Machine Learning of Biomedical Text for Gene Network Knowledge

This manuscript (permalink) was automatically generated from carlosrojas/genome_graph_paper@2db7338 on May 6, 2023.

Published: February 1, 2023

Machine Learning of Biomedical Text for Gene Network Knowledge

A project Report
Presented to
The Faculty of Computer Engineering Department
San Jose State University

In Partial Fulfillment
Of the Requirements for the Degree
Bachelor of Science in Software Engineering

By
Tue Do
Ausin Fong
Thinh Huynh
Mary Markart
02/2023

Authors

- **Mary Markart** ✉
  · marymarkart
  Charles W. Davidson College of Engineering, San Jose State University

- **Austin Fong**
  · austinf01
  Charles W. Davidson College of Engineering, San Jose State University

- **Tue Doe**
  · tuedolm
  Charles W. Davidson College of Engineering, San Jose State University

- **Thinh Huynh**
  · thinhh
  Charles W. Davidson College of Engineering, San Jose State University

✉ — Correspondence possible via GitHub Issues or email to Mary Markart <mary.markart@sjsu.edu>.

Abstract

Machine Learning of Biomedical Text for Gene Network Knowledge

By Tue Do, Austin Fong, Thinh Huynh, Mary Markart

Genetic research is the study of DNA to find out what genes and environmental factors contribute to diseases. There are multiple studies conducted on individuals or groups of genes and how they impact health and disease. Thus, there is no lack in the amount of data to draw conclusions from and researchers have the potential to make more connections from within the data.

An issue arises when it takes a long time to find and sift through biomedical research papers to find knowledge useful to researchers. There are already ideas present in literature, but also a lack of finding a connection between already present ideas. Researchers are constantly looking to find these connections; however, there is a limiting factor of time and can be overwhelming without the right tools.

Data from text mining biomedical texts was warehoused through the College of Engineering HPC or AWS DynamoDB and AWS Redshift. Data processing utilized BERT and GNormPlus models to categorize and normalize input. TigerGraph was used to represent the knowledge accumulated to visualize relationships. This project reduced the amount of time needed to search through biomedical research by making it easier to discover knowledge about various biological mechanisms using machine learning.

Chapter 1

1.1 Project Goals and Objectives

The goal of this project was to create a usable gene-to-gene knowledge graph to aid in research within the biomedical field. The application takes an input, which is a gene name, and outputs a knowledge graph surrounding the connections of diseases and topics related to the inputted gene. Project objectives consisted of ranking the relevance of the outputs, only giving related results, and giving a result in a reasonable amount of time.

1.2 Problem and Motivation

Researchers must analyze research papers and draw connections to other papers in the process of conducting research. The finding and analyzing of research papers can be a very time-consuming process. Gene research is also very scattered; there can be multiple diseases related to a single gene and finding all these connections can be very cumbersome. Our project solves both of these problems by decreasing the time for research, as well as creating a visualization to help in understanding gene relationships.

1.3 Project Application and Impact

Genetic research studies how individual genes or groups of genes affect health and disease. By improving research times and offering more relevant research surrounding the topic of genes, our project will act as a tool for the creation of new technologies or discoveries that will benefit the well-being of society and the prevention and treatment of diseases.

1.4 Project Results and Deliverables

Our project's results are a knowledge graph over genes and their relationships. The output is a knowledge graph with a label of their connections to a specific gene, which is entered as an input for the tool. Project deliverables consist of eight deliverables.

Deliverable 1: Research problem understanding and preliminary research

The research goal is to find the best correlation papers for an input biomedical gene paper for a map of the most correlated papers and their references. Preliminary research is currently being conducted in machine learning on text mapping and name entity recognition and biomedical research papers to find similar vocabulary for some pre-mapping for training the model later.

Deliverable 2: Collecting data

Biomedical research papers on gene and abstract data will be collected through PubMed and Gene. These abstract data will be mass-collected and analyzed using pre-trained / predefined models like GNormPlus for gene name recognition.

Deliverable 3: Pre-processing data

The data from Pubmed is formatted in either XML or TXT. This data will be stored in the HPC cluster with additional storage power by AWS. Currently, the project will utilize DynamoDB for noSQL data as the data input could become more diverse (Redshift for data warehousing) This data will be stored and automatically processed using python script with some third-party library besides Pandas. The data then needs to be pre-processed to become unified xml or json data.

Deliverable 4: Name entity recognition and relation extraction

After preprocessing data, the data will be processed using BioBERT for text mining training on HPC Cluster to summarize research papers and further create wording correlation mapping on words and weight parameters for graph database construction (maybe the frequency of having similar words between two paper could become the correlation weight scale between two paper). Other parameters like a title / abstract / past collaboration or citations of input paper can be utilized.

Deliverable 5: Knowledge Representation

Once the weight correlation is determined for the connections between papers, TigerGraph will be used to initiate a graph database. This will be implemented for visualizing recommendation research papers with a weight scale to reduce preliminary research time and increase research productivity.

Deliverable 6: Model Evaluation

The model base will be evaluated and compare the model output statistics to some model / service out there in the market on research paper recommendation.

