Probabilistic Argumentation for Patient Decision Making

Kawsar Noor

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

September 29, 2023

I, Kawsar Noor, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Medical drug reviews are increasingly commonplace on the web and have become an important source of information for patients undergoing medical treatment. Patients will look to these reviews in order to understand the impact the drugs have had on others who have experienced them. In short these reviews can be interpreted as a body of arguments and counterarguments for/against the drug being reviewed. One of the challenges of reading these reviews is drawing out the arguments easily and forming a final opinion; this is due to the number of reviews and the variety of arguments presented.

This thesis explores the use of computational models of argumentation in order to extract structured argumentation data from the reviews and present them to the user. In particular I propose a pipeline that performs argument extraction, argument graph extraction and visualisation.

Acknowledgements

I would like to express my deepest gratitude to my first supervisor, Anthony Hunter, whose advise and guidance has been invaluable throughout my PhD. Admittedly there have been a number of personal obstacles I have encountered during my programme which he has looked upon with patience and understanding. Additionally many of the ideas I had during my research would not have taken their full form had it not been for the conversations and guidance from him. I would also like to thank my second supervisor Astrid Mayer, whose clinical input shaped a lot of my opinions throughout my research.

Next I would like to thanks my family who have been behind me throughout my work. Their unconditional support and encouragement has been invaluable in seeing me through to the end of my work.

Finally I would like to dedicate this thesis to my late sister, who passed away during the earlier part of my PhD. My earliest memories of doing any form of study involves her as she was my first teacher. In part I feel the privilege I have had in doing the PhD has been due to her.

Impact Statement

User generated reviews and feedback for products are an important to prospective users trying to understand the product they are about to buy into. It is not uncommon to see most websites enabling users to provide feedback on their products. These reviews provide readers with arguments for and against the product in question. This feedback is useful to prospective users since a lot of this information may not have been disclosed on even known to the product owner.

Formulating a final opinion on the product/item once having read all the reviews can be challenging since the reader needs to consider all of the arguments/counterarguments they have read before drawing a conclusion. To this end it seems natural that computational models of arguments could be employed to help draw out this structured information and present it to the user in order to better assist them in coming to a conclusion on the product.

In this thesis I present a pipeline which extracts structured argument related information from each review and presents it the the user. Whilst the data we have worked with in this thesis has been exclusively in the medical domain we do not believe that it is restricted only to this domain. The proposals can easily be extended to any type of product review so long as there are ratings. In the case there are no ratings we also explain that so long as one is able to predict the polarity of the review the proposals can be adapted to account for this.

Our proposals in Chapter 5 and 6 also have application for general probabilistic argumentation. There has been a number of works recently looking at learning algorithms for probabilistic argumentation and we believe our approach differs from these in that there is a unsupervised and supervised learning component that com-

plement each other. We believe that what we have proposed can be extended to settings where one wishes to learn a distribution over a set of argument graphs.

More generally we believe that the pipeline proposed here can be used to better structure the contents of reviews and be used to create better interfaces for users to interact with the contents of these reviews. This in turn can help users make better decisions and conversely product creators can also benefit from knowing what users are experiencing with their products.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

For many years now there has been a general interest in increasing the patient's involvement in directing their treatment [1] [2]. The UK National Health Service (NHS) has long been advocating the Shared Decision Making (SDM) approach to clinical care in which patients and clinicians have a joint responsibility in directing the patient's treatment. This agenda was captured in the book 'No Decision About Me Without Me: Making Shared Decision a Reality' [1] which lists numerous benefits for both the patient and health care provider. Of these benefits are improved patient participation which in itself aids care providers in appropriately directing the treatments as per the patient's needs as well as greater confidence from the patients in decisions undertaken by the health care providers [3].

