Drug-Drug Interaction Extraction

Joseph Killian Jr.
CS 498 Capstone Paper
Capstone Advisor: Bruno Andriamanalimanana
Computer & Information Science Department
SUNY Polytechnic Institute
Utica, NY

***Abstract* -** Information on adverse drug reactions, and specifically drug-drug interactions, is growing rapidly, and this research needs to be sorted through to pull out those relationships. Work has been done on this problem using mainly neural networks, and methods of deep learning, with results slowly improving year on year, but still not good enough to be used in practice. This paper aims to see if a simpler statistical approach could attain similar or better results. The prediction algorithm used attained results of 28% accuracy, and this is similar to others who have approached this problem from this angle. The detection algorithm results of 75% accuracy are much more promising, but do not extract as much information. The results showed that the neural approach is still superior, but this methodology has the benefit of being more transparent. With further work, and more data, this method could potentially approach the results of many of the state-of-the-art methods currently available.

I. Introduction

Adverse drug reactions are a big problem in the field of medicine. These are described as an injury that results from the taking of a medication. The estimated additional costs from these reactions range in the hundreds of billions, and adverse drug reactions account for one in five injuries or deaths to patients who are hospitalized [20]. Some of these effects are related to how an individual reacts to the drugs administered to them, but these adverse reactions can also be a result of multiple drugs being taken together and interacting negatively with one another. The probability of a negative interaction only increases with the number of drugs taken concurrently.

A major concern with the pharmaceutical industry is how new drugs will interact with ones that are already present. These are called drug-drug interactions (DDI). Some amplify the effects of drugs taken alongside, while some depress the effects of others. Others may not change any of the effects at all. According to the CDC, from the years 2015-2018, as many as 12.8% of Americans have taken 5 or more prescription drugs in the past month [14]. Unforeseen interactions could cause serious harm to the person taking the medications and increase the medical costs incurred. The amount of information on this topic is rapidly evolving every day, with the database Medline releasing between 10,000 and 20,000 papers every week [16], and another database PubMed releasing almost a million new biomedical articles every year [13]. There are a few databases available containing information on DDI, but the most up to date information is hidden away in these new publications.

The COVID-19 pandemic has also shown the rapid pace at which new research and knowledge is produced. Papers about many different types of possible treatments, methods for creating a vaccine, and the different side effects and symptoms were published every day all over the world. This creates a lot of data that no one person can possibly sort through on their own. This is where NLP can possibly help to streamline the process of finding which papers bring more value than others, and the areas where more research needs to be done. Being able to determine and compare the effectiveness of certain drugs compared to others, without needing to spend hours reading papers all day could save a lot of time.

Natural Language Processing is the technology that people interface with the most in their daily lives without even realizing it. Whether it be autocorrect when sending a text message, suggested automatic email replies, one the many digital assistants, or the ability to easily translate almost any language in the world with Google Translate. It is an integral part of society at this point, and the field has been moving at a rapid pace. The technology has been utilized heavily in many consumer products, but one area where it seems to be underutilized is the medical field, and the biological sciences in general.

There are many areas where this technology could be made useful in healthcare, and not just in regard to COVID. It could be used to help sort through the data that is produced every day in the countless number of electronic health records. They could be used to help read test results more accurately, make more accurate diagnosis, and to convert a lot of the unstructured data that is produced, into a much more normalized and readable format. There is much efficiency to be gained, so there should be great incentive for these tools to be developed, not just from the hospitals point of view, but also the many insurance companies whose business relies on healthcare.

This leads into where NLP can be applied to finding drug-drug interactions. This problem closely mirrors the general problem in NLP of Named Entity Recognition, and Relation Detection. Finding the named entities in this case is identifying the drug names and finding the drug-drug interactions is the same problem as relation detection is in other contexts. For these reasons, this seems to be a problem where NLP is very applicable. There are many methods in order to solve both of these subtasks of finding the drug names, and their relationship, but this paper aims to use a simpler statistical method to extract this information. The aim of this paper is to try and extract these DDI, using the DDI corpus produced in 2013[15,16].

II. Background

The following section is meant to give some background information regarding Natural Language Processing (NLP) and some of the basic concepts and techniques that are being used. There will also be an explanation of Information Extraction (IE) that will be gone into further detail later in the paper. There will also be a section describing some of the terms and definitions related to drug-drug interactions.

What is NLP?

A general definition of NLP is the use of software or other similar computing technology to process natural languages, which are the languages we use in everyday speech [1]. There are two major problems that any NLP application hopes to solve. One is the problem of natural language generation, which is the task of attempting to generate language that could have been spoken or written by a human being [1]. The other major task is that of natural language understanding, which is much harder to define.

Whether a computer/program understood the language that was presented to it is up to interpretation, and generally, this task is judged based on the performance of the application. A Turing test [4] can also be used, which usually involves an interrogator asking a computer and human questions and trying to guess which one is real and which one is the computer. If the two are indistinguishable then, the computer passes this test. There are many other Turing-like tests that can be used to help determine how an NLP/AI application is performing.

NLP Applications/Challenges

As one could imagine, there are countless possible applications of NLP. Anyone who uses a phone is familiar with autocorrect, and every word processing software has some kind of spelling/grammar check going on in the background while you are typing. Of the applications, these are some of the simpler ones [1]. Something like Google translate is an example of machine translation. This is a bit more challenging than the earlier ones to get working properly. The early days of google translate were sometimes very inaccurate, but the advances in attention mechanisms developed by google have helped a lot with the accuracy of translation [5]. Attention mechanisms have also found use in other areas of AI, not just in machine translation. Some of the most difficult NLP applications involve more direct communication between the system and the user. Things like interpreting speech and answering questions, or summarizing a given text [1]. This can be seen in the many assistants out there like Siri, Alexa, etc. These technologies have also made quite a lot of progress, but most people have personal experience of frustration with these technologies working incorrectly.

There are many reasons for why implementing these various applications has been difficult in the NLP community. Much of the flexibility and ambiguity that we can handle without thinking, is a much harder task for a machine [1]. They lack the common-sense and overall background knowledge that people gain from working with a language over a lifetime. Some words have the same meaning in different contexts, and there are often many interpretations for any given piece of language [1]. Natural language in contrast to things like programming languages is very unstructured due to its natural development, and it is constantly evolving everyday [1]. There is so much data out there for NLP to take advantage of but converting it into a form that is usable by software can be particularly challenging.

