Literature Mining of Antibiotic Resistance

Author

Seungchan Kweon

Instructor: Dr. Edward Fox

December 7, 2022
Blacksburg, Virginia

Literature Mining of Antibiotic Resistance

(ABSTRACT)

Antibiotic resistance is a very important part of human health that needs to be constantly kept track of in order to give proper medication to patients. However, the large number of papers that come out makes it very difficult for a human to keep up. Thus, this project aims to create an automated way to extract meaningful information out of antibiotic resistance papers. It aims to create an automated way to collect and parse the papers.Then, analyze the papers to extract information out of it.

Acknowledgments

I would like to thank professors Lifu Huang and Liqing Zhang for proposing and helping me out with the project.

I would like to thank the instructor for this course Dr. Edward A. Fox.

Contents

List of Figures

List of Tables

List of Abbreviations

PMC PubMed Central Identifier

PMID PubMed Unique Identifier

Chapter 1

Introduction

1.1 Clients

The 2 clients for this project are computer science professors Lifu Huang and Liqing Zhang. Assistant Professor Lifu Huang is interested in natural language processing and leads the natural language processing lab at VT. Professor Liqing Zhang is a computer science professor with background in biology, who is interested in using big data to help out biology. The project proposed is very much related to the combined interest of the two professors.

1.2 Problem

Antibiotic resistance is a global problem threatening human health. Bacteria that gain resistance to currently available medication make it harder to treat diseases every day. This needs to be considered when prescribing medications to patients. So it is very important to keep track of all the new works that come out. However with the speed at which new works come out it is extremely challenging to keep up, even for human experts. Thus the clients are looking for a way to use machine learning to analyze the works and address this knowledge challenge.

1.3 Goal

The specific goals for the project are as follows. The first is to write a series of codes required for data mining. This includes code for gathering data and parsing it. Second is to write a detailed explanation of how to use DeepEventMine, which is a tool that I'll be using for this project. Last is to write the code that will be used to analyze the outputs of DeepEventMine along with the analyzed output.

Chapter 2

Method

2.1 Timeline

The timeline for this project is given in Table 2.1

Dates	***Goal***
09/12∼09/19	Gather antibiotic resistance genes
09/19∼10/03	Gather PubMed papers
10/03∼10/31	Extract entities and events from the papers
10/31∼11/14	Analyze the entities and events
11/14∼11/28	Documentation

Table 2.1: Timeline

2.2 Gathering Antibiotic Resistance Genes

The first part of the process is to gather a list of antibiotic resistance genes to be used as search queries for later on. This step is done because the clients for this project specifically wanted to know more about genes.

Two sources were used to devise a list of antibiotic resistance genes. The first is CARD, the comprehensive antibiotic resistance database [6]. This website is mostly used to find information on a particular gene but also has a file containing the full database that can be downloaded.

The second source used is the Argminer database [2]. This is a crowd sourced database that is run by VT. It is a website where people can read medical papers, find new antibiotic resistance genes, and add them to the database.

Since there were a lot of duplicates contained in the databases some parsing was done using Python to create a list of distinct genes.

2.3 Gathering PubMed Papers

The next step is to gather the PMID, PMC, abstracts, and full papers from PubMed [7] using the genes as search queries. PubMed provides a way to get PMID, PMC, and abstracts in a single search. Searching PubMed with "format equals pubmed", results in a page that has all three information elements in a easy to parse manner. Additionally advanced search was used to get more relevant data.

The actual gathering process was done with Python using urllib and regular expressions. First a URL was made using the genes gathered from the previous step as the search term and with some additional settings. Then, a request was sent once per second, to retrieve the results. Next, regular expressions were used to parse the results.

The last step remaining is to gather the full text using the PMCs. For this step the BioC API [3] was used, because PubMed specifically says that only certain tools can be used for gathering full text articles, and BioC is one of those tools that did exactly what was required. It works with URLs similar to how PubMed works, except it gives the result in an XML format.

Actually gathering the full text was done in similar fashion regarding how the abstract was gathered, by using Python with urllib and regular expressions. Again only one query was sent per second.

Some additional parsing was done since the XML format contained a lot of additional text, and the full text contained a lot of unnecessary information like tables and references. The abstracts and the full texts were combined so if a paper had a full text file available, the abstract file was not used.