Progress on this front has lead to the development of decision aids [4] [5] [6] and informational guides for various treatments. These resources are designed to inform patients of the expected outcomes of a treatment, success rates and potential side effects. Commonly such information is conveyed using medical statistics and statistical visualizations. Whilst these resources have helped health care providers step towards making SDM a reality, the challenge of navigating, understanding and ultimately reasoning with the information remains for the patients. Patients come from various educational backgrounds and have different levels of health literacy. These factors negatively influence the patient's ability to synthesize decision aids employing medical statistics [7] [3]. This difficulty is often exacerbated by the fact that the outcomes are uncertain and information incomplete.

Furthermore it is often the case that decision aids may not provide adequate answers to all of a patient's queries. As an example a professional athlete suffering from bone cancer would likely be much more concerned about major surgery decisions than an elderly retired patient and as such expect a detailed insight into those aspects of the operation that will likely effect their profession. A challenge for such patients is obtaining appropriate, varied and relatable information that informs them of the consequences of a decision and the likely outcomes. In such cases the first point of reference will likely be the health care providers with whom the patient would meet with on a regular basis. The health care providers in some instances will attempt to redirect the patient towards patient-friendly material or even other medical specialists; all so as to allow the patient room synthesize the information at their own pace.

Whilst this approach may suffice for some patients, undeniably many are increasingly however turning to the internet for medical information [8] [9] [10]. A survey in the US [1] found in 2012 59% of patients had used the internet to search for medical information and furthermore 35% had gone online specifically to better understand their personal condition. Understandably this trend has been viewed cautiously by some medical professionals due to the unaudited nature of the content found on the internet. Indeed there is even evidence to suggest in certain cases repeatedly searching for health related information can adversely effect the patient and lead to anxiety [11]. There is also evidence to suggest that the patient's exposure to medical information online also influences their subsequent interactions with their healthcare providers [12]. Patients are often not sufficiently qualified to interpret and verify much of the advice and information they find on the internet [13] and this in turn can lead to various adverse outcomes such as self-medicating [14], conflict with their physician which may leave them dissatisfied and uncomfortable with their treatment [13].

Whilst this concern is not unwarranted there is evidence to the contrary to suggest that the quality of information is not always poor [15] and in fact can serve

[1] http://www.pewinternet.org/2013/02/12/the-internet-and-health/

the purpose of alerting patients on specific side effects of a drug. Recognising this much work has been done in developing tools to help patients access better quality health related information on the web [16] [17] [18] [19] [20]. Whilst much of this work has focused on guiding patients towards carefully curated and authored medical information on the web there remains the issue of understanding and helping patients process less regulated sources of information such as that found on social media and drug review websites [21]. In these sites patients report symptoms and medical experiences in ways that are different to how a physician would report.

These patient-provided reviews on the internet represent the 'voice of the patient' and capture up to date feedback and sentiment regarding medical drugs and treatments. This data is thus highly valuable to patients interested in learning about the consequences of a particular treatment as well as to healthcare providers and pharmaceuticals interested in what patients are experiencing [22].

1.1 Problem Statement

Whilst there has been concerted effort to direct patients to curated resources such medical decision making tools and guides these efforts have not addressed the often parallel phenomenon of patients interacting with and being influenced by non-curated material. In particular much less attention has been given to helping patients during their interaction with social media and patient provided feedback on the internet. Perhaps the best effort in recent times has been from an Israeli startup called Treato.com that attempted to aggregate all patient-provided drug reviews on the internet with the goal of extracting adverse side effects from the aggregated data. Their goal was to help patients and pharmaceuticals have better visibility of what patients have experienced and reported when taking medications.

Whilst simply knowing side effects of a drug may be an important concern for patients, drug reviews also provide other interesting insights that may be of value to the patient. Consider below a drug review for acne medication review which had a rating of 3 out of 10:

"The physician assured me that it was an established medicine and that

it had been around for decades. After approximately two months on this I was amazed at how clear my skin had become. However my teeth were beginning to develop brown spots on them as if I had eaten chocolate. I followed up with my physician who advised me stop continuing the medication."

