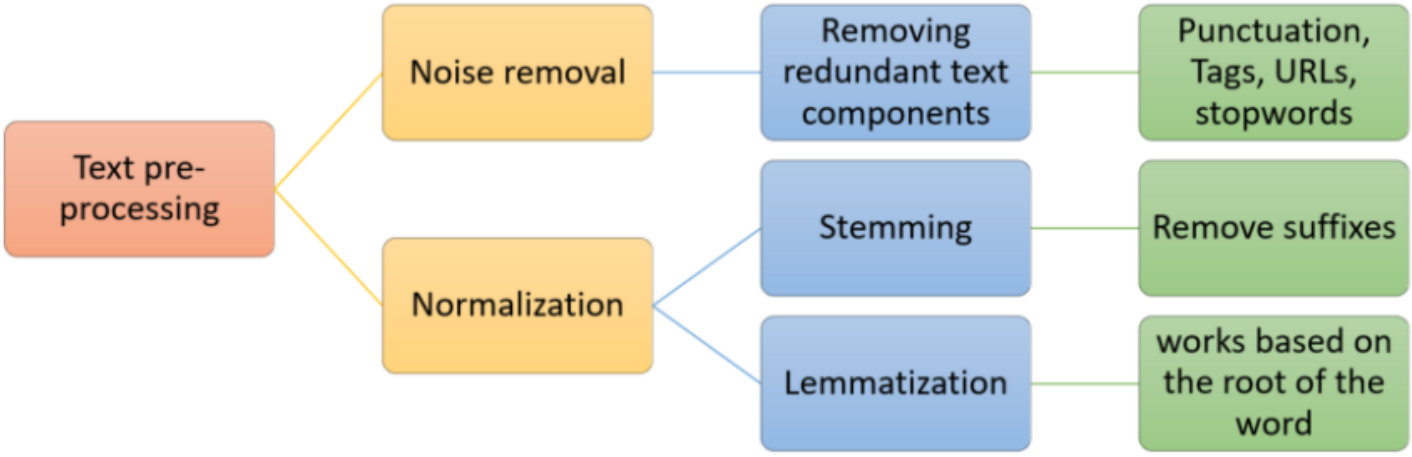
DESCRIPTIVE STATISTICS OF WORD COUNT

Next we move on to text preprocessing.



Sparsity*:* In text mining, huge matrices are created based on word frequencies with many cells having zero values. This problem is called sparsity and is minimized using various techniques.

Text pre-processing can be divided into two broad categories — noise removal & normalization. Data components that are redundant to the core text analytics can be considered as noise.



Handling multiple occurrences / representations of the same word is called **normalization**. There are two types of normalization — **stemming** and **lemmatization**.

Ex: Let us consider an example of various versions of the word learn — learn, learned, learning, learner. Normalization will convert all these words to a single normalized version — “learn”.

Stemming normalizes text by removing suffixes.

Lemmatization is a more advanced technique which works based on the root of the word.

Removing stop words: Stop words include the large number of prepositions, pronouns, conjunctions etc. in sentences. These words need to be removed before we analyze the text, so that the frequently used words are mainly the words relevant to the context and not common words used in the text.

There is a default list of stop words in python NLTK library. In addition, we might want to add context specific stop words for which the “most common words” that we listed in the beginning will be helpful.

```
[ ] ##Creating a list of stop words and adding custom stopwords
    stop_words = set(stopwords.words("english"))

    ##Creating a list of custom stopwords
    new_words = ["using", "show", "result", "large", "also", "one", "two", "new", "previously", "shown"] #More can be added if required
    stop_words = stop_words.union(new_words)
```

The text is now pre-processed step-by-step to get a clean and normalized text corpus:

```
[ ] corpus = []
    for i in range(0, 68):
        #Remove punctuations
        text = re.sub('[^a-zA-Z]', ' ', dataset['abstract'][i])

        #Convert to lowercase
        text = text.lower()

        #remove tags
        text=re.sub("&lt;/?.*?&gt;"," &lt;&gt; ",text)

        # remove special characters and digits
        text=re.sub("(\\d|\\W)+"," ",text)

        ##Convert to list from string
        text = text.split()

        ##Stemming
        ps=PorterStemmer()
        #Lemmatisation
        lem = WordNetLemmatizer()
        text = [lem.lemmatize(word) for word in text if not word in
                stop_words]
        text = " ".join(text)
        corpus.append(text)
```

Text preparation: Text in the corpus needs to be converted to a format that can be interpreted by the machine learning algorithms. There are 2 parts of this conversion — **Tokenization** and **Vectorization**.