Deliverable 7: Model Deployment

After doing quality assurance, the model was deployed on HPC.

Deliverable 8: Driving insights and generating Tableau dashboard

Tableau dashboards can be created to compare metrics between the model produced and other models on the market for performance comparison as well as the current model and previous performance to make decisions based on insights to change the machine learning model parameters for improvements.

Chapter 2. Background and Related Work

2.1 Background and Technologies

Background

The problem we solved is how to connect the dots between gene-to-gene relationships by utilizing machine learning with data from biomedical research papers. Discovering and creating a relationship network between a mass amount of research papers would not only accelerate the research process but also make research more intuitive and fun.

Design Patterns

For fast and intuitive implementation, we decided to design this system by implementing a market-ready machine learning model like BERT and GNormPlus for unsupervised learning with the intent to learn and encode our massive text data for vectorizing and mapping words in order to map relationships between one research document and another. For data / feature engineering pipeline, we will discuss in depth in the solution architecture section

Asynchronous Programming

In order to implement our project aychoronously, we utilized the College of Engineerings HPC to run jobs using `srun` and `sbatch`. This method allowed us to run multiple programs at the same time to make significant progress without being tied up with processing power on our local machines.

Project Estimation

The maximum number of data entries from the PubMed database is 34 million citations and abstracts of biomedical literature. While we did not intend to utilize this entire database, this proves that we have more than enough data points for training, validation, and testing cycles in our machine learning model. Our machine learning models are open source of BioBERT and GNormPlus for natural language processing and they were computed by the College of Engineering HPC.. Later on, we migrated the database to TSQL and utilized TigerGraph for graph representation of our data.

We estimated that the first model available would be mid-February 2023, and the production-level model would be be available by early May 2023 for showcasing and academic research purposes.

Scientific and Mathematical Models

Our entire model idea is wrapped around the concept of word embedding, more specifically how to create a contextual representation of each research paper abstract and map these representations to know the nearest related paper to references and conduct research on similar topics.

Technology

We collected data through HuggingFace Datasets such as `pubmed` and `pubmed-summarization`, use pre-trained models like BioBERT and GNormPlus for gene name recognition and relation extraction, HPC for data storage, and Tiger Graph for knowledge representation.

HuggingFace:

Hugging Face is an open-source natural language processing library open to all ML models with support from libraries like Flair, Asteroid, ESPnet, Pyannote, and more to come. HuggingFace has a library called Datasets that allows quick loading and manipulation of large datasets as well as models available for fine-tuning.

BioBERT & GNormPlus:

Bidirectional Encoder Representations (BERT) is a transformer-based machine learning technique for natural language processing pre-training developed by Google. This will be paired with a gene recognition tool and can be applied to PubMed to filter the results and apply operations to determine a weighted score for correlations between connected papers.

TigerGraph:

A native parallel graph database purpose-built for loading massive amounts of data and analyzing their relationships to create connections between papers.

High Performance Computation (HPC):

Computing system available through San Jose State University with multi-core multi-socket servers, high performance in regards to storage, GPUs, and a significant amount of memory available on an

interconnected network.

2.2 Literature Search

Literature searches in biological science and bioinformatics are more complicated than most fields because of the numerous and complex components like DNA, RNA, Gene, and Proteins. The article Bi-directional Methodology for Literature Extraction from PubMed Abstracts using Web Scrapper and Web Crawler applies a Bi-directional methodology to extract data that comes from the abstracts available on Pubmed obtained from a web scrapper and web crawler, then mapping the genes from that data to the Gene Card database [6]. The article also applies a second method that uses Gene Expression profiles that are analyzed with exploratory data analysis (EDA) to map the gene data extracted from Pubmed to a Gene Card database. Performing Principal Component Analysis (PCA) and clustering using k-means are used to group the differentially expressed genes. The experiment uses the web crawler called scrappy, available as a python library, to extract data from Pubmed's abstracts. It then extracts the names of genes using regular expressions. The experiment targets lung cancer genes that are extracted from the data by comparing to a Gene Card database, then the Gene Profiles are analyzed and mapped to the Pubmed abstracts. The article Building a PubMed knowledge graph explores how Pubmed abstracts can be used to create a knowledge graph [10]. For a Named Entity Recognition (NER) tool, researchers used a biomedical text mining tool called BioBERT based on Bidirectional Encoder Representations from Transformers (BERT) for bio-entity extraction. The data was then normalized using multi-type normalization, including GNormPlus for genes and proteins. Researchers also needed to extend multi-source information integration in order to build relationships to complete a comprehensive overview of the PubMed dataset. Data from NIH ExPORTER, ORCID, and fine-grained region and location information from the MapAffil 2016 was used to integrate data. The knowledge graph produced facilitates researcher-centric and bio-entity-centric activity.