C. Implementing NLP

The three main approaches to NLP have been the use of rules-based systems, statistically based systems, and the more modern deep neural approach [1]. The rules-based systems use a series of complex rules that often need to be handcrafted by experts in the field of what the application is going to be used in. Earlier on in NLP's lifetime they were the best that was available, but with the sheer amount of data that is now available, they are often not viable. Statistical approaches use a variety of information theoretic measures as well as probabilistic ones to help identify patterns in the text [1]. The most common approach used in research today is the deep neural approach. This often involves the use of many layered neural networks that often use much more than the simple feed-forward neural network architecture [1]. None of these methods is objectively better in every situation than the others, but all have their place depending on the type of application.

In order to use many of the above techniques, the raw text needs to go through some sort of preprocessing. Working with raw text as long strings is very cumbersome and does not function with many of the NLP frameworks that are available. Some common preprocessing tasks are obtaining the tokens from the text and breaking these tokens up further through some type of normalization [1]. Tokens are usually words but can also be punctuation or symbols. Tokens can be simplified further by breaking words down to their roots, or by removing the stems. The text can also be parsed, which involves obtaining the structure of the text. There are many other fundamental tasks similar to these, that depending on the application, need to be completed before any real processing can occur.

Another common preprocessing step in NLP is embedding. This involves converting the raw text data into a vector of real numbers. There are many ways to do this, and some common algorithms are word2vec, GloVe, and ELMo. There are two philosophies when it comes to these embeddings, they can either be sparse and long. Or very dense and short. The sparse embeddings have many zeros and are usually learned on the fly when processing [1]. The dense embeddings have very few entries but are often learned beforehand due to the cost of calculating them [1]. This process of embedding has been the most successful method in NLP as it allows for this information to be utilized in many deep learning frameworks.

Once these embeddings have been learned, they can then be used in deep learning frameworks like Pytorch, and TensorFlow. These embeddings can be viewed as tensors, which is what these frameworks accept as input. A tensor is simply just an n-dimensional array [1]. These tensors can then be fed through a deep neural architecture in order to help train the model to create the desired outputs. Since these tensors are n-dimensional arrays, more complicated layers in the network can be used in order to add complexity in the network. This complexity is often necessary to model the complex relationships present in natural language [1].

Information Extraction

The application in particular that is being looked into for the purposes of this paper is Information Extraction. This involves turning the unstructured information embedded in texts into structured data [2]. This commonly involves the following steps. The raw text is first broken up into segments and then divided into its tokens using a tokenizer. Then some sort of tagging of the elements of the text takes place next. Then some sort of Named Entity Recognition (NER) comes next, which involves finding the named entities in the text that are of concern to the application [3], in this case it would be the names of drugs. Then finally some sort of Relation Recognition/Detection (RD) is used to help determine the relation between different entities in the text. This data is then usually placed into some sort of table or database [3].

E. Drug Drug Interactions

There are a few interactions between drugs that are of most concern to this paper, as they are the ones that are used in the DDI corpus dataset that is going to be used. One interaction is called Pharmacokinetics interaction, which is the result from the interference of drug absorption, distribution, metabolism, and/or elimination of a drug by another drug [16]. The other interaction is Pharmacodynamics interaction, which occurs when the effects of one drug are modified by another [16]. Four different relationship types are also annotated in this dataset: mechanism, effect, advice, and interaction. Mechanism is used when a pharmacokinetic mechanism of a DDI is described. Advice is used when advice is given regarding a DDI. Effect is assigned as the DDI when two drugs effect one another. Interaction is used when there is the acknowledgment of a DDI but is left more or less ambiguous [16].

There are also four entity types annotated in the corpus: drug, brand, group and drug_n. These entity types describe how drugs are referred to in their source material. The drug type is used for the generic names of medicines, and the drugs commonly referred to by their brand name fall in the brand category [16]. Groups of drugs are often described as well in these source texts, so those entities fall into this category. The last category is drug_n which refers to substances not approved for human use like toxins and pesticides [16].

III. Related Works

This section goes through some related work on not only the topic of DDI, but also on the work that is being done using information extraction in general in the healthcare domain. Each section gives a brief description of the methods and goals of the article, as well as what information or background was gained from it, and how it applies to this paper. This section is much more of a literature review to give an idea of what the current best methods and practices have been used so far to solve this problem.

DDI Extraction using an RNN [6]

The first paper that was looked at discussed how a neural network could be used to extract DDI from literature, so it was very similar to the goal of this paper. The paper proposed that a recursive network is the best fit for this task, since it has been found that the grammatical structure of natural languages is recursive [6]. It also gave some good background regarding the interactions and relations that the model will be looking for. It discussed the concept of Pharmacokinetics, which is how the body processes and excretes a drug, and the concept of Pharmacodynamics, which is how the drug affects the organism in question [6]. The paper also talked about how further preprocessing steps had a positive effect on the performance of the model [6], and how overall it outperformed some of the state-of-the-art models available at the time on the testing datasets that were used. This paper also used some domain specific embedding techniques as well as a tool that replaced some of the common names of drugs with their real names for further normalization.

Open Information Extraction [7]

This next paper had a slightly different goal with a more general information extraction in mind. It aimed to take much of the unstructured data that is produced from clinical records, and electronic health records and change it into a more structured format involving table, charts and graphs. Its main architecture involved coreference resolution, extractive text summarization, parallel triple extraction, and entity enrichment and graph representation [7]. So, the paper talked about Open information extraction which is a much more generalized task than that of DDI extraction which is more specific.

<u>IE from EHR with RNN and Contextual Embeddings</u> [8]

This paper discussed developing a system to extract information from electronic medical records using recurrent neural nets and pre-trained embeddings using ELMo. It also made use of attention mechanisms. This paper gave some useful background information into why the current most common methods are being utilized. It mentioned the shortcomings of rule-based systems, the weaknesses of feature engineering, and the benefits of pre-trained word embeddings, especially those tailored to the application. This study posed a model that combined contextual word embeddings, multitask recurrent neural networks, and attention mechanisms [8]. The multitask mechanism that they used separates the named entity recognition task into named entity discovery and named entity classification, with the classification as the primary task and discovery being the secondary task [8]. This paper was not as specific to this project as [6] but used a recurrent neural network as well. The task was slightly different in extracting info from medical records instead of looking for DDI from literature.