2.4 Extracting the Entities and Events

DeepEventMine is an entry and event extraction tool made specifically for medical papers [5]. It is run on Linux and uses PyTorch. It provides a pre-trained model to do the extraction. It outputs annotations that are designed to be used with brat [1].

The tool is available on GitHub and it comes with a detailed readme that explains how to set up and run the program. For the most part the instructions were good but there were some cases where things didn't work properly. The first is regarding the scikit-learn [4] version. The program needs a particular version of scikit-learn to run properly, version 0.21.3, but there is no mention of that anywhere. Second is regarding installing brat, which is the tool that is going to be used to visualize the output. The installation method mentioned in the readme no longer works, so that had to be done differently. Lastly, once the basic setup

is done there are some additional files that need to be downloaded. They provide a way to download the files automatically but it doesn't work, so the files need to be downloaded and set up manually.

The steps for running the tool are as follows. First, put the gathered papers in the data folder. Second, parse the papers using the shell script that they mention in the readme. It basically just splits the text into individual sentences. Third, create the settings file to tell exactly what will be done with the papers. Fourth, run the tool.

Finally, there are two additional problems that could happen while running the program. The first has to do with the parsing program that they give. It has an infinite loop error on some papers. It does not happen that often but it is something that has to be manually taken care of. For this project the papers that were causing the error were simply deleted. Second is a memory problem. The tool runs out of memory when running large batches of papers at once. So the inputs needed to be split into 70 batches.

2.5 Analysing the Entities and Events

The last step is to analyze the output of DeepEventMine. The tool outputs individual .ann files for each paper, that are very hard to understand by a human reader. So the results need to be combined and put in a way that is easy to understand. Also, since the clients were specifically interested in antibiotic resistance genes, some analysis will be done to give stats related to the genes.

The whole process was done using Python. First, the meaningful parts of the outputs were extracted and combined. The two main parts that were extracted are the protein names which include genes and the relations between proteins. The relations are specifically about one protein having a positive or negative effect on another protein. Second, the resulting data was filtered to be about genes in particular, giving a list of most commonly occurring genes and relations that included genes. Lastly, a count was done of which proteins commonly appear in the same paper as a particular gene.

Chapter 3

Results

3.1 Gathering Antibiotic Resistance Genes

The first step created a list of 5334 unique genes that were divided into 54 files for easier use later on. Below (see Figure 3.1) is a small list of random gene names that was collected. When ordered it could be seen that most of the genes had similar names (see Figure 3.2).

```
'QnrS2', 'mexL', 'OXA-89', 'fusB', 'OXY-2-5', 'CTX-M-115', 'OXA-2', 'TEM-152', '
aadA24', 'pbp1a', 'SRT', 'CTX-M-142', 'OXA-20', 'VIM-25', 'CMY-47', 'adeN', 'OCH
-4', 'tet(30)', 'baca', 'SHV-157', 'CMY-71', 'CTX-M-21', 'OXA-32', 'QnrB44', 'BL
1_fox', 'MdtK', 'FosK', 'TEM-118', 'BL2be_ctxm', 'TriA', 'TEM-87', 'OXA-49', 'va
nWI', 'YkkC', 'SHV-112', 'smeD', 'vanWB', 'tet36', 'SHV-81', 'TEM-112', 'y56', '
CTX-M-20', 'OpcM', 'tet34', 'oprM', 'OXA-320', 'dfrA5', 'Bcr', 'TEM-88', 'BL2a_1
', 'SHV-33', 'TEM-1', 'CMY-2', 'vanRA', 'cphA2', 'CMY-50', 'OXA-179', 'CMY-43',
'oleB', 'SHV-38', 'THIN-B', 'SHV-85', 'VEB-8', 'SHV-110', 'TEM-147', 'OXA-115',
'rif', 'OXA-245', 'MacB', 'TEM-60', 'NDM-9', 'cat', 'OCH-7', 'ACT-27', 'SIM-1',
'mexF', 'emrB', 'amrB', 'CMY-90', 'MdtN', 'LEN-14', 'CeoA', 'NmcA', 'SHV-1', 'Ks
gA', 'EreA2', 'acrA', 'qnrb71', 'KPC-10', 'ErmE', 'OXA-83', 'CMY-30', 'mdtF', 'M
dtH', 'acrB', 'SHV-49', 'TEM-33', 'GES-23', 'CGB-1', 'OXA-334', 'KPC-14', 'MOX-6
', 'pbp2', 'IND-12', "AAC(6')-Ib", 'CARB-6', 'TEM-105', 'vanRL', 'SME-4', 'mecR1
', 'vanJ', 'smeB', 'KPC-15', 'QnrB48', 'PDC-10', 'NorA', 'mexX', 'TEM-197', 'CTX
-M-6', 'OXA-255', 'OXA-1', 'macB', "AAC(6')-Ii", 'mdtG', 'tetM', 'ceoB', 'aadA15
', 'OXA-114a', 'OXA-23', 'mexM', 'CMY-5', 'OXA-80', 'BL2e_fpm', 'bcrA', 'OXA-132
', 'sul2', 'PmrA', "AAC(6')-Iq", 'CTX-M-1', 'Erm(31)', 'CTX-M-93', 'PC1', 'smeE'
```