Accurate identification of gene and protein names is an essential step in many commercially highly relevant applications, such as patent retrieval, prior art search, or patent classification [3]. Testing was run on the precision of tools in identifying correctly recognized gene names or identifiers by the total number of annotated gene entities or identifiers of a document averaged over a collection of documents. Based on their execution times, the ranking order from slow to fast is Gimli, GNormPlus, GeneTuKit, and GNAT. Results show that in a performance comparison between GNAT, Gimli, and GNormPlus, GNormPlus performs the best out of the rest when taking the recall and F-measure results (based on test accuracy), while GNAT performs better on precision values and runs faster.

The article Drug Repurposing Using TigerGraph & Graph Machine Learning uses the latest graph database and machine learning tools to create a Drug Repurposing Knowledge Graph (DRKG) by using repositioning, reprofiling redirection, and rediscovery in the investigation of existing drugs for new applications in medicine [8]. DRKG is a knowledge graph that links related genes, compounds, diseases, biological processes, side effects, and symptoms. The researchers used machine learning and a graph database TigerGraph to gain knowledge about drug repurposing, specifically targeting genes that are linked to a particular disease that interact within cells that are affected by genes that are targeted by other drugs with a focus on the biological components specific to Covid-19. The DRKG process consists of creating a TigerGraph schema, importing data in TSV format, querying the data, and then visualizing the graph and fine-tuning the model and predictions. The machine learning framework PyTorch was used for link prediction since researchers wanted to utilize six built-in datasets containing gene knowledge: DrugBank, Hetionet, GNBR, String, IntAct and DGIdb. The biggest challenge in the implementation of DRKG was time and resources since it was computationally heavy. In an effort to reduce time, researchers used pre-trained models and fine-tuned them to fit the specific purpose of discovering edges between compounds, disease, and gene-to-gene interactions that previously were not recognized. The graph database was also deployed on TigerGraph cloud and used REST interface on the large volume of data. Requirements included the integration of Nvidia for

embedded graph machine learning, using Spark or Dask for processing to complete jobs, allowing for graph transformations and projections using tools such as Neo4j, and implementing the business requirements of the machine learning algorithms.

State-of-the-art

The USA National Center for Biotechnology Information (NCBI) contains information on 27.5 million journal articles. “The NCBI Nucleotide Database (including GenBank) had data for 243.3 million different sequences and dbSNP described 997.3 million different genetic variants” [11]. Pubmed entries can be searched using up to 272 thousand unique terms. The NCBI also provides access to a total of 50 databases through a web interface and online API. The eUtils API provides endpoints for searching each of the databases it covers and can find cross-references among those records and fetch particular records. This is just one of many reliable sources for biotechnology information and is the largest database covering this topic.

GNAT and Gimli have been known as the baseline tools with the state-of-the-art performance on scientific articles [4]. GeneTuKit works using four main node modules. The first module is for gene mention recognition, the second is for gene ID candidate generation, the third is for gene ID disambiguation to distinguish between genes, and the fourth module is for generating a confidence score for each predicted gene ID. GNAT and GeneTuKit have reasonable execution times, when scaled up, GeneTuKit takes 9 times as long as GNAT. However, GNormPlus and Gimli take an impractical amount of time when scaled up. At the document level, GnormPlus performs the best; however, when scaling up then GNAT gives the most precision within a reasonable amount of time.

To visually represent a knowledge graph, applications such as TigerGraph are used as a backend graph database to store the knowledge graph and the newly discovered relationships together with some graph machine learning techniques.TigerGraph has recently been used in a Drug Repurposing Knowledge Graph (DRKG) to generate visual representations for newly found gene relations. Drug Repurposing Knowledge Graph (DRKG) is a comprehensive biological knowledge graph relating genes, compounds, diseases, biological processes, side effects and symptoms.[10]. DRKG is freely available to import into our TigerGraph database for the general purpose of recreating new graphs for the gene of interest.

Chapter 3 Project Requirements

3.1 Domain and Business Requirements

[Use UML 2 activity diagram to draw process summary diagram and a set of process decomposition diagrams. Draw a domain class diagram of business classes with attributes; draw a set of state machine diagrams for key business classes.]

3.2 System (or Component) Functional Requirements

Functional Requirements

No.	Functional Requirement	Essential/Desirable/Optional
1	Should extract data from Pubmed	Essential
2	Shall use APIs from sources besides Pubmed: - Semantic Scholar - Hetionet - IntAct - STRING - DGIDB	Desirable

No.	Functional Requirement	Essential/Desirable/Optional
3	Shall generate a list of related papers	Optional
4	Should pre-process data	Essential
5	Should Use BioBERT to recognize gene relationships	Essential
6	Shall Use GNormPlus to normalize gene names	Optional
7	Shall Use other NLPs to recognize gene relationships	Optional
8	Shall use other NLPs to normalize gene names	Optional
9	Should create model trained from extracted data	Essential
10	Should create graph using TigerGraph	Essential

Table 1: Project Requirements

Extracting data from Pubmed is an essential requirement because that is where we are sourcing the biotechnology information to create the training set and the testing set. It would also be desirable to use APIs from sources besides Pubmed to have more diversified training and testing sets and increase our credibility. One of the goals of our project is to generate a list of related papers of one genome, therefore it is an essential requirement of the project. The pre-processing of data is essential because we need to convert the extracted data from the xml and txt format into solely xml or json format to make it usable. Through Natural Language Processing (NLP), the document can be scanned for gene names and will be used inside the model in order to generate the list of related papers. The last essential requirement is to create a correlation between papers based on their weights using TigerGraph, making the graph scalable would create a result in a reasonable amount of time allowing more correlated papers to be listed at a given time. We also defined an optional requirement in that the user will receive the results in an appropriate amount of time related to their request.