<u>NER and RD for Biomedical Information Extraction</u> [9]

This paper did a really good job of collecting a lot of information regarding how Named Entity Recognition and Relation detection is being carried out in the Biomedical field. This paper was a review of the current methodology as opposed to proposing a new solution to the problem. It goes through and explains the issues regarding how much literature is available, even in very specialized fields and how hard it is to stay up to date without any sort of NLP. It went through what named entity recognition is, and some of the measures used to evaluate these NLP models like precision, recall, F-score, and how they are calculated [9]. It also gave a brief description of one-hot encodings, also known as sparse encodings, and the problems they face (The loss of context, and the amount of space needed) [9]. There were a few big takeaways from this paper that were definitely beneficial. One of them was the descriptions of some of the big, state of the art, encoding algorithms. In particular Word2Vec, GloVe, fastText, and BERT/BioBERT. This explanation was particularly useful as it provided a lot more specific information on each of these encoding algorithms. This paper also, like many other papers, explained the idea of rule-based systems and their advantages and drawbacks. It also goes through and gives a good explanation of coreference and why it is an important task in regard to information extraction in general, but more specifically to relation detection and NER [9].

<u>DDI Extraction using a CNN</u> [10]

This particular paper was the only one that was looked at that exclusively utilized Convolutional Neural Networks. The DDI corpus from 2013 was the dataset for this paper and is the same dataset that was used for this project. This paper gave a good description of how CNNs work, and a cursory explanation of how the architecture of one works. This paper also described the problem of needing to remove negative instances from the data. Negative instances are situations where two drugs share the same name, if one drug is an acronym or abbreviation of another, or in general, non-interacting drug pairs [10]. It also described the four types of DDIs relative to this dataset, those being mechanism, effect, advice, and interaction [10]. This particular paper used the Order algorithms in order to train its word embeddings from abstracts from Medline [10]. This paper also utilized positional embeddings to help improve the performance of their models.

<u>Clinical Information Extraction</u> [11]

This paper is only tangentially related to the topic of drug-drug interactions and how they are extracted, but it gives a lot of background as to how NLP is being used in other areas of medicine. Things such as electronic health records, and clinical notes. It also described the problem of a general lack of knowledge about NLP in this domain in general. The papers related to EHR greatly outpace the papers related to NLP [11]. It gave potential reasons for this gap, some of them being the experts in the field of NLP having a lack of exposure to these topics. Other reasons could be a lack of collaboration between NLP experts and clinicians, and lack of available data due to privacy/HIPPA concerns. There is also hesitancy to utilize some of these deep learning/machine learning systems in the medical field [11]. How an answer was arrived at by a DL system is rarely very obvious. The decisions that were made are abstracted from the one designing the system. If there are any errors, it is also hard to find out why those errors occurred. That is why clinical NLP has been dominated by mostly older rules-based systems, which has also distanced itself even more from the general NLP research community. To combat these issues the authors proposed that there should be more cross-disciplinary collaboration and training in order to help educate NLP experts to be able to better assist in this particular field [11].

<u>Position Aware, Deep Multitask Learning, DDI Extraction</u>[12]

This paper used a Bi-LSTM, along with positional encodings in order to extract information [12]. The system is directed at finding drug-drug interactions, rather than trying to extract information from EHR in the previous paper that also utilized a Recurrent Neural Network. This paper also employed multitask learning to increase its accuracy. The first task being that of trying to predict whether two drugs interact with one another, and the second task being a further classification to distinguish what type of interaction is occurring between drugs [12]. Like many other previous papers, the need to filter out negative instances was emphasized, in order to better balance the data and obtain better results overall. Not only that, but it was also discussed how some of the information would be lost in preprocessing if a large number of instances were to be filtered out, so there needs to be some sort of balance of the two. So, the system was similar to the one that was used in [8] but it was applied to the problem of DDI.

DDI Extraction using LSTM [13]

Similar to the previous paper, the authors of this article utilized an RRN with a long short-term memory network in order to implement their DDI extraction system. This paper also provided some interesting information motivating this problem that many of the other more technical papers lacked. It included information from the CDC stating that one in 10 Americans are taking five or more medications, and also that negative consequences can both increase healthcare costs and worsen the condition of the patient [13]. It also stated that the database PubMed contains over 27 million citations and there are almost a million new articles added every year [13]. This helped to show the need for automatic extraction, as sorting through all that information manually would be extremely costly and difficult. This paper also used a different dataset to many of the ones discussed prior, using the SemEval-2013 Task 9 dataset [13]. It is very similar to the DDI corpus 2011, and 2013 as both were created from MEDLINE abstracts and the database DrugBank. Much of the technical specifications were similar to other Bi-LSTM systems that were discussed previously, but this paper gave more attention to the reason why this particular problem is particularly important to try and solve.

The DDI Corpus [16]

This paper is of particular interest as it describes the dataset that will be used for this project. It gives some valuable background information on the problems of DDI, and the importance of so-called gold standard datasets for research and development. It gives a description of Pharmacokinetics, and Pharmacodynamics, as well as a description of the entity types: drug, brand, group and drug_n, and the relations between drugs: Mechanism, Advice, Effect, and Interaction [16]. The paper also gave a description and tables regarding how the data was broken up between the two sources it was obtained from, Medline and DrugBank [16]. This paper was particularly useful for this project.

As discussed in this section, most of the current methodology relies on neural techniques, which have drawbacks that were touched upon by a few of the papers above. One problem that was touched upon by the authors of [11] was the hesitancy to use deep learning in medicine due to the need for a transparent decision-making process. The methods that will be discussed below go over a different approach, that is statistical in nature, with the goal of having higher transparency of decision making and a much simpler algorithm for classifying DDI.

IV. Methods:

The methodology used to extract interactions could best be described as a statistically based system. The goal was to create a system with a transparent decision making process that could approach the accuracy of the many neural techniques that have been proposed. On the way to this goal, there are many subtasks that all need to fit together in order for everything to function properly. This section will give an overview of how the problem was approached by breaking down each subtask, and how they contributed to trying to solve the DDI extraction problem

The first task is obtaining labeled data in order to learn more about the problem and what methods can be used to solve it. It also serves as a basis for judging the performance of the system when it is completed. For the method that was utilized, there was the need to create a comprehensive list of, ideally, every drug and type of drug that is out there. Once that is accomplished, then some preprocessing of the data, and exploration was needed in order to develop some sort of algorithm for classifying DDI. Once the algorithm is functioning to a reasonable level, then it needed to be tested out on some real papers to see how well it works, which involves extracting text from a pdf in some way. Then finally bringing that all together to have a front to back system that takes in a paper and outputs the drug-drug interactions.