Figure 3.1: Random gene names

```
'CMY-101', 'CMY-103', 'CMY-104', 'CMY-105', 'CMY-108', 'CMY-11', 'CMY-110', 'CMY
-111', 'CMY-112', 'CMY-113', 'CMY-114', 'CMY-115', 'CMY-116', 'CMY-117', 'CMY-11
8', 'CMY-119', 'CMY-12', 'CMY-14', 'CMY-15', 'CMY-17', 'CMY-18', 'CMY-19', 'CMY-
2', 'CMY-21', 'CMY-23', 'CMY-24', 'CMY-25', 'CMY-26', 'CMY-27', 'CMY-28', 'CMY-2
9', 'CMY-30', 'CMY-31', 'CMY-32', 'CMY-33', 'CMY-34', 'CMY-35', 'CMY-36', 'CMY-3
7', 'CMY-38', 'CMY-4', 'CMY-41', 'CMY-42', 'CMY-43', 'CMY-44', 'CMY-45', 'CMY-46
', 'CMY-47', 'CMY-48', 'CMY-49', 'CMY-5', 'CMY-50', 'CMY-51', 'CMY-53', 'CMY-54'
, 'CMY-55', 'CMY-56', 'CMY-57', 'CMY-58', 'CMY-59', 'CMY-60', 'CMY-61', 'CMY-62'
, 'CMY-63', 'CMY-64', 'CMY-65', 'CMY-66', 'CMY-67', 'CMY-68', 'CMY-69', 'CMY-7',
 'CMY-70', 'CMY-71', 'CMY-72', 'CMY-73', 'CMY-74', 'CMY-75', 'CMY-76', 'CMY-77',
 'CMY-78', 'CMY-79', 'CMY-8', 'CMY-80', 'CMY-81', 'CMY-82', 'CMY-83', 'CMY-84',
'CMY-85', 'CMY-86', 'CMY-87', 'CMY-9', 'CMY-90', 'CMY-93', 'CMY-94', 'CMY-95', '
```

Figure 3.2: Similar gene names

3.2 Gathering PubMed Papers

Gathering the PMID, PMC, and abstracts using PubMed took about 3 hours to complete. The majority of the time spent is attributed to the one second sleep in between requests. Unfortunately the majority of the genes gave no results. Out of the 5334 genes searched only 1253 genes gave a result and most of the results that were given are duplicates. Below is the list of top 50 gene names that gave the most number of results including duplicates (see Figure 3.3). In the end a total of 11487 PMID and PMID abstracts were gathered and 5563 PMC were gathered.

Gathering the full text took about 2 hours. Again the majority of the time spent is attributed to the one second sleep in between requests. In the end a total of 3784 full texts were gathered, which is less than the number of PMCs because some of the papers were not publicly available.

```
[('protein', 4546), ('Mdr', 994), ('tet', 826), ('mecA', 727), ('ErmB', 588), ('
sul1', 555), ('tetM', 500), ('tetA', 485), ('MCR-1', 368), ('sul2', 350), ('CTX-
M-15', 289), ('NDM-1', 277), ('vanA', 271), ('facT', 267), ('cat', 248), ('tetO'
, 246), ('tet(M)', 246), ('tetB', 243), ('tetW', 239), ('TEM-1', 234), ('QnrS',
226), ('qnrS', 226), ("AAC(6')-Ib", 221), ("AAC(6')-Ib'", 221), ('OXA-48', 220),
 ('L1', 178), ('aadA2', 177), ('ErmC', 172), ('ErmA', 172), ('OXA-23', 166), ('a
adA', 165), ('dfrA1', 160), ('Mbl', 154), ('QnrB', 153), ('tet(A)', 148), ("AAC(
6')-Ib-cr", 142), ('tetR', 142), ('mecC-type_BlaZ', 141), ('tetL', 140), ('tetQ'
, 137), ('tetK', 135), ('tetC', 135), ('mefA', 129), ('vanWB', 126), ('vanB', 12
6), ('tetX', 125), ('tet(O)', 123), ('tolc', 121), ('TolC', 121), ('tolC', 121)]
```