Tokenization is the process of converting the continuous text into a list of words. The list of words is then converted to a matrix of integers by the process of vectorization. Vectorization is also called feature extraction.

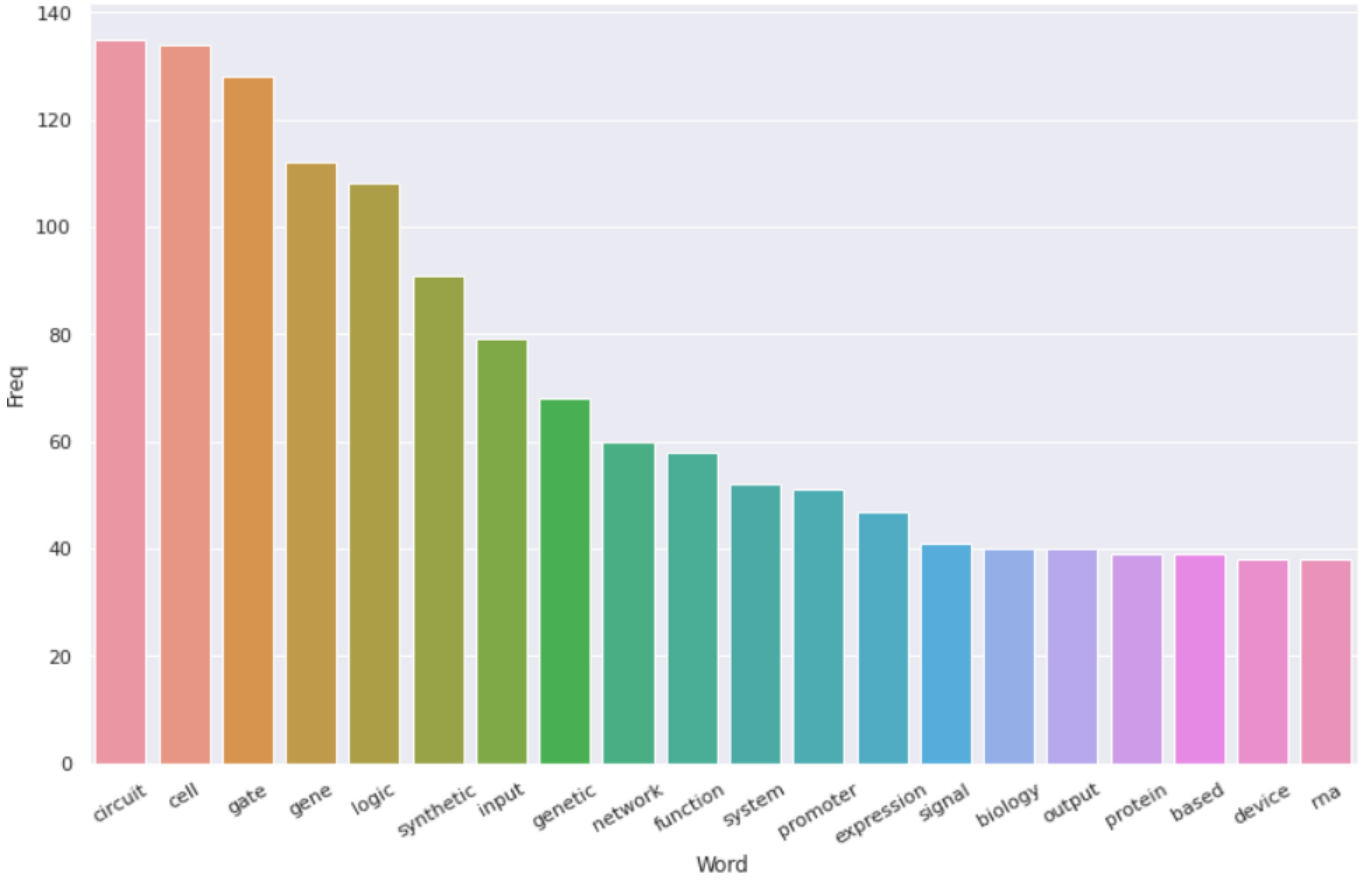
For text preparation we use the ***bag of words model*** which ignores the sequence of the words and only considers word frequencies.