Obtaining data

Many of the papers discussed all utilized the same dataset, that being the DDI corpus 2011/2013. This is an annotated corpus that contains sentences with drugs and DDIs labeled. As described in the background section, the positive instances, meaning the sentences containing a relationship, are broken up into four categories. Those being: Interaction, Mechanism, Advise, and Effect. Overall, there are around 18,000 sentences in this dataset

Obtaining the data itself posed no real issues as it was freely available. The original site that contained information regarding the challenge that was associated with this dataset no longer exists as the challenge was 8 years ago. It is now available freely on GitHub at [15] and the original paper that describes the data and how it was created is found at [16]. One of the main problems that was found with this dataset was the balance of data present. Only about 30% of the sentences have any relationship between the drugs present, the rest being sentences that either contain no DDI or no drugs at all. Even further, the positive instances in the dataset that do contain a relationship are unbalanced, with the biggest problem being the lack of interaction DDI present in the data. There are only 284 sentences containing an interaction, which only amounts to 5.6% of the positive instances. Whereas effect DDI are 41.1% of the positive instances with 2069 entries, advise DDI are 20.9% with 1050 entries, and mechanism DDI are 32.3% with 1625 entries. This caused some issues and will be discussed further in the challenges section.

B. Building the Drug List

There are many ways to tackle the task of NER (Named Entity Recognition). Some methods rely on the structure of the sentence and require some sort of POS (Parts of Speech Tagging) in order to try and identify the named entity you are looking for. There are also some prebuilt libraries that have NER built in as functions for things like names of people, locations, organizations, etc. Unfortunately, these libraries have nothing built in for identifying drugs in a given sentence.

Aiming for simplicity, the best method available to me with the time constraints was to handcraft a large list of drugs. That way each word in a sentence could be looked at to determine if it matches any of the words of the drugs list. This removed the need to do any sort of POS tagging and could achieve just the goal of finding the drugs in the sentence. There is unfortunately nothing comprehensive to be found online that serves this purpose. There were databases that had a list of certain types of drugs, but none of them had a list of all the drugs that are relevant to this project

In order to build up this list, information from various sources needed to be amalgamated. One of the first resources that was used was the FDA [18] which was surprisingly not very useful as it did not have as much information as one would hope for. Luckily there were two sources that were found that account for the bulk of the drugs list that was created. The website Drugs.com [19] had a lot of what was needed but did not have it compiled all in one place, so the method that was used was just a simple copy and paste of all the names from all the pages that made up their list of drugs. This makes up about half of the list of drugs. The other half comes from DrugBank [17] and their free database of drugs and all the different variations of their name. This source was particularly useful as it is what makes up the other half of the list of drugs and contains almost all of the drugs that are in the DDI Corpus dataset. DrugBank also provided them in an easy-to-read csv file which made pulling them all out and adding to the list very simple. Finally, the rest of the drugs in the list were taken from the training set of DDI Corpus 2013. When extracting information from the files of the corpus, the drug names from the training set of data were appended to this list.

Overall, there are 32,360 unique names in this list. Some of these names are common names for drugs, some are types of drugs, some are brand names, classes of drugs, and many other different types of names. It contains a very broad spectrum in order to be able to find drug names in as many contexts as possible. The problem with this method is new drugs come out every day, and it is unlikely that every drug that currently exists is found today. That being said, this list was able to perform well for the purpose that it was needed for.

C. Preprocessing

There are a few tasks in this project that can be described as preprocessing. One being building the drugs list, as that required some programming to merge all the sets of drugs that were found, but that was a relatively trivial part of that task. The more challenging preprocessing tasks that will be discussed here concern pulling out the data from the DDI Corpus, tokenizing the data, performing NER to find the drugs in each sentence, and sorting the sentences as positive instances or negative instances. Not all of these tasks are preprocessing in the traditional sense, but they all transform the data in a way that helped use it more effectively and learn more about the problem.

The data in the DDI Corpus is contained in xml files. There are many different libraries that can be used for extracting this information, but for the methodology that was used, it had to be extracted in a slightly different manner than the standard extraction using the libraries. Each sentence in the dataset had varying amounts of information alongside it. If it was a negative instance, then all it had was the sentence itself, or the sentence with the drugs it contained, along with a false DDI for every combination of drugs in that sentence. Positive instances always contained two or more drugs along with the relationship between those drugs. Some of these positive instances had relationships between every drug, while some had a few false relationships, and a few true ones.

The way that was chosen to pull out all this data was into a collection of tuples and lists. Drugs were paired with their relationship, and then the sentences that they originated from. So, a single sentence could have no DDIs paired with it, or it could have 50+ DDIs paired with it. It all depended on how many drugs were paired in the sentence. During this process of extracting information from the files, three sets of data were obtained. The first set was the set of all sentences in the data, the next was the list of all the drugs, and the last was the list of all sentences with the DDIs contained within them. All of these sets were used at some point in further processing.

Once the drugs list and the sentences were obtained, NER was a fairly easy task to perform. There are many libraries that do the hard parts of NER for you, those being tokenizing the data and cleaning it. Natural Language Toolkit (NLTK) was used to do any of the tokenizing that was needed, and to pull out the sentences from the corpus, or the words from a sentence. Once the words from a sentence were obtained, the words were normalized to remove any punctuation, and lower the case of all the words in a sentence. Once that was done, the task was to simply check if each word in the sentence matched any of the names from the list of drugs, then pull out those named entities, and NER for this problem was more or less completed.

One of the final preprocessing steps that needed to be completed was sorting all the sentences. This proved to be a bit more complicated than first assumed it would be. For reasons that will be discussed in the section on the algorithms that were used, Sentences needed to be sorted out into four categories, those being: positive and negative instances with a single drug pair, and positive and negative instances with any number of drug pairs.