Figure 3.3: 50 genes that gave the most results

3.3 Extracting the Entities and Events

Running DeepEventMine on all the papers took about 90 hours. Since the tool was running on a laptop without a GPU the time should be far shorter in a better environment. In the end a total of 8900 non-empty .ann files were generated. Figure 3.4 is an example of a .ann file and Figure 3.5 is a visualized output using brat [1] of a different .ann file.

```
T1      Protein 1364 1368       ipaH
T2      Protein 1388 1414       invasion plasmid antigen H
T3      Protein 16702 16728     invasion plasmid antigen H
T4      Protein 16730 16734     ipaH
T5      Protein 16839 16843     ipaH
T6      Positive_regulation 1027 1036   increased
T7      Gene_expression 1355 1363       detected
T8      Gene_expression 16829 16835     source
E1      Gene_expression:T7 Theme:T1
E2      Gene_expression:T8 Theme:T5
```

Figure 3.4: .ann file example

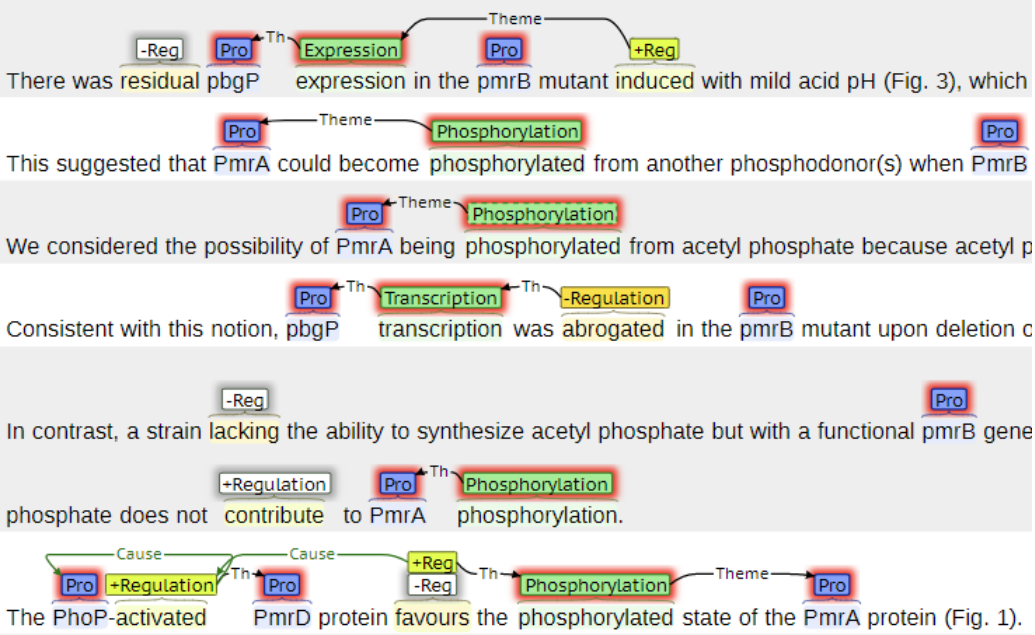

Figure 3.5: brat example

3.4 Analysing the Entities and Events

Analyzing the .ann files resulted in multiple files containing various statistics and information on antibiotic genes. In terms of statistics a total of 41283 unique proteins were annotated of which 806 of them were gene names. A total of 2735 relations were found of which 98 of them contained a gene. Figure 3.6 is the list of the 50 most commonly appearing genes and Figure 3.7 is an example of some of the genes to protein relations that was found. The relationship shows that the gene in the front has a positive or negative effect on the proteins inside the parenthesis.

```
[('mecA', 2542), ('NDM-1', 1732), ('sul1', 1538), ('OXA-48', 1026), ('sul2', 959
), ('tetA', 870), ('MCR-1', 703), ('tetM', 664), ('CTX-M-15', 638), ('qnrS', 558
), ('vanA', 494), ('KPC-2', 423), ('mgrB', 384), ('tet', 379), ('OXA-23', 376),
('tetB', 346), ('TEM-1', 336), ('sul3', 292), ('PmrA', 291), ('tet(W)', 278), ('
dfrA1', 274), ('tetQ', 272), ('tetW', 271), ('acrB', 264), ('floR', 260), ('H-NS
', 259), ('tetO', 257), ('rmtB', 254), ('cat', 247), ('marA', 246), ('mefA', 244
), ('TolC', 236), ('tet(M)', 229), ('tetC', 223), ('CRP', 219), ('ramA', 216), (
'NDM-5', 216), ('ompF', 203), ('acrA', 196), ('CTX-M-1', 189), ('vanB', 179), ('
tetK', 179), ('adeB', 179), ('tetX', 178), ('AcrB', 178), ('msrA', 176), ('VIM-2
', 175), ('optrA', 174), ('mexR', 170), ('OXA-51', 170)]
```

Figure 3.6: 50 most common genes found

```
ramA positive {'acrAB': 2, 'acrEF': 1, 'tolC': 1}
ramA negative {}
mecI positive {'mecA': 1, 'MecA': 1}
mecI negative {'mecA': 1}
mexR positive {'mexAB': 1}
mexR negative {'mexAB-oprM': 1}
mecR1 positive {'mecA': 6, 'MecA': 1}