In the above review we can see that the drug did its job well but unfortunately the patient experienced an adverse side effect. From reading this we can conclude that from the patient's perspective the side effects outweighed the benefits here since the rating is quite low (3/10). In addition we see fact that the medication was terminated and assume this has also contributed to the low rating. In other words there is an interaction between different parts of the review that inform the patient's, and more importantly potentially the reader's, final assessment of the drug. This type of insight cannot be captured by simply aggregating and reporting side effects. In fact sometimes this relationship can be even more subtle. Consider the review below for Acne medication that received a rating of 6 out of 10:

"Beyaz has helped with my sudden outbreak of adult acne just like my previous birth control. I got off of over a year ago However my period cramps are about the same as if I wasn t on birth control at all No severe side effects but would like to see improvement in this area. Slight nausea when first taking it at night but taking it in the morning with food seemed to eliminate the nausea. Not bad but could be better"

In the above review we can see that the patient did experience nausea however she did manage to remedy the symptom by adjusting when she takes the medication. For a reader this might imply that it is a manageable side-effect and hence less of a concern. In summary simply looking at what medical outcomes (side effects / benefits) a review has reported will not capture the full richness that it has to offer and that there is in fact an additional layer of 'relations' between statements made in the review that may influence the reader's/patient's final opinion. There is therefore a strong case to help patients reason with reviews and feedback generated by other patients.

Condition	No. of Drugs	Reviews	
		Total Reviews	Avg No. of Reviews per Drug
Birth Control	181	38436	212
Depression	115	12164	106
Pain	219	8245	38
Anxiety	81	7812	96
Acne	127	7435	58
Bipolar Disorder	82	5604	68
Insomnia	85	4904	57
Weight Loss	22	4857	220
Obesity	43	4757	110
ADHD	58	4509	77

Table 1.1: Statistic of drug reviews taken from medical information and drug review website Drugs.com for the top 10 most popular conditions

My work focused on modelling drug reviews as a source of arguments and counter arguments to the use/purchase of the drugs. Modelling as arguments ensures that we capture the sense of conflict that exists between statements whilst also having the ability to model the relationships between the arguments.

A second issue is the influence of reading a number of reviews on the conclusion the reader draws on the medicine. Patients investigating a drug have the challenge of parsing large volumes of reviews before they can draw a balanced conclusion. To illustrate this consider the data in Table 1.1 which summarises drug reviews for a set of common conditions scraped from a popular medications/medical information providing website called Drugs.com in [23]. We can see that not only are there numerous drugs for a given condition but for each drug the patient is forced to read and process large numbers of reviews. Whilst, to the best of my knowledge, this problem has not been studied in the context of medical drug reviews we can look to analogous evidence in studies done on consumer behaviour when reading product reviews. The outcome(s) related of reading reviews (e.g. purchasing and opinions) is found to be influenced by the volume of reviews read, the sequence in which they are read, the proportion of number of negative and positive reviews read and even how reviews are presented to the user (where on the internet website etc.) [24]. Thus we can say that in order to draw a balanced conclusion from one's

reading one must ensure that the breadth of arguments, for and against the product, available across the entire set of available reviews need to be considered by the reader.

In the case of consumer product reviews the opinion from their reading is unlikely to influence the healthcare outcomes for the reader. In contrast one can imagine that if the same phenomena were to occur when patients read medical drug reviews it will influence patient health outcomes. For example a patient may forego a particular medication or may self-prescribe medication thus affecting their health. This is the second motivation behind this thesis which is to allow patients to navigate the arguments/counterarguments in a set of reviews to ensure they are able to form a balanced opinion from them.