Filtering the sentences with exactly two drugs was the easier of the two tasks. The first step was to find all the sentences that had 2 drugs in them. This required first figuring out how many drugs were in each sentence, but that was provided in the original files, so no NER was needed for this step. There are five 'bins' that these instances were sorted into. These bins were: Interaction, Advise, Mechanism, Effect, and None. The first four are related to positive instances, and the 'none' category is for the negative instances. For sorting the negative instances, it was checked whether the DDI was false for that sentence. If it is, then it gets placed in the 'none' bin. Then for sorting the positive entities. It checked the DDI type for each sentence and sorted them into their respective bins. So, if a sentence had an interaction DDI, then it got sorted into that bin. Tables 1 and 2 show the breakdown for the number of sentences in each bin. This was done for the data in both the training set and the test set

Table 1

Sentences with a Single Pair of Drugs: Train Set

Int.	Eff.	Mech.	Adv.	None	Total
18	331	174	227	840	1590

Table 2

Sentences with a Single Pair of Drugs: Test Set

Int.	Eff.	Mech.	Adv.	None	Total
5	78	40	55	251	429

Filtering the sentences with any number of drugs was a bit more complicated to complete. There were once again five bins to sort the instances into, but the logic for doing so needed to be slightly different. When sorting the negative instances in this set, sentences were added to this bin that had one drug or less, or that had a false relationship for every DDI pair present in the sentence. Then for the positive instances, the same sentence has to be allowed to be contained in multiple different bins. This is because some sentences contain multiple DDI types. So, a sentence that has an interaction and an advise DDI can be placed in both the 'advise' sentences bin, and the 'interaction' sentences bin. So, the sum of all the sentences in all these bins summed together is more than the total number of sentences used in the dataset, since there are some sentences with multiple copies. There are no duplicates in regard to the negative instances since they do not overlap with any of the positive instances. As shown in tables 3 and 4, this set of sentences was a lot larger than the set of sentences with pairs.

Table 3

Sentences with any number of Drugs: Train Set

Int.	Eff.	Mech.	Adv.	None	Total
56	838	736	485	4861	6944

Table 4

Sentences with any number of Drugs: Test Set

Int.	Eff.	Mech.	Adv.	None	Total
19	203	159	120	1477	1964

The last task to perform before really digging into the data a bit more was to take these bins of sentences and combine them all into one large string so that it could then be tokenized those sets of data for further analysis

D. Data Exploration

The goal was to try and find a simple way to classify each of these sentences, so the method that was chosen to look at the data was to find the frequency distributions of each of the different classes of sentences that had been sorted earlier.

The top 25 most frequent tokens for each set of sentences were examined. These frequent words excluded any of the drugs that were found in the list of drugs, any stop words, which are the most common words in the English language, any punctuation, and other more frequent words like 'may' and 'patients' which were frequent in this dataset and were not unique to any particular set of sentences. The reasoning for doing frequency distributions for each set of sentences was to see if there were any differences for what was frequent for sentences with just a pair of drugs or for any number of drugs. The tables 5 and 6 show the results of each of the most common words for each of the sets of sentences. The code attached associated with this project also shows the charts for these frequency distributions

Table 5

Most Frequent words for single drug pairs

Int.	Adv.	Eff.	Mech.	None.
interaction	used	effect	plasma	drugs
drug	therefore	effect	increase	effects
interactions	therapy	reported	increased	effect
agents	recommended	concomitant	concentration	drug
oral	administered	administration	decreased	plasma
vitro	caution	use	metabolism	concentrations
interact	dose	drugs	levels	administration
inhibitors	concomitant	agents	auc	interaction
pharmacokinetic	concomitantly	increased	administration	dose
effect	receiving	oral	clearance	levels

Table 6

Most frequent words for any number of drug pairs

Int.	Adv.	Eff.	Mech.	None.
interaction	dose	effects	plasma	drugs
drugs	administered	effect	increase	drug
interactions	caution	drugs	levels	effect
interact	used	agents	mg	effects
drug	drugs	reported	increased	studies
clinical	therapy	concomitant	concentrations	interactions
agents	recommended	administration	auc	dose
inhibitors	concomitant	use	administration	interaction
studies	concomitantly	increase	decrease	clinical
possible	therefore	inhibitors	clearance	plasma

These distributions are what was used to make the decisions for how to classify each of the sentences in the algorithms in the next section. There was no exact science to what words from each of these top ten, was used for classification, but care was taken to choose words that were unique to each class to reduce the possibility of classifying sentences incorrectly. The one big problem with using the distributions was the lack of data for interaction DDI, and the most frequent words for the 'none' category overlapped with many of the other categories, especially with effect DDI. There were not a lot of sentences to go off of for interaction, so the most frequent words were a bit unreliable for that set.

E. Developing the Algorithms

The problem faced with developing the algorithms for extracting the DDIs from sentences is the number of relationships that can exist between drugs in a sentence. If there are two drugs there is obviously just one potential relationship between them, but if there are ten drugs there could be as many as 45 DDI. Most of those relationships are likely to be false, but with the method of just looking for particular words in a sentence, there was no way of telling the exact relationship between drugs unless there are only two drugs in the sentence. That is what forced the decision to create two different types of algorithms for extracting DDI. The two different algorithms were predict_ddi, and detect_ddi. Table 7 shows the words that were picked for classification. The number of words chosen for each type is unequal due to certain classes lacking in unique words for their category.

Table 7

Words Chosen for Algorithm Classification

Interaction	Advise	Mechanism	Effect
interact	caution	increase	effect
interacts	therefore	increased	effects
interaction	recommended	decrease	reported
inhibitors	used	decreased	use
X	dose	plasma	receiving
X	X	levels	X
X	X	concentrations	X

Predict_ddi is meant to make predictions about the relationships between drugs in sentences with only two drugs present. It has the simplest and most understandable output, returning a triple containing (drug0, ddi, drug1) for the sentence that was input. The algorithm goes through and obtains the tokens in the sentence and does NER to obtain the two drugs that are contained in the sentence. The algorithm then looks for words in the sentence that match any of the words from table 7. If it matches any of those words, then it classifies the DDI as that type it matched with. If it does not find any of the words in the table above, then it outputs that there is no relationship between the drugs in the sentence.

There are some drawbacks for the implementation of this particular algorithm. It goes through each word in the sentence from start to finish, and once it finds a word it stops looking and makes its classification. This may not cause problems sometimes, but it can occasionally result in a misclassification. For example, if a sentence has an effect DDI, but also contains the word 'caution' before any of the words associated with an effect DDI, then it will miss those words and classify as an 'advise'. This highlights the problem of common words that overlap between the different types of DDI

Detect_ddi works a little differently than predict_ddi. To try and deal with the problem of many potential relationships in a sentence with many drugs, this algorithm will try and detect which types of DDI are present in the sentence, without making an explicit choice about which drugs interact with one another. It takes in a sentence and similarly to predict_ddi, it tokenizes the sentence and performs NER to extract the drugs. It then goes word by word to see if any of the words from the table match any of the words from the sentence. If it finds a match, it adds that DDI type to the list and moves onto the next word. If it does not match any words, then it returns that there are no DDIs present. In general, this algorithm returns the list of DDIs it detected. If this list is empty, then it returns that there was no DDI detected in the sentence. The benefit of this algorithm is that it checks every word in the sentence as opposed to the predict_ddi algorithm.