mecR1 negative {}
ParR positive {'arn': 1, 'MexS': 1}
ParR negative {}
mexS positive {}
mexS negative {'MexT': 1, 'mexEF-oprN': 1}
QnrB positive {'DnaA': 3, 'oriC': 1}
QnrB negative {}
tetR positive {'tetA(G)': 1}
tetR negative {'murA': 1}
LmrA positive {}
LmrA negative {'LmrC': 1, 'lmrC': 3}
adeN positive {'adeFGH': 1, 'adeIJK': 1}
adeN negative {}
cat positive {'acetyltransferase': 1}
cat negative {'sly': 1}
MexR positive {}
MexR negative {'mexA': 1, 'oprM': 1}
PvrR positive {}
PvrR negative {'cupD': 3}
```

Figure 3.7: Gene to protein relations

Chapter 4

Discussion

4.1 Conclusion

In conclusion I would have to say that the project still leaves a lot to be desired. I think the code for data gathering and parsing is good and can be used for various other projects working with medical papers. The later steps of analyzing could definitely use more work or changes in the methodology.

4.2 Future Work

After having gone through the project, there are some changes and additional work that could be done to improve and extend the work. The first change is to check the antibiotic resistance gene names. As can be seen from Figure 3.3, the name "protein" gave the most amount of results, which was not intended. Also the extremely similar names that are shown in Figure 3.2 could be combined to make the searching process faster. The second change would be to add the term "resistance" and its variations when searching for the papers as there were lots of papers that were collected that only had to do with antibiotics and not with antibiotic resistance. The main portion of this work that could be extended is in the analysis part. The outputs of DeepEventMine contained lots of unfinished relations such as: <blank> has a positive effect on a particular gene. It might be possible to find out what that <blank> is through further analysis, or to find a way to make use of the unfinished relations which were ignored in this project.

Bibliography

[1] brat rapid annotation tool, 2012. URL https://brat.nlplab.org/. Accessed 5 December 2022.

[2] Antibiotic resistance gene miner database, 2019. URL https://bench.cs.vt.edu/argminer/#/home. Accessed 5 December 2022.

[3] Bioc, 2019. URL https://www.ncbi.nlm.nih.gov/research/bionlp/APIs/BioC-PubMed/. Accessed 5 December 2022.

[4] scikit-learn, 2019. URL https://scikit-learn.org/stable/. Accessed 5 December 2022.

[5] DeepEventMine, 2020. URL https://github.com/aistairc/DeepEventMine. Accessed 5 December 2022.

[6] The comprehensive antibiotic resistance database, 2022. URL https://card.mcmaster.ca/home. Accessed 5 December 2022.

[7] Pubmed, 2022. URL https://pubmed.ncbi.nlm.nih.gov/. Accessed 5 December 2022.