Whilst there are numerous data sources from which patients may derive medical information online drug reviews have not been investigated extensively within the literature. Online reviews are increasingly commonplace with a growing number of websites now providing a platform to read and share reviews. Testament to this is the the drug-review website Treato.com, which aggregates drug reviews from all across the web, which found 319,612 results for the cancer drug Tamoxifen alone. Patient-provided drug reviews provide commentary and insight into the patient's experience with cancer drugs. These reviews provide an alternative perspective on the drugs in the sense that they are almost miniature case studies for the drug; providing personal insight into the performance and outcomes of using the drug [25] [26]. Another merit of these reviews is that they are relatable to patients. Patients can empathise with and relate better to comments that come from individuals undergoing similar circumstances as themselves; this is in contrast to hard facts expressed statistically in informational guides.

1.1.1 Research Question

The primary research question I am thus addressing in this thesis can thus be summarised as follows:

> **Question: 'Is it possible to assist patients in synthesising the information presented in online medical drug reviews?'.**

To recap the motivation behind this question let us recall what patients need to do in order to develop/form an opinion on a drug. Patient's need to read over numerous reviews in order to draw an informed conclusion on the treatment they are investigating. Patient's, like other review readers, will be presented with the challenge of navigating the volume of reviews. In addition their conclusion will be biased based on how many reviews they get round to reading, what mixture of positive/negative reviews they read and in what sequence they read the reviews. What is therefore required is some method for aggregating and presenting the reviews in such a way that patients can better synthesise the contents of the reviews. This top-level problem can further be broken down into the following constituent problems:

Constituent Question 1: 'How can we extract information from the reviews to present to the reader?

Whilst algorithms/techniques exist to extract symptoms from patient reviews they do not address the extraction of other relevant statements/arguments from the reviews that also concern the patient, e.g. medical benefits, advice on how to manage medication etc.

Constituent Question 2: 'How can we structure this information to allow the information to be easily synthesised by the reader.?

Reviews are dialectic in nature and consequently, as discussed, there are often further relationships between statements in the single review that also need to be extracted. This additional 'structured' data needs thus to be extracted and identified for the user to interact with and sythnesise.

Lastly there is also the concern of how the extracted structured information is rendered to the patient.

In addressing the aforementioned problems I propose a pipeline of tasks as depicted in Figure 1.1. The pipeline breaks the task of helping patients retrieve and reason with drug reviews into four smaller tasks. The first is argument extraction which I define as the process of retrieving reviews and finding arguments within them. This is processing stage is necessary to identify arguments and counterarguments. The next stage of the pipeline is argument reasoning which allows us to

reason with the collected arguments. Various problems exist in reasoning with multiple arguments such as weighing the value of arguments if they are mentioned by multiple reviewers, or how arguments form coalitions etc. The third stage of the pipeline is to navigate the set of reviews according to the preferences of the user. This is important as it allows the user to filter his/her search and focus on relevant arguments. The last stage of the pipeline is visualising the the arguments. In the following I elaborate on each stage.

1.2 Argumentation Pipeline

1.2.1 Extracting Arguments

I start with extracting arguments from reviews so that they can be processed and evaluated by patients. This involves a degree of argument mining and information extraction. This is done first in Chapter 3 where I introduce a set of simple classification rules which facilitate the extraction of arguments from a number of medical drug review websites. I demonstrate that through the application of these rules I are able to represent reviews as a bag-of-arguments. Furthermore I propose using the numerical rating, which is often a feature of these websites, as an indicator of how the arguments in review can be topologically arranged within an abstract argument graph. I extend this work in chapter 4 by leveraging machine learning to produce a more scale-able method for argument extraction.

1.2.2 Reasoning with Arguments

In Chapter 5 I explore the notion of representing reviews as a bag of arguments. However the topology of the argument graphs in each case is still unknown. The current literature offers probabilistic argumentation as way to handle uncertainty over arguments and in our case, uncertainty over the topology of an argument graph.