So if there is a particular sentence that contains three different DDI: advise, effect, and mechanism, then the algorithm outputs that it detected those three types of DDI. Unlike predict_ddi, it does not try to find the particular drugs that are interacting, just that there are those interactions in the sentence. This is a much easier problem to solve, and attained much better results, as will be explained in the results section, but may not be as useful practically as it would require further inspection from someone to read the sentence and find those interactions manually. Ideally the system would make explicit predictions for the DDI between each drug in a sentence, but given the time constraints, this was the solutions that was chosen

F. Processing PDFs of Papers

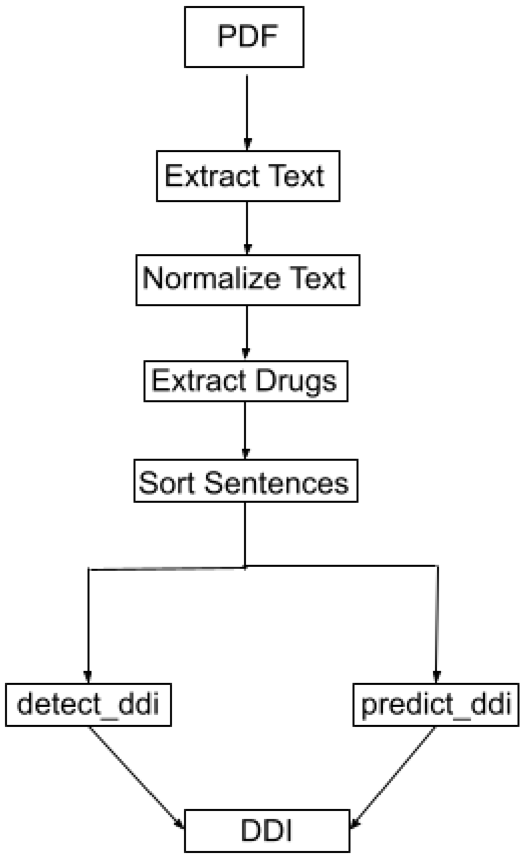
The final task of this project was to pull all these pieces together to create a front to back function that would allow for the user to enter a pdf of their choosing that would be processed and output the DDI predictions made, as well as the DDIs detected in each eligible sentence. This consisted of multiple subtasks combining all of the previous work done that was used to make the algorithms.

The first task was to take the pdf and extract the text from it. The library pdfplumber was used to extract all of the text from each page of the pdf. Other libraries such as PyPDF2 were tried out, but the output was not as reliable as it needed to be from testing different types of pdf files. Some of the sentences and words extracted from the pdf are not extracted exactly as they were in their original state, but for the purposes of this system it worked well enough to find most of the sentences, with some of the problems being a result of the references page at the end of a paper, or how the pdf is structured at the beginning.

After extracting the text, it then had to be tokenized into sentences and then further the words of each sentence had to be extracted. Once that was completed then NER was carried out to find how many drugs were found in each sentence. This step was particularly important, because it is the metric used to determine if there is potentially a DDI present in the sentence. If there are two drugs or more present, then the sentences get sorted out into a list of sentences with a potential DDI, as sentences with one drug or less cannot have an interaction between drugs. The sentences with possible DDI are then further sorted based on how many drugs they contain. If they have two drugs they are then fed into the algorithm for prediction(predict_ddi). If they have more than two drugs, they are then fed into the algorithm for detection(detect_ddi)

Once the sorted sentences are sent into each of their respective algorithms, the output for the PDF entered is returned. It collects all of the output from both of the DDI classification algorithms and lists all of the results. So, for the prediction algorithm the sentences with two drugs are displayed, along with the predicted drug-drug interaction between the two drugs in the sentence. For the detection algorithm, a similar output is displayed, with the sentence in question, and whether or not it detected certain types of drug-drug interactions. Below is a diagram of how this front to back processing of a pdf entered into this processPDF function works.

processPDF Function Diagram



V. Results:

Overall, given the simplicity of the algorithm used to extract the drug-drug interactions, the results were promising. These results are not really comparable to the results obtained by many of the neural techniques referenced in the related works section, since this method is more of a statistically based system. The results based on the detection algorithm may not be as comparable to some other references as it is a different type of task, but the results based on the prediction algorithm are more tangible as they attempt to directly extract DDI.

Below are the results for the prediction algorithm on sentences with pairs of drugs, the detection algorithm on pairs of drugs, and the detection algorithm on any number of drugs found in a sentence. The distinction between the training set and test set used is that the algorithm is based off of the analysis of the training set and knows all the drugs contained in that set. The drugs in the test set were not directly added to the drugs list, whereas the drugs in the training set were. So, in the traditional sense these are not exactly training, and test sets, but they held a similar purpose. The test set sentences are used to signify new sentences that could be from and given paper.

Table 8

Prediction Results for Drug Pair Sentences

	Int	Adv	Mech	Eff	None	Overall
Train	0.50	0.37	0.38	0.26	0.31	0.32
Test	0.60	0.35	0.38	0.24	0.26	0.28

Table 9

Detection Results for Drug Pair Sentences

	Int	Adv	Mech	Eff	None	Overall
Train	0.56	0.65	0.97	0.54	0.48	0.57
Test	0.80	0.65	0.88	0.50	0.63	0.63

Table 10

Detection Results for all Sentences

	Int	Adv	Mech	Eff	None	Overall
Train	0.64	0.50	0.70	0.53	0.77	0.71
Test	0.68	0.52	0.64	0.47	0.82	0.75

For evaluating the prediction algorithm, it had an accuracy of around 30%, which compared to many other papers using statistical or rule-based systems are comparable. The performance was slightly worse for the test set compared to the training set. A big reason for the lower accuracy overall is that the algorithm has to be perfect in finding the drugs and the exact DDI, whereas the detection algorithm has a bit more wiggle room to get some things wrong, but still detect the right things along the way. Another issue that caused the low accuracy was the lack of data. There are obviously fewer sentences with exactly two drugs than with any number of drugs.