Creating a vector of word counts: As the first step of conversion, we used the *CountVectorizer* to tokenize the text and build a vocabulary of known words. We first created a variable "cv" of the CountVectorizer class, and then evoked the fit_transform function to learn and build the vocabulary.

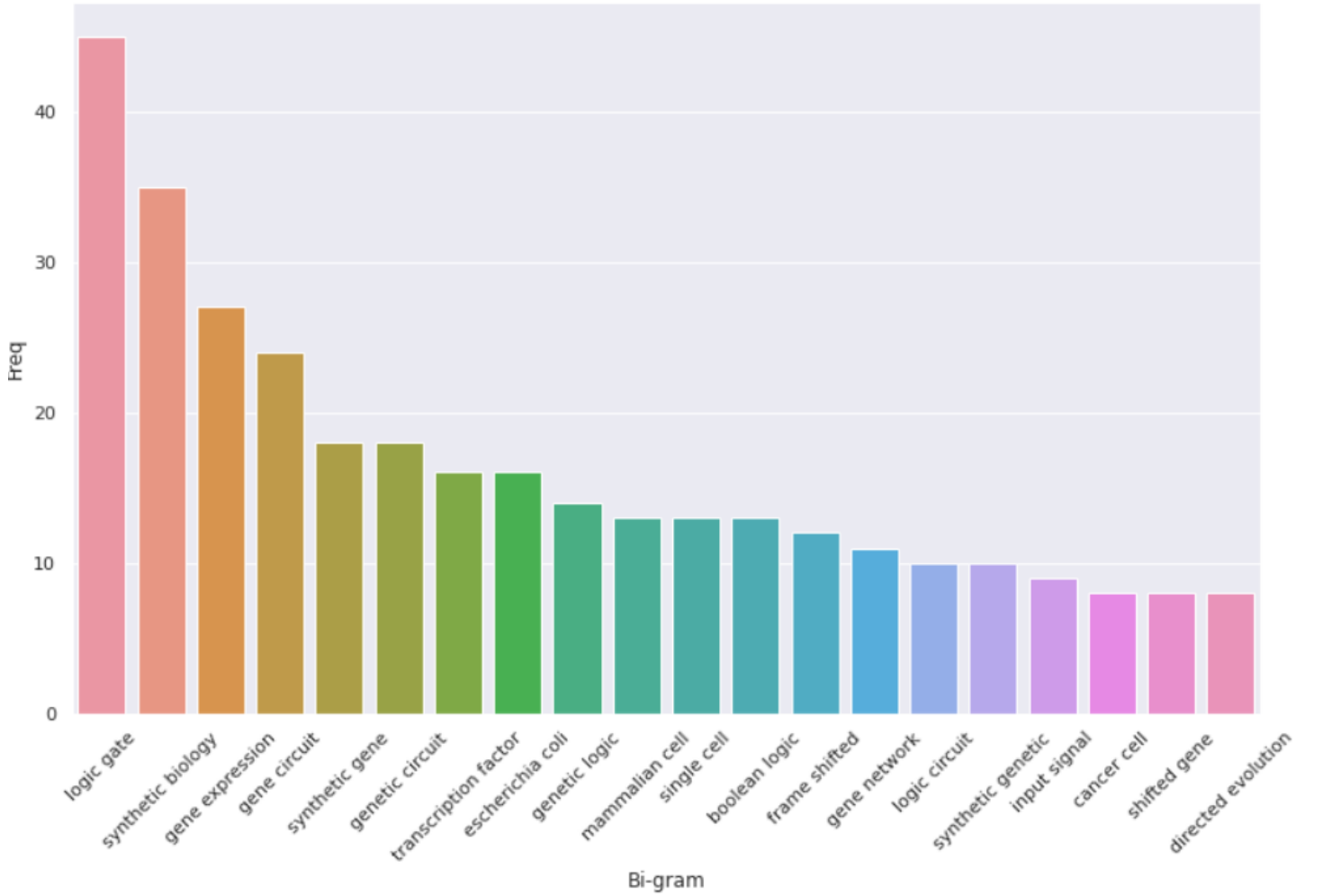
Parameters passed into the function are:

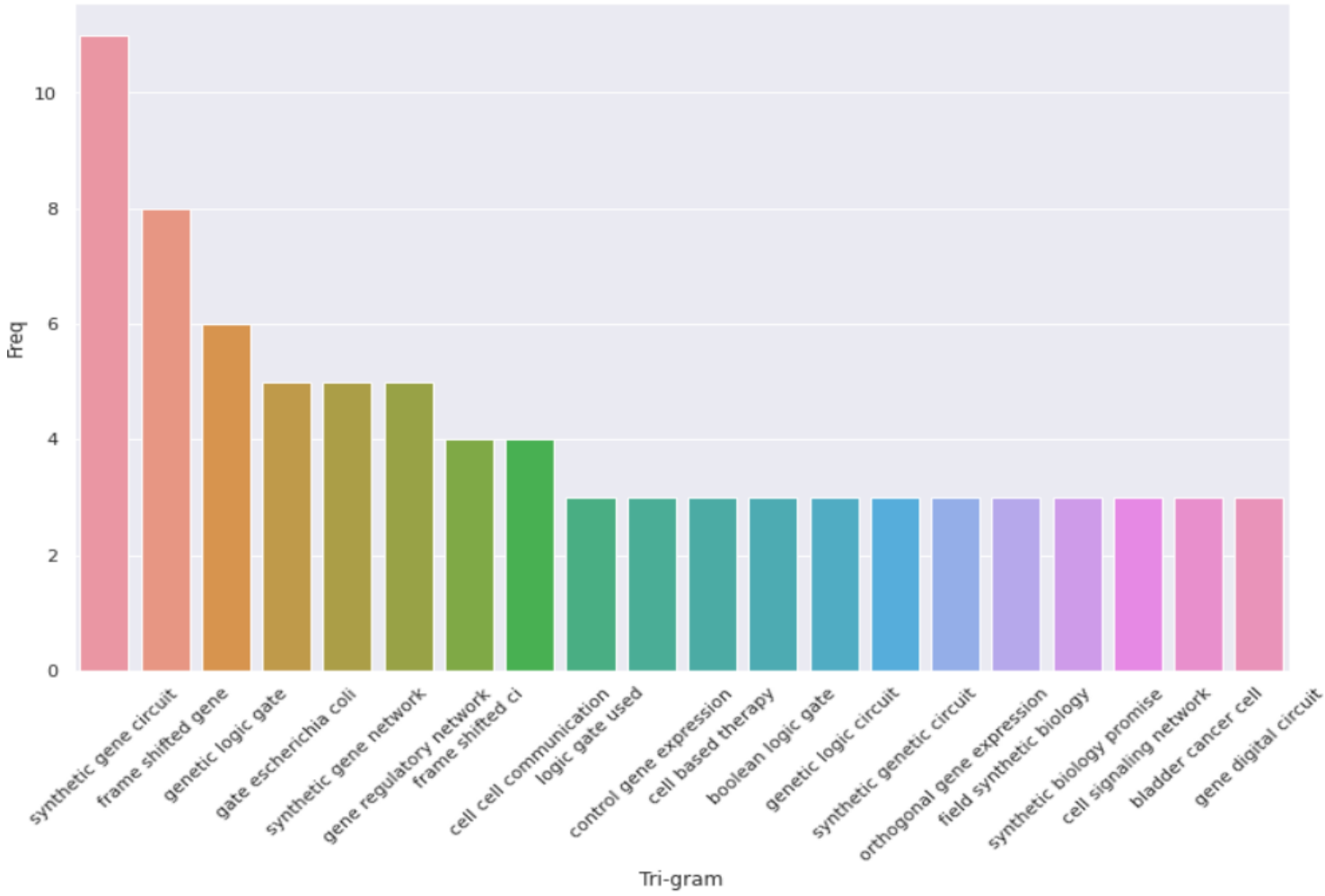
- cv=CountVectorizer(max_df=0.8,stop_words=stop_words, max_features=10000, ngram_range=(1,3))
- max_df — When building the vocabulary ignore terms that have a document frequency strictly higher than the given threshold (corpus-specific stop words). This is to ensure that we only have words relevant to the context and not commonly used words.
- max_features — determines the number of columns in the matrix.
- n-gram range — we would want to look at a list of single words, two words (bi-grams) and three words (tri-gram) combinations.