Many of drug reviews websites allow reviewers to provide a numeric rating. This rating, I believe can serve as an indication as to which of the arguments presented in the review are the ‘winning’/‘losing’ arguments. On this basis I propose a sequence of tasks which allow us to convert each review into a probability distribution over a set of possible graph topologies (subgraphs). The motivation for

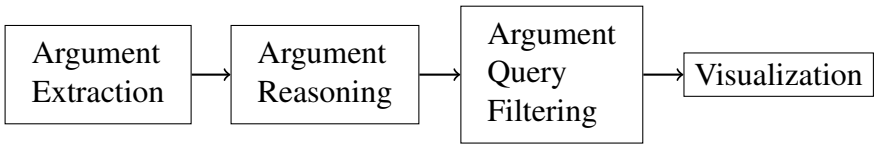


Figure 1.1: Pipeline of tasks

this is to merge the distributions of multiple reviews into an aggregate distribution which can then be used to measure dissimilarity between reviews. I go further by also proposing methods by which reviews which do not have accompanying ratings can use the aggregate distribution in order to identify the most probable topologies given their mixture of arguments. I provide some experimental results and compare my approach to other common machine learning approaches.

1.2.3 Interacting with Arguments

Once the previous steps of the pipeline have been executed the review the system would have extracted arguments and argument relations for that review. This structured data needs then to be stored and retrieved by the user (patient). To this end in Chapter 5 I propose an algorithm for retrieving arguments and relations for a review (based on the previous two steps in the pipeline) as well as a data model for storing the mined data in a structured database. I also note that since most users are not capable of constructing a query in a structured query language (SQL) I explore the use of text2SQL models to help patients convert free text queries such as 'what are the arguments for drug xyz?' into corresponding SQL.

1.2.4 Visualising Arguments

Whilst a large portion of mywork focuses on identification of and reasoning with arguments I dedicate Chapter 6 to helping patients visualize the argumentation process. I do this via a Python implemented visualization tool. The prototype is a tool which allows patients to browse through patient reviews on specific drugs. Whilst browsing the reviews sentences are highlighted as being of a specific argument type and furthermore the sentences are highlighted as attacking each other.

1.3 Published Works

Chapters 3,5,6 contain published works at [27], [28] and [29] respectively.

Chapter 2

Background

In this chapter I review background literature that is relevant to our pipeline. This chapter is intended to cover the topics at a high level and I leave other relevant literature to the individual chapters where the work is specific enough.

2.1 Abstract Argumentation

Abstract argumentation is reviewed as it provides us with an intuitive framework for modelling reviews as argument frameworks/graphs. Whilst alternatives, such as structured argumentation exist, our justification for using abstract argumentation is that drug reviews do not typically present arguments arranged in a premise and claim format, rather the reviews are more akin to a collection of complaints and/or praises regarding the drug. Abstract argumentation models arguments as atomic entities and in our case affords us the ability to abstract away from the details of each complaint/praise whilst providing us with an argument framework and all of the associated extensions.

In the following I present an overview of abstract argumentation. Abstract argumentation is founded on argumentation theory which is the interdisciplinary study of how conclusions can be drawn by using logical reasoning to evaluate claims and their associated premises. Argumentation has been widely studied for its use in knowledge representation and reasoning from contradictory information [30]. Within computational argumentation two broad classes of approaches have been proposed; structured and abstract argumentation. Structured argumentation is con-

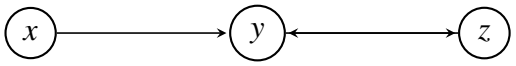


Figure 2.1

cerned with the construction of individual arguments and defining and determining the various relationships between them. Abstract argumentation on the other hand 'abstracts' away from the contents of the individual arguments and models arguments and their attacks as atomic entities. It enables one to abstract away from the details of the arguments, i.e. premises and warrants, and focus on individual arguments and their relationships with other arguments, known as attacks. Whilst many frameworks for abstract argumentation have been proposed one of the earliest, and arguably most popular, is the one proposed by Dung [31]. In it Dung proposes an argument framework as a set of arguments A and the attacks between then R.

Definition 1. An abstract argument graph is a pair (A,R) of arguments A and attack relations $R \subseteq A \times A$.