The detection algorithm on the drug pair sentences was able to achieve around 60% as can be seen in table 9. Somewhat surprisingly the algorithm performed better on the test set than it did on the training set. This is a promising outcome as it shows that the algorithm can generalize to new types of sentences that it has not seen before. Also surprising was the very high accuracy for detecting mechanism DDI in both the training and test set. One reason could be that the words associated with mechanism are mutually exclusive to some of the words associated with other DDI. There were also a lot of sentences with mechanism DDI to learn from.

The best results came from the detection algorithm being able to achieve 75% accuracy on the testing set of all sentences in table 10. The accuracy of detecting negative instances, listed as 'none' in the table 10 was high for this set. This is particularly important for this problem since in general most sentences will not have a drug-drug interaction in them. So accurately classifying and excluding these negative entities is important. Overall, the worst performing DDI classification was for effect DDI across all metrics used. This is something for which a solution was not found, because many of the most common words for effect DDI were also the most common words for sentences with no DDI at all, so a lot of those effect DDI were swallowed up by the "none" category.

VI. Reflections/Challenges:

I had previously worked on a project for another course about how artificial intelligence is utilized in healthcare, and more specifically, how it is utilized in the field of mental health. I have multiple close family members that work in healthcare, so seeing how the field of AI could be applied to help make things easier seemed like a good road to go down. To me at least, it seems to be one of the more tangible ways that software can be used to help make the world a better place and push the world in a better direction. Seeing the direct effect that the COVID-19 pandemic has had has made this even more clear. This led to looking into these DDI and trying to see what type of contribution could be made by researching this topic.

As with any big project or paper, the methods and approaches that were used evolved over time. There were a few decisions that had to be made throughout the process of not only the actual programming, but also the decisions made about the structure used. These may or may not positively affect the performance of the system. The answer to these challenges is not obvious so there may have been alternative choices that could have been made to improve the performance.

The Drugs List

The method used for building the drugs list does not scale well to the future. It would need to be updated constantly as it does not know how to handle unknown drugs and will just miss them. For the time allotted for attacking this problem, it was a quicker solution so that further work could be done with extracting DDI. If future work was to be done with this method, it would be beneficial to use a more sophisticated method to do NER and extract drugs, that would be able to better work with new drugs or any drugs not currently in the dataset. This would eliminate the need for a drug list or would relegate it to being used as a backup solution.

There were a few challenges that popped up with the drugs list, specifically, what words to include. Obviously, any common name, or brand name of drug needs to be added. Classes of drugs also made sense to include, as it makes sense in certain contexts to refer to the class of a drug rather than the specific drug itself. For example, the word 'antibiotics' is included in the drugs list, and in certain contexts this could be used to describe a drug-drug interaction in a sentence, and in others it is not a useful word to extract. The word 'drug' is excluded from the list, as it could be used to describe a DDI, but more often than not it is not used in that manner, so including it would negatively impact the DDI extraction method.

DDI Pairs

The combinations of relations between drugs in a sentence can be very complicated. As described earlier, a sentence containing ten drug names could have up to 45 relationships. For the method chosen to attempt to solve the DDI extraction problem, looking for certain words in a sentence, there was simply no way that one could find all those specific relationships between every possible pair of drugs. Looking for particular words in a sentence is too simplistic to find that particular complexity. That is where the idea of having both a prediction algorithm for a pair of drugs, and a detection algorithm for everything else came from. This is not an ideal solution as it leads to different types of output depending on the sentence, and the output for the detection algorithm, requires further human interaction to read the sentence and pull out the relationships between the specific drugs themselves. If more time was available, a possible way to help solve this problem could be to focus more on the sentence structure, and extraction relations based off of those criteria.

DDI Corpus Dataset

The imbalance that this dataset suffers from is a result of what most papers that are on DDI consist of. Most sentences will not have a DDI in them, so most sentences will be negative instances and need to be excluded from the data. The DDI Corpus is around 80% negative instances and only around 20% positive ones. Of the positive instances, the number from each category is not balanced either. The information available in this dataset on the interaction DDI is very low, as there are very few sentences available, which most definitely hurt the accuracy of identifying these sentences. This makes sense since interaction DDI is somewhat of a leftover category that does not fit any of the others, but also does not have any unique definition with less than one hundred sentences to work with, the results of classifying these sentences are likely not accurate. A possible solution could be to try and classify these sentences differently to better fit the other three categories, but the ambiguity of natural language is what makes this a difficult problem

Other Attempted Methods

Another route that was explored for this project was to look at the most frequent pairs of words, and triples of words. The hope was that this would give more unique distinctions between the types of DDI. The issue that was run into was not having enough sentences to look at. For single words, the number of sentences was just about enough, but more data would always be better. For the pairs, and especially the triples, there was not nearly enough data to go off of. For the most frequent words, they hovered around in the 100s for appearance, but for the triples and doubles, the most frequent groups were only in the teens. If there were a lot more data to work with then this may be a better method, and one could even look at groups of words that are found between the drugs in the sentence, but for the data on hand this was not a viable option.

Narrowing the Focus

The goal of this project was originally to try and extract more general information from papers/articles and present that in a more digestible fashion. Some of the papers discussed did just that, but a quick realization was made that this is a much bigger problem than it first appeared to be. That is why it was narrowed down to focus on something more concrete, and landed on looking for drug-drug interactions, and extracting that information. This proved to be a good decision as it was a problem that was a bit more achievable over the course of a semester, but there is more work to be done in extracting general information from papers. This general information might include the drug-drug interactions as well as the general results from the paper. So potentially the work done for this project, or the methodology behind it could be used in a larger system in the future.

VII. Conclusion:

Medical research is a very fast-moving field, and in the era of the COVID-19 pandemic, this research has accelerated to a large degree. A large number of papers are published every day on new drugs and medications, and how those medications react with one another is an important problem in the field of medicine. Having some system to sort through all of this new information, and to find these drug-drug interactions could make the lives of medical professionals much easier. This paper sought to use simple statistical methods to try and extract these drugs and relationships.