Visualize top N uni-grams, bi-grams & tri-grams:



	Word	Freq
0	circuit	135
1	cell	134
2	gate	128
3	gene	112
4	logic	108
5	synthetic	91
6	input	79
7	genetic	68
8	network	60
9	function	58
10	system	52
11	promoter	51
12	expression	47
13	signal	41
14	biology	40
15	output	40
16	protein	39
17	based	39
18	device	38
19	rna	38


Freq
0
10
20
30
40
logic gate
synthetic biology
gene expression
gene circuit
synthetic gene
genetic circuit
transcription factor
escherichia coli
genetic logic
mammalian cell
single cell
boolean logic
frame shifted
gene network
logic circuit
synthetic genetic
input signal
cancer cell
shifted gene
directed evolution
Bi-gram


Freq
0
2
4
6
8
10
synthetic gene circuit
frame shifted gene
genetic logic gate
gate escherichia coli
synthetic gene network
gene regulatory network
frame shifted ci
cell cell communication
logic gate used
control gene expression
cell based therapy
boolean logic gate
genetic logic circuit
synthetic genetic circuit
orthogonal gene expression
field synthetic biology
synthetic biology promise
cell signaling network
bladder cancer cell
gene digital circuit
Tri-gram

Converting to a matrix of integers: The next step of refining the word counts is using the TF-IDF vectorizer. The deficiency of a mere word count obtained from the countVectorizer is that, large counts of certain common words may dilute the impact of more context specific words in the corpus. This is overcome by the TF-IDF vectorizer which penalizes words that appear several times across the document. TF-IDF are word frequency scores that highlight words that are more important to the context rather than those that appear frequently across documents.

TF-IDF consists of 2 components:

- TF — term frequency
- IDF — Inverse document frequency

- $\text{TF} = \dfrac{\text{Frequency of a term in a document}}{\text{total number of terms in the document}}$

- $\text{IDF} = \dfrac{\log(\text{Total documents})}{\text{\# of documents with the term}}$

Based on the TF-IDF scores, we can extract the words with the highest scores to get the keywords for a document.

Ideally for the IDF calculation to be effective, it should be based on a large corpora and a good representative of the text for which the keywords need to be extracted. Currently we have used abstracts instead of the full article text, but using the later would make the IDF extraction more effective.

(iv) An efficient PUBMED query search

Constructing an appropriate question

Asking the right question is the primary key to creating a good search. The questions must be answerable, if the question is too broad, the search will yield more information than you can possibly look through.

Using the appropriate database(s)

Knowing what type of information is needed, what type of articles we are searching for, what we will do with the information gathered, etc., will determine the type of information you need and types of articles that may provide that information. It also may influence the type of database you search for that information.

Two types of literature that are often the focus of systematic searches are indexed, peer-reviewed literature, and gray literature.

- Peer-reviewed literature is scholarly work that generally represents the latest original research in a field. These articles undergo expert screening before publication to ensure meaningfulness within the context of other research in the discipline and, at least in theory, sound methodology.
- "Gray" literature refers to material that is not formally published by commercial publishers or peer-reviewed journals, including reports, fact sheets, white papers, conference proceedings, and other documents from various organizations and government agencies.

Indexed peer-reviewed articles will give us the best available and most current data, which includes millions of citations. We will see how to refine and limit the search in order to find exactly what we need.

The initial approach would be to:

- Type a word or phrase into the query box, including subject, author, and/or journal
- Click on the search button or press the "enter" key

- Results will be displayed in summary format.
- To retrieve more information about the search results, we can use the display settings menu (upper left corner) to view the abstract, change the number of items that appear per page, and sort by recently added, publication date, first author, last author, journal, or title.

Advanced searching in PubMed—MeSH terms and the MeSH database:

Medical subject headings (MeSH)

It is important to understand that PubMed uses a controlled vocabulary to index journal articles called MeSH and uses "automatic term mapping" to find MeSH terms when you search. MeSH terms are organized in a hierarchy called a tree, with more specific (narrower) terms arranged beneath broader terms. By default, PubMed includes in the search all narrower terms; this is called "exploding" the MeSH term. Inclusion of MeSH terms enhances and optimizes the search strategy.

MeSH database features

MeSH vocabulary contains over 25,000 descriptors and is updated weekly and reviewed annually. Features include:

- Allows you to identify and select appropriate MeSH terms for a search and to see their definitions
- Builds a PubMed search strategy
- Displays MeSH terms in the hierarchy (MeSH tree) allowing you to broaden/narrow a search
- Limits MeSH terms to a major concept/topic heading for a search
- Allows you to broaden your search by choosing not to explode a term
- Attaches subheadings for a search creating complex search strategies. The list of subheadings includes terms paired at least once with a given heading in MEDLINE.
- Focuses searches using other types of MeSH terms including publication types, substance names or registry numbers, and pharmaceutical actions.