3.3 Non-function Requirements

Non-functional Requirements

No.	Non-Functional Requirements	Essential/Desirable/Optional
1	The system commands shall deliver a response within 3 seconds of requesting an output.	Essential
2	The graph database shall be scalable	Essential
3	The system shall be available	Essential
4	The system shall be secure	Essential

Table 2: Non-Functional Requirements

3.4 Context and Interface Requirements

[Specify the context environments supporting your development, testing, and deployment of your project results. You also need to describe the interface requirements for your hardware/software components and system.] | No | Context and Interface Requirements | Essential/Desirable/Optional

| |—-|————————————————————-|————————————| | 1 | The knowledge graph should run on any operating system. | Essential | | 2 | The knowledge graph should be hosted on Tigercloud | Essential | | 3 | BioBERT model shall be trained on HPC | Essential | | 4 | TigerGraph should be used to store gene relationship data | Essential | | 5 | Web Application shall be used to display knowledge graph | Desireable | ##### Table 3: Context and Interface Requirements {#tbl:context-reqs}

3.5 Technology and Resource Requirements

No	Technology and Resource Requirements	Essential/Desirable/Optional
1	The system shall use a minimum of 1 compute node on HPC	Essential
2	The graph database should handle 35 Million entries	Essential

Table 4: Technology and Resource Requirements

Chapter 4. System Design

4.1 Architecture Design

4.1.1. Architecture Design

Figure 1: Project Architecture

1. Data Extraction: pulling in biotechnology information from the various APIs listed.
2. Exploration and Validation: focuses on the content and the structure of the data. The goal is to have a set of metadata in xml or json file format. The data needs to be structured in a way to be homogenous to one another to make it usable in SQL.
3. Data Wrangling: clean the data, reformatting particular attributes or correcting errors in the data.
4. Data Storing: The data is stored in the SQL Data Warehouse
5. Data Splitting: At the data warehouse, the data is split into training and validation/testing datasets to be used in producing the machine learning model.
6. Model Engineering: The machine learning algorithm is applied to the training data to train a machine learning model, which includes feature engineering, using domain knowledge to extract features from the data, and hyperparameter tuning to maximize the model's performance for better results.
7. Model Evaluation: running the model with the validation and testing data to ensure it is suitable for production.
8. Model Packaging: exporting the model in a specific format to be used in the final application.
9. Model Deployment: Deploying the application using HPC or online cloud computing to host the application.
10. Model Visualization: use a weighted graph or vectors to determine what search results show for the application in the outputted results.
11. Model Performance monitoring and logging: fine tune our model using feedback or reviewing the results in performance and visualization of the application.

4.1.2. Project Design

There are five necessary design elements to implement the project. ##### Data extraction: HuggingFace For data extraction, we have decided to use PubMed abstracts as a data source. The information can be gained from the datasets available on HuggingFace for PubMed.

Gene Name Recognition: BioBERT

For gene name recognition, we have decided to use Bidirectional Encoder Representations from Transformers for Biomedical Text Mining (BioBERT) [7]. BioBERT is a pre-trained biomedical language representation model for biomedical text mining. It is built off of Bidirectional Encoder Representations from Transformers (BERT) with the specific purpose of text-mining biomedical data. It is initialized with weights obtained from BERT. It is then pre-trained with biomedical data obtained from PubMed. BioBERT must be fine-tuned for the Name Entity Recognition and Relation Extraction necessary to implement the project.

Normalization: GNormPlus

For normalization, we are using the tools GNormPlus for genes/proteins and sieve-based entity linking for diseases. GNormPlus: GNormPlus resolves composite gene and protein names using advanced text-mining techniques [9]. The normalization step uses GenNorm, a generalized normal continuous random variable, a simplification tool for composite names, and an abbreviation tool to handle abbreviated gene names.

Graph Database: TigerGraph

For the graph database, our project uses TigerGraph. TigerGraph is built on a native graph database and implements an SQL-like query language [1]. TigerGraph can show how entities, such as genes, are related to each other. Knowledge can be gained about gene-to-gene relationships through a graph created in TigerGraph. The resulting graph can be hosted on Tiger Cloud.

High Performance Computing (HPC)

A high performance computing system is available to use through the San Jose State University College of Engineering (CoE) [4]. This system is necessary to implement all the computationally complex tools to extract, text-mine, and normalize our data.