It utilized a handmade list of drugs to find all of these named entities being the drug names and used the frequency of words in sentences related to each category of DDI to create the algorithms for classifying these relationships. For the simplicity of the system the results were promising, but there were many areas in which it could be improved. From the performance of the two algorithms, the accuracy of prediction may be a little too low for the time being to be useful. For detection, the accuracy was quite a bit higher, and thus, could have some utility for finding drug-drug interactions with a bit more time and work. The DDI problem is one of many that natural language processing can help to solve. With the amount of data growing exponentially in almost every aspect of life, hopefully more work can be done to utilize NLP to try and solve the information-based problems in medicine.

**Code for this paper can be found on github at: Jkillian29/CS498CapstoneDDI: Capstone Project on Extracting Drug-Drug Interactions from Medical Research (github.com)

REFERENCES:

[1] Andriamanalimanana, B 2020, lecture slides, Natural Language Processing CS 518, SUNY Polytechnic Institute, delivered Spring 2020

[2] JD. Jurafsky and J. H. Martin, *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. India: Dorling Kindersley Pvt, Ltd, 2014.

[3] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python*. Beijing Etc.: O'reilly, 2009.

[4] Wikipedia Contributors. (2018, December 3). *Turing test*. Wikipedia; Wikimedia Foundation. https://en.wikipedia.org/wiki/Turing_test

[5] Vaswani, Ashish & Shazeer, Noam & Parmar, Niki & Uszkoreit, Jakob & Jones, Llion & Gomez, Aidan & Kaiser, Lukasz & Polosukhin, Illia, "Attention is all you need" , 2017.

[6] Lim S, Lee K, Kang J (2018) Drug drug interaction extraction from the literature using a recursive neural network. PLoS ONE 13(1): e0190926. https://doi.org/10.1371/journal.pone.0190926

[7] Papadopoulos D, Papadakis N, Litke A. A Methodology for Open Information Extraction and Representation from Large Scientific Corpora: The CORD-19 Data Exploration Use Case. *Applied Sciences*. 2020; 10(16):5630. https://doi.org/10.3390/app10165630

[8] Yang J, Liu Y, Qian M, Guan C, Yuan X. Information Extraction from Electronic Medical Records Using Multitask Recurrent Neural Network with Contextual Word Embedding. *Applied Sciences*. 2019; 9(18):3658. https://doi.org/10.3390/app9183658

[9] Perera N, Dehmer M and Emmert-Streib F (2020) Named Entity Recognition and Relation Detection for Biomedical Information Extraction. *Front. Cell Dev. Biol.* 8:673. doi: 10.3389/fcell.2020.00673

[10] Shengyu Liu, Buzhou Tang, Qingcai Chen, Xiaolong Wang, "Drug-Drug Interaction Extraction via Convolutional Neural Networks", *Computational and Mathematical Methods in Medicine*, vol. 2016, Article ID 6918381, 8 pages, 2016. https://doi.org/10.1155/2016/6918381

[11] Yanshan Wang, Liwei Wang, Majid Rastegar-Mojarad, Sungrim Moon, Feichen Shen, Naveed Afzal, Sijia Liu, Yuqun Zeng, Saeed Mehrabi, Sunghwan Sohn, Hongfang Liu, Clinical information extraction applications: A literature review, Journal of Biomedical Informatics, Volume 77, 2018, Pages 34-49, ISSN 1532-0464, https://doi.org/10.1016/j.jbi.2017.11.011. (https://www.sciencedirect.com/science/article/pii/S1532046417302563)

[12] Deyu Zhou, Lei Miao, Yulan He, Position-aware deep multi-task learning for drug–drug interaction extraction, Artificial Intelligence in Medicine, Volume 87, 2018, Pages 1-8, ISSN 0933-3657, https://doi.org/10.1016/j.artmed.2018.03.001. (https://www.sciencedirect.com/science/article/pii/S0933365717306310)

[13] Sunil Kumar Sahu, Ashish Anand, Drug-drug interaction extraction from biomedical texts using long short-term memory network, Journal of Biomedical Informatics, Volume 86, 2018, Pages 15-24, ISSN 1532-0464, https://doi.org/10.1016/j.jbi.2018.08.005. (https://www.sciencedirect.com/science/article/pii/S1532046418301606)

[14] "FastStats - Therapeutic Drug Use," 2019. https://www.cdc.gov/nchs/fastats/drug-use-therapeutic.htm

[15] I. Segura-Bedmar, "isegura/DDICorpus," *GitHub*, Jul. 29, 2021. https://github.com/isegura/DDICorpus (accessed Nov. 14, 2021).

[16] María Herrero-Zazo, Isabel Segura-Bedmar, Paloma Martínez, Thierry Declerck, The DDI corpus: An annotated corpus with pharmacological substances and drug–drug interactions, Journal of Biomedical Informatics, Volume 46, Issue 5, 2013, Pages 914-920, ISSN 1532-0464, https://doi.org/10.1016/j.jbi.2013.07.011. (https://www.sciencedirect.com/science/article/pii/S1532046413001123)

[17] "DrugBank Release Version 5.1.8 | DrugBank Online," *go.drugbank.com*. https://go.drugbank.com/releases/latest#open-data (accessed Nov. 18, 2021).

[18] C. for D. E. and Research, "Drugs@FDA Data Files," *FDA*, Dec. 2021, Accessed: Nov. 19, 2021. [Online]. Available: https://www.fda.gov/drugs/drug-approvals-and-databases/drugsfda-data-files.

[19] Drugs.com, "Drugs.com | Prescription Drug Information, Interactions & Side Effects," *Drugs.com*, 2018. https://www.drugs.com/.

[20] C. for D. E. and Research, "Preventable Adverse Drug Reactions: A Focus on Drug Interactions," *FDA*, Jan. 2020, [Online]. Available: https://www.fda.gov/drugs/drug-interactions-labeling/preventable-adverse-drug-reactions-focus-drug-interactions#ADRs#:~:text=These%20studies%20estimate%20that%206.7%25%20of%20hospitalized%20patients.

[21] A. Khandeparkar and P. Rataboli, "A study of harmful drug–drug interactions due to polypharmacy in hospitalized patients in Goa Medical College," *Perspectives in Clinical Research*, vol. 8, no. 4, p. 180, 2017, doi: 10.4103/picr.picr_132_16.

[22] T. M. Penning, S. Jonnalagadda, P. C. Trippier, and T. L. Rižner, "Aldo-Keto Reductases and Cancer Drug Resistance," *Pharmacological Reviews*, vol. 73, no. 3, pp. 1150–1171, Jul. 2021, doi: 10.1124/pharmrev.120.000122.