Example 1. Consider two individuals deciding on whether to eat out or not. Person 1 offers Argument x: 'We should eat at the Italian restaurant tonight'. Person 2 counters with Argument y: 'We should eat at the Spanish restaurant'. Lastly person 1 offers a rebuttal z: 'The Italian restaurant is closed for refurbishment'. Figure 2.1 depicts the arguments in a directed graph $(\{x,y,z\},\{(x,y),(y,z),(z,y)\})$. It illustrates argument x attacking y, y attacking z and z attacking y.

By analysing the attacks between arguments one can begin to determine coalitions of arguments that withstand conflict together, referred to as extensions. In order to analyse such extensions I start by defining a conflict-free set of arguments as those where there exists no attack between any two members of the set.

Definition 2. Given an argument framework (A,R) a set $S \subseteq A$ is conflict-free if for each $a,b \in S, (a,b) \notin R$.

Example 2. Take Figure 2.1. The conflict free sets that exist include $\{x,z\}$ and $\{y\}$.

In order for a coalition of arguments to survive conflict together they are required to defend their members

Definition 3. Given an argument framework (A,R) an argument $S \in A$ is defended w.r.t $S \subseteq A$ if for each $b \in A$ with $(b,a) \in R$ there exists an argument $c \in S$ s.t. $(c,b) \in R$.

This leads to the definition of an admissible set which is a conflict free set where each member of the set is defended by the set.

Definition 4. Given an argument framework (A,R) and a set $S \subseteq A$, S is admissible if:

- S is conflict free
- Each $a \in S$ is defended by S

Example 3. Going back to Example 2.1, we see that set $\{x,z\}$ is an admissible set. We see an attack (y,z) which is defended countered by the attack (x,y).

An admissible set then is one which has the ability to defend itself. The intuition is that it is a collection of arguments that if attacked always offer a counter argument. Using this definition Dung provides the following extensions (classes of admissible sets):

Definition 5. Given an argument framework (A,R) a set $S \subseteq A$ is a:

- Complete Extension- if it is an admissible set and every acceptable argument w.r.t S belongs to S.
- Stable Extension - If it is a conflict free set that attacks all arguments $A \setminus S$
- Preferred Extension - If it is a maximal (w.r.t set inclusion) admissible set
- Grounded Extension - If it is a minimal (w.r.t set inclusion) admissible set

Example 4. If we look at Figure 2.1 the conflict free sets are $\{\},\{x\},\{y\},\{z\},\{x,z\}$. The admissible sets are $\{\},\{x\},\{z\},\{x,z\}$. The set $\{x,z\}$ is a complete, stable, preferred and grounded extension.

It is worth mentioning that analysis of argument graphs in terms of extensions is one method for determining acceptable arguments or sets of arguments. Another popular alternative is via the 'labelling' approach. In the labelling approach one assigns 'labels' to arguments denoting the acceptability status of the argument. In [32] for example $\mathsf{in}, \mathsf{out}, \mathsf{un}$ are some labels that can be assigned to an argument. An *in* label for example indicates that the argument is accepted, *out* indicates it is rejected and *un* indicates an undecided status.

Definition 6. Given an argument graph $G = (A, R)$ a $\{in, out, un\}$ labelling is a total function $L : A \to \{in, out, un\}$.

Example 5. If we consider Figure 2.1 we might assign the label in to argument x since no other argument is attacking it. This would imply y is out since it is attacked and z is in since its only attacker is labelled out.

Dung's initial proposal however may not be suitable for all contexts and in certain cases the framework will be insufficient in modelling more complex relations between arguments [33]. For example the proposal does not allow for the modelling of argument strength nor attack strength etc. Furthermore treating the acceptability of an argument as a binary 'accepted/rejected' status may be too strong in some cases. To address these shortcomings various extensions to Dung's original proposal have thus been made.