To access MeSH from PubMed, click on MeSH Database on the PubMed homepage or click MeSH under “more resources” in “advanced search.”

Also, clicking “links” adjacent to the MeSH term desired, will give you a drop-down menu which offers several options:

- PubMed: search PubMed with the term
- PubMed—Major topic: search PubMed with the MeSH term, retrieving only citations where the term is a major focus
- Clinical queries: put the MeSH term into the Clinical Queries box where the search may be further refined
- NLM MeSH browser: show the MeSH browser descriptor data for this term including scope note, allowable qualifiers, and the MeSH tree

Refining the search

- Replace general search terms with more specific terms (the MeSH database would be a great resource for this)
- Add terms or combine search terms with connector words: AND, OR, or NOT using upper case letters (called Boolean logic)
    - AND between terms returns only records that contain all of the search terms
    - OR between terms returns all records that contain any of the search terms
    - NOT between search terms returns only records that contain the first term and not the second
- Truncate terms. Place an asterisk (*) at the end of a string of characters to search for all terms that being with that string. PubMed searches the first 600 variations of a truncated term.
- Use a wildcard. Use a “?” to replace a letter or denote an extra letter where spelling or word variation is possible.
    - Example: behavio?r will find behaviour or behavior
- Use the “limit” option in PubMed to limit citations by age group, language, publication type, date, human studies, etc.

- Use the “advanced search” option to look up a term as it is indexed in PubMed
- Use the MeSH database features

By combining terms (using Boolean logic), truncating a term, and using the limits option we were able to narrow our search down from **36,023 articles** to **7,643 articles** (with plans to further reduce this number).

Boolean operators	Command	Example
AND	Find articles where both terms appear	HPV AND cervical cancer
OR	Find articles where either term appears	Granuloma inguinale OR donovanosis
NOT	Find articles where one term appears only when the other term is not present	Angiofibromas NOT tuberous sclerosis

Literature searching requires practice, intuition, and some trial and error. While there is a basic structure, a set of guidelines and many tools for assisting one with basic searches, there are a variety of nuances and advanced techniques that may be required for more specialized searches. For systematic reviews as an example, extensive searches are required and may take numerous hours, involving many databases (including those for gray literature), and a combination of advanced search strategies in order to be methodologically sound. Use of personnel with specialized expertise in conducting such searches may provide the best results and be the most resource effective and time efficient.

RESULT AND FUTURE WORK

The database of logic gates collected will be very useful to the scientific community and help obtain information easily and quickly.

The database currently consists of **389 logic gates** with plans to further increase it while keeping it open sourced, thereby allowing people to add on to it making it an ever growing database.

The PUBMED query search, post NLP processing, has reduced the article number from **36023 to 7643**, and we plan to reduce this further by analyzing the entire text of the research papers and not just the abstract.

Having a smaller search number would make it feasible to check all the papers related to synthetic biology and logic gates from the inception of the idea till the current time. It would also enable future researches to look up papers related to exactly this by putting in the query search that we would be providing.

Our final plans with regard to the project would be to provide the best query search, put up the database in a place that would be accessible to everyone, and publish our results explaining the detailed process involved in building a successful database.

We also plan to emphasize the importance of the dataset by showcasing various scenarios where the use of the information provided by us would not only be helpful but also reduce the amount of time that would normally take in performing manual search, thereby reducing the work of researchers allowing them to focus on other aspects of their project.

CONCLUSION

Synthetic biologists have deciphered genetic logic operations in living cells that may track key moments in a cell's life or change the fate of a cell, and diverse genetic logic gates integrating biological DNA/ molecular inputs and output have been constructed. However, comprehensive understanding of the mutual relationship between biological logic operations in living cells for the execution of complex biological tasks via electrical circuits remains elusive.

Thus, genetic logic gates are essential basic operational tools for building biologically based digital devices to simulate cell signaling. Consequently, we have systematically collected experimentally verified genetic logic gates and established a database centered on the relationship between biological logic operations incorporating biological DNA/molecular inputs and outputs.

The database will be of particular interest to the life sciences community, and facilitate the application of rational engineering principles to the design of biological systems based on genetic logic gates, thereby combining electronic engineering with cell biology.

Finally, we will continue to collate reference data and update this database.