Figure 2: Project Design - Gene-to-Gene Knowledge Graph Framework

4.2 Design Constraints, Problems, Trade-offs, and Solutions

4.2.1 Design Constraints and Challenges

Computational Complexity

There are challenges in running a gene recognition tool on a high quantity of data that we plan on running it on. A choice in different tools can yield more accurate results but take an exponential amount of time. Therefore, we need to ensure our project utilizes a combination of tools that yields sufficient results in a reasonable time.

Computational Power

Our project is very computationally heavy so we need to use the College of Engineering's High Performance Computing (HPC) system. If the HPC system is not sufficient enough in terms of processing power, we will need to seek other cloud computing systems,

Data Processing Requirements

The challenges of data processing are a set rate on the amount of data pulled from APIs and a large storage requirement. For example, The pubmed APIs allow us to retrieve only 10 requests/second. Each paper that is retrieved from the API needs to be stored to be processed, which requires a large amount of storage space.

4.2.2 Design Solutions and Trade-offs

Computational Complexity

GNormPlus and BioBERT are more computationally complex than its alternatives. GNormPlus requires more comparisons to be made with mapping words to gene nomenclature, parsing through text and keeping track of genes within a paper. If the time it takes to run GNormPlus is not within a reasonable time for searching data (within a few hours) or it is too performance intensive, then we will have to switch to less computationally complex tools, such as GNAT, that give up precision in gene name recognition for faster processing times.

Computational Power

Our team believes using the College of Engineering's High Performance Computing (HPC) system will provide the resources necessary to complete the processes for data extraction and processing. In the future, if our team runs into a problem with pricing for storage or computational capacity, then we should be able to find a more affordable alternative for cloud computing services.

Data Processing Requirements

Pubmed API allows us to retrieve up to 10,000 results and requests without an API key have a rate limit of 3 requests/second. Requests with an API key have a default rate limit of 10 requests/second. Our group will consider this amount of data as acceptable for now. However, if we want a larger amount of dataset, we can improve the algorithm that we use for the API. Funds are available to reimburse the cost of AWS if necessary.

[Chapter 5 System Implementation]

5.1 Implementation Overview

In order to create a knowledge graph representing gene-to-gene relationships, we needed to find a source of data, a machine learning model to get relationships from the data, a way to graph these relationships, and enough processing power to handle these tasks. We are able to achieve this by using the College of Engineering's High Performance Computing (HPC).

The methodology framework is based on the design from "Building a pubmed knowledge graph," [4]. Instead of relating the nodes by Pubmed entries, the nodes will show gene-to-gene relationships through the relation extraction functionality available in BioBERT [1].

Once we have trained our BioBERT model on the Pubmed data, we will use a TigerGraph schema to map relationships and host our graph on TigerCloud.

Data Extraction

For data extraction, we used PubMed abstracts as a data source. To implement this, we are using datasets from HuggingFace. HuggingFace datasets are available in the library called Datasets and can

be split into the train, validation, and test sets. HuggingFace was selected because its datasets are easy to manipulate for our intended purpose and contain all the abstracts available on PubMed. Pubmed contains about 35 million articles, which are readily available through HuggingFace

Named-Entity Recognition, Relation Extraction and Normalization

For named-entity recognition and relation extraction, we used BioBERT pre-trained models on the PubMed datasets and GNormPlus for gene name normalization. We used pre-trained models available through BioBERT on HuggingFace and fine-tuned them using datasets available from "Tagging genes and proteins with BioBERT" [2]. For BioBERTs relation extraction, we use a rule-based approach that will look for gene-to-gene interactions along with relationship types such as passive, reactive, and active.

Graph Database

For the graph database, our project uses TigerGraph. For this portion, we will base it on the Drug Repurposing Knowledge Graph [3] and Pubmed Knowledge Graph [1] that we will fine-tune to meet our specifications and schema.

5.2 Implementation of Developed Solutions

We have implemented the data extraction by using the HuggingFace datasets. From these datasets, we have created train, test, and validation sets. Using the pre-trained BioBERT model, we have trained the model using our test sets with a semi-supervised approach to analyze the model. We calculated the F1 score of the model and deemed it appropriate to continue the rest of implementation. After fine-tuning the model, we used it to run relation extraction on the datasets and find the gene-to-gene interactions and relationships. We then created a knowledge graph utilizing TigerGraph to plot the interactions and relationships we found on the genes. 5.3 Implementation Problems, Challenges, and Lessons Learned

Implementation Problems and Challenges

The problems and challenges of implementation include having to workaround the potential downtime of the HPC and connecting the output of the name entity recognition with the relation extraction.

Lessons Learned

Lessons we learned are that we should have started implementation earlier to have more time to work through the problems. As well as researched more on similar issues that we had before the implementation phase of the project.

Chapter 7 Testing and Experiment

7.1 Testing and Experiment Scope

In order to test our project, we will use classic machine learning metrics to evaluate the performance of our model on test datasets including F1 Score, precision, and recall. These metrics will help determine how well our model performs. BioBERT provides datasets for Relation Extraction that we will use to evaluate our model.

We will also evaluate the user interface of our Gene Knowledge Graph by assessing the time it takes to display results, the relevancy of the results, and how readable to graph is.