Preference-base argumentation [34] [35] [36] for example allows the inclusion of preferences on the order of arguments. This makes it possible to utilise these preferences when selecting winning arguments such that more preferable arguments can defeat less preferable ones. Value-based argumentation on the other hand considers arguments as promoting certain values; the preferences subsequently exists between these values.

In certain cases however it may be desirable to represent argument preference numerically enabling one to quantify argument strength and describe relations in terms of degrees as oppose to fixed categories. To handle this, a separate family of semantics, called ranking-based semantics, have been studied [37] [38] [39] [40].

Ranking semantics generally allow the assignment of numbers to attack relations and arguments. These numbers quantify the degree of strength of argument/attack relative to other arguments in the framework. This notion gives rise to a separate class of acceptability criteria that is centred on analysis of these ranks.

2.1.1 Other Argumentation Frameworks

Besides abstract argumentation there are other argumentation frameworks available. Many of these are structured argumentation frameworks that are frameworks that focus on the construction of the individual arguments as oppose to abstracting the detail to an atomic level. In structured argumentation an argument can be decomposed into a tuple consisting of the argument's claims and premises; other information may also be recorded such as the warrant connecting the claim to the premise. Attacks are binary relations over the arguments denoting an attack between two arguments. In structured argumentation there is often a formal definition to determine whether an attack holds or not.

Notable frameworks for structured argumentation include ABA, ASPIC and DeLP. ABA [41] models each argument as a collection of claims and associated assumptions and inference rules to prove that the assumptions warrant the claim. Attacks in ABA are defined using the notion of 'contrary of assumption' and so if a claim contradicts the assumption of another argument this is considered an attack. ASPIC+ [42] acknowledges that inference rules are not always strict and facilitates weak forms of inference. This further facilitates three types of attacks in ASPIC+; namely an attack based on the uncertainty of an argument's premises, its defeasible inferences and lastly by either attacking the conclusion or the inference. DeLP (defeasible logic programming) [43] is an approach based on logic programming. In DeLP a premise supporting a claim is denoted as a literal. This premise is warranted if all arguments against it are defeated. Attacks can be directed at claims, premises and internal points of a claim. Such analysis leads to a dialectical tree whose root nodes are original argument and whose connected nodes are marked defeated and undefeated.

Whilst structured argumentation enables one to undertake a finer grained anal-

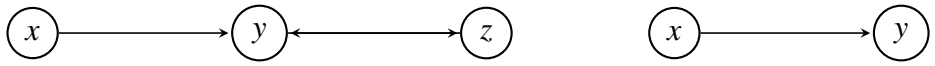
ysis of arguments by breaking them into constituent premise/claim structures this is not wholly needed for this use case where most arguments can not be decomposed into such components. In the case of drug reviews, comments are often for or against the drug and hence more akin to enthymemes; in this case abstract argumentation offers a sufficient framework to accommodate this information.

2.2 Probabilistic Argumentation

Probabilistic argumentation combines argumentation theory with probability theory. It enables a quantification of the various types of uncertainty that exist in argumentation using probabilities. It is well suited for modelling real world scenarios where uncertainty regarding the acceptability of arguments and/or attacks occurs in degrees and is best expressed numerically. Probabilistic argumentation provides methods for handling and reasoning with such uncertainty by allowing for this uncertainty to be expressed in terms of probabilities.

Broadly speaking proposals for probabilistic argumentation can be thought of in terms of two major categories of approaches; these are the **constellations approach** and the **epistemic approach**. In the constellations approach there is uncertainty in the topology of the graph (i.e whether a particular attack or argument exists in a graph). The constellations approach therefore will typically be useful when we wish to identify a probability distribution over the set of various possible topologies for the argument graph. The epistemic approach on the other hand assumes that the topology of the graph is known but the belief in an argument or attack is uncertain and better expressed in probabilities.