7.2 Testing and Experiment Approach

Since we will not be able to evaluate the model on the unlabeled PubMed data, we will use a semi-supervised model that uses both labeled and unlabeled data to evaluate the model. The metric used to assess the quality of the model is the F1 score, generated by using the precision and recall values.

$$F1Score = \frac{Precision * Recall}{(Precison + Recall)} \tag{1}$$

Precision is calculated by dividing the true positives by anything that was predicted as positive. A true positive is a positive predicted value with a positive actual value. A false positive is a value that was predicted as positive but with a negative actual value.

$$Precision = \frac{TruePositive}{(TruePositive + FalsePositive)} \tag{2}$$

Recall is calculated by dividing the true positives by anything that should have been predicted as positive. A false negative is a negative predicted value with a positive actual value.

$$Recall = \frac{TruePositive}{(TruePositive + FalseNegative)} \tag{3}$$

The closer the F1 score is to 1, the more accurate the result. The model that has the highest F1 score will be used in the project for generating the knowledge graph.

In order to evaluate the Gene Knowledge Graph, we will observe the clarity of the graph, the time it takes for the graph to display the results for a specified gene, and the relevancy of the results. We will select 10 random genes to observe these graph characteristics. We will measure in seconds the time it takes to display the results. For graph clarity, we will provide an analysis of whether the nodes and edges were easy to interpret. For relevancy, we will choose one relationship of the graph to fact-check for accuracy.

7.3 Testing and Experiment Results and Analysis

[Describe testing and experiment results and analysis. For example, test execution and test result summary, performance test result analysis, test coverage, bug distribution report, and so on. This section must include textual description accompanied with figures and/or tables.]

Here we will analyze

Relation Extraction Model Results

Model	F1 Score	Precision	Recall

Knowledge Graph Results

Gene	Result relevancy	Time to Display (sec)	Graph Clarity
	The results were relevant/not relevant	xx:xx	The graph displayed clear nodes and edges displaying the relationships

This manuscript is a template (aka “rootstock”) for Manubot, a tool for writing scholarly manuscripts. Use this template as a starting point for your manuscript.

The rest of this document is a full list of formatting elements/features supported by Manubot. Compare the input (`.md` files in the `/content` directory) to the output you see below.

Basic formatting

Bold text

Semi-bold text

Centered text

Right-aligned text

Italic text

Combined *italics and* ***bold***

~~Strikethrough~~

1. Ordered list item
2. Ordered list item
    a. Sub-item
    b. Sub-item
        i. Sub-sub-item
3. Ordered list item
    a. Sub-item

- List item
- List item

- List item

subscript: H_2O is a liquid

superscript: 2^{10} is 1024.

unicode superscripts^{0123456789}

unicode subscripts$_{0123456789}$

A long paragraph of text. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Putting each sentence on its own line has numerous benefits with regard to editing and version control.

Line break without starting a new paragraph by putting
two spaces at end of line.

Document organization

Document section headings:

Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

A heading centered on its own printed page

Horizontal rule:

`Heading 1`'s are recommended to be reserved for the title of the manuscript.

`Heading 2`'s are recommended for broad sections such as *Abstract*, *Methods*, *Conclusion*, etc.

`Heading 3`'s and `Heading 4`'s are recommended for sub-sections.

Links

Bare URL link: https://manubot.org

Long link with lots of words and stuff and junk and bleep and blah and stuff and other stuff and more stuff yeah

Link with text

Link with hover text

Link by reference

Citations

Citation by DOI [1].

Citation by PubMed Central ID [2].

Citation by PubMed ID [3].

Citation by Wikidata ID [4].

Citation by ISBN [5].

Citation by URL [6].

Citation by alias [7].

Multiple citations can be put inside the same set of brackets [1,5,7]. Manubot plugins provide easier, more convenient visualization of and navigation between citations [2,3,7,8].

Citation tags (i.e. aliases) can be defined in their own paragraphs using Markdown's reference link syntax:

Referencing figures, tables, equations

Figure 1

Figure 2

Figure 3

Figure 4

Table 1

Equation 4

Equation 5

Quotes and code

> Quoted text

> Quoted block of text
>
> Two roads diverged in a wood, and I—
> I took the one less traveled by,
> And that has made all the difference.

Code `in the middle` of normal text, aka `inline code`.

Code block with Python syntax highlighting:

```
from manubot.cite.doi import expand_short_doi

def test_expand_short_doi():
    doi = expand_short_doi("10/c3bp")
    # a string too long to fit within page:
    assert doi == "10.25313/2524-2695-2018-3-vliyanie-enhansera-copia-i-
        insulyatora-gypsy-na-sintez-ernk-modifikatsii-hromatina-i-
        svyazyvanie-insulyatornyh-belkov-vtransfetsirovannyh-geneticheskih-
        konstruktsiyah"
```

Code block with no syntax highlighting:

```
Exporting HTML manuscript
Exporting DOCX manuscript
Exporting PDF manuscript
```

Figures

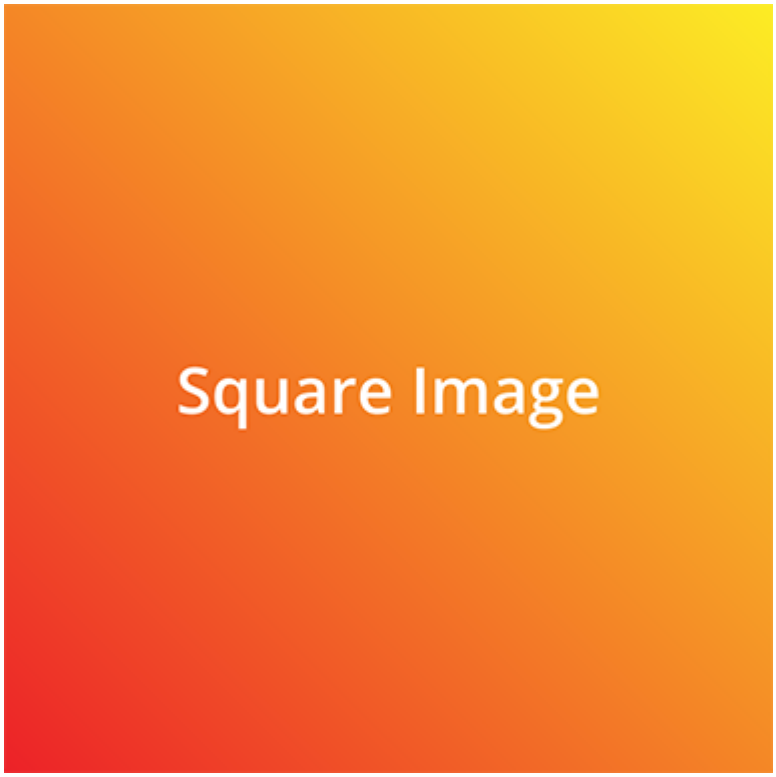

Figure 1: A square image at actual size and with a bottom caption. Loaded from the latest version of image on GitHub.

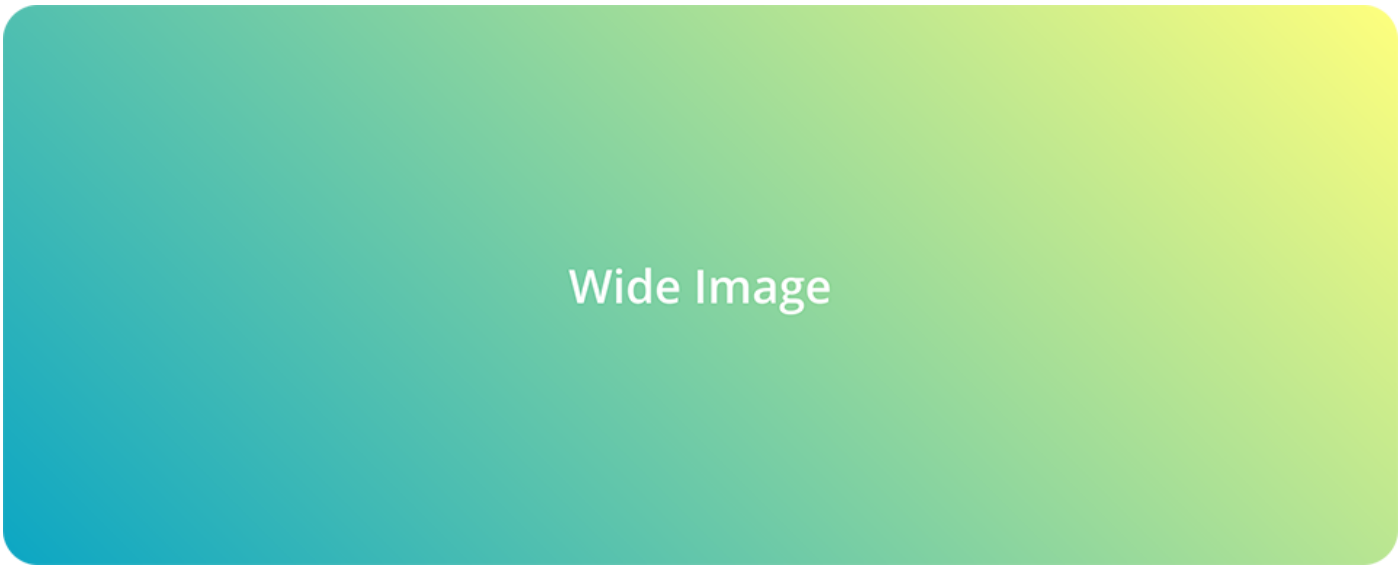

Figure 2: An image too wide to fit within page at full size. Loaded from a specific (hashed) version of the image on GitHub.

Figure 3: A tall image with a specified height. Loaded from a specific (hashed) version of the image on GitHub.

Figure 4: A vector `.svg` image loaded from GitHub. The parameter `sanitize=true` is necessary to properly load SVGs hosted via GitHub URLs. White background specified to serve as a backdrop for transparent sections of the image. Note that if you want to export to Word (`.docx`), you need to download the image and reference it locally (e.g. `content/images/vector.svg`) instead of using a URL.

Tables

Table 1: A table with a top caption and specified relative column widths.

Bowling Scores	**Jane**	**John**	**Alice**	**Bob**
Game 1	150	187	210	105
Game 2	98	202	197	102
Game 3	123	180	238	134

Table 2: A table too wide to fit within page.

	Digits 1-33	**Digits 34-66**	**Digits 67-99**	**Ref.**
pi	3.14159265358979323 846264338327950	28841971693993751 0582097494459230	78164062862089986 2803482534211706	`piday.org`
e	2.71828182845904523 536028747135266	24977572470936999 5957496696762772	40766303535475945 7138217852516642	`nasa.gov`

Table 3: A table with merged cells using the `attributes` plugin.

	Colors	
Size	**Text Color**	**Background Color**
big	blue	orange
small	black	white

Equations

A LaTeX equation:

$$\int_0^\infty e^{-x^2} dx = \frac{\sqrt{\pi}}{2} \tag{4}$$

An equation too long to fit within page:

$$\begin{aligned} x = a + b + c + d + e + f + g + h + i + j + k + l + m + n + o + p + q + r + s + t \\ + u + v + w + x + y + z + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 \end{aligned} \tag{5}$$

Special

⚠ **WARNING** *The following features are only supported and intended for* `.html` *and* `.pdf` *exports. Journals are not likely to support them, and they may not display correctly when converted to other formats such as* `.docx`*.*

LINK STYLED AS A BUTTON

Adding arbitrary HTML attributes to an element using Pandoc's attribute syntax:

Manubot Manubot Manubot Manubot Manubot. Manubot Manubot Manubot Manubot. Manubot Manubot Manubot. Manubot. Manubot. Manubot.

Adding arbitrary HTML attributes to an element with the Manubot `attributes` plugin (more flexible than Pandoc's method in terms of which elements you can add attributes to):

Manubot Manubot Manubot Manubot Manubot. Manubot Manubot Manubot Manubot. Manubot Manubot Manubot. Manubot. Manubot. Manubot.

Available background colors for text, images, code, banners, etc:

`white` `lightgrey` `grey` `darkgrey` `black` `lightred` `lightyellow` `lightgreen` `lightblue` `lightpurple` `red` `orange` `yellow` `green` `blue` `purple`

Using the Font Awesome icon set:

✔ ? ★ 🔔 ⊗ •••

Light Grey Banner
useful for *general information* - manubot.org

Blue Banner
useful for *important information* - manubot.org

Light Red Banner
useful for *warnings* - manubot.org

References

1. **Sci-Hub provides access to nearly all scholarly literature**
Daniel S Himmelstein, Ariel Rodriguez Romero, Jacob G Levernier, Thomas Anthony Munro, Stephen Reid McLaughlin, Bastian Greshake Tzovaras, Casey S Greene
eLife (2018-03-01) https://doi.org/ckcj
DOI: 10.7554/elife.32822 · PMID: 29424689 · PMCID: PMC5832410

2. **Reproducibility of computational workflows is automated using continuous analysis**
Brett K Beaulieu-Jones, Casey S Greene
Nature biotechnology (2017-04) https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6103790/
DOI: 10.1038/nbt.3780 · PMID: 28288103 · PMCID: PMC6103790

3. **Bitcoin for the biological literature.**
Douglas Heaven
Nature (2019-02) https://www.ncbi.nlm.nih.gov/pubmed/30718888
DOI: 10.1038/d41586-019-00447-9 · PMID: 30718888

4. **Plan S: Accelerating the transition to full and immediate Open Access to scientific publications**
cOAlition S
(2018-09-04) https://www.wikidata.org/wiki/Q56458321

5. **Open access**
Peter Suber
MIT Press (2012)
ISBN: 9780262517638

6. **Open collaborative writing with Manubot**
Daniel S Himmelstein, Vincent Rubinetti, David R Slochower, Dongbo Hu, Venkat S Malladi, Casey S Greene, Anthony Gitter
Manubot (2020-05-25) https://greenelab.github.io/meta-review/

7. **Opportunities and obstacles for deep learning in biology and medicine**
Travers Ching, Daniel S Himmelstein, Brett K Beaulieu-Jones, Alexandr A Kalinin, Brian T Do, Gregory P Way, Enrico Ferrero, Paul-Michael Agapow, Michael Zietz, Michael M Hoffman, … Casey S Greene
Journal of The Royal Society Interface (2018-04) https://doi.org/gddkhn
DOI: 10.1098/rsif.2017.0387 · PMID: 29618526 · PMCID: PMC5938574

8. **Open collaborative writing with Manubot**
Daniel S Himmelstein, Vincent Rubinetti, David R Slochower, Dongbo Hu, Venkat S Malladi, Casey S Greene, Anthony Gitter
PLOS Computational Biology (2019-06-24) https://doi.org/c7np
DOI: 10.1371/journal.pcbi.1007128 · PMID: 31233491 · PMCID: PMC6611653