The constellation approach was first proposed in [44] where the authors introduce the notion of probability that an argument or attack appears in an argument graph. Through these probabilities it is possible to identify a probability distribution over the subgraphs of an argument graph. The approach enables one to determine the probability that a set of arguments exists within an extension but uses the assumption that the likelihood of arguments appearing is independent of the presence of other arguments. This assumption was later relaxed in [45] since there certain sit-



(a) A full subgraph **(b)** A spanning subgraph

Figure 2.2: Example of subgraphs given arguments $\{x, y, z\}$

uations in which one cannot assume independence between arguments. In [45] the proposal was to have the probability distribution only across to only the spanning subgraphs of the argument graph.

Definition 7. Let $G = (A, R)$ be an argument graph. A subgraph is a graph $G' = (A', R')$ where $A' \subseteq A$ and $R' \subseteq R$. The subgraph G' is called **full** if $A' \subseteq A$ and $R' = (A' \times A') \cap R$. The subgraph $G' = (A', R')$ is called **spanning** if $A' = A$ and $R' \subseteq R$.

Example 6. Given an argument graph $G = (A, R)$ where $A = \{x, y, z\}$ and $R = (A \times A)$ Figure 2.2 depicts an example of a spanning subgraph and a full subgraph. The spanning subgraph must contain all of nodes of the original graph in contrast to a full subgraph which can contain fewer.

Definition 8. Given a argument graph $G = (A, R)$ a subgraph distribution is a function $P^c : Sub(F) \rightarrow [0, 1]$ where $\sum_{F' \in Sub(F)} P^c(F') = 1$.

Example 7. To illustrate the approach Table 2.1 which shows the a probability distribution the subgraphs of the argument graph $(\{x, y\}, \{(x, y)\})$.

The constellations approach is useful for understanding probabilities associated with different coalitions of arguments. If we look at Table 2.1 we can see that the combined probability from all graphs, $G_1, .., G_5$, for arguments $\{x, y\}$ being in the grounded extension, which I denote as $P_{Gr}(\{x, y\})$, is equal to 0.45; given by graph G_2. Other coalitions are $P_{Gr}(\{x\}) = 0.97$, $P_{Gr}(\{y\}) = 0.49$ and $P_{Gr}(\emptyset) = 0.01$.

Probabilistic argumentation embeds the notion of probability with arguments graphs. In so doing it affords us the ability to explore specific learning problems within argumentation. In my case I am interested in learning argument graph representations based on the arguments presented by reviews on the internet. This learning problem is discussed the next section.

No	Graph	Probability
G_1	$x \rightarrow y$	0.5
G_2	$x \quad y$	0.45
G_3	x	0.02
G_4	y	0.04
G_5	$\emptyset$	0.01

Table 2.1: Probability distribution over subgraphs

2.3 Probabilistic Argumentation for Learning Abstract Argument Graph Structures

Probabilistic argumentation as a means for learning abstract argument graph structures has previously been proposed. In the following I consider some notable examples and assess their suitability for our problem.

[46] proposes an on-the-fly algorithm which uses an input feed of labellings of arguments, where each argument is labeled using $\{in, out, un, off\}$ acceptability semantics, in order to output a weighted argument graph.The resulting weighted graph is meant as a meaningful representation of the input labellings. During runtime weights are assigned to the attacks between the arguments in accordance to a set of rules, referred to as credit rules, which use input labellings to credit specific attacks once the criteria of the rule has been met. The rules work by considering the labellings of two arguments at a given time.

- Rule 1: If L(a) = in and L(b) = in then:

$$(b,a) \leftarrow (b,a) - 1$$

- Rule 2: If L(a) = in and L(b) = un then:

$$(b,a) \leftarrow (b,a) - 1; \quad (a,b) \leftarrow (a,b) - 1$$

- Rule 3: If L(a) = un and L(b) = un and for any possible attacker c of a where $L(c) \neq in$:

$$(a,b) \leftarrow (a,b) + 1; \quad (b,a) \leftarrow (b,a) + 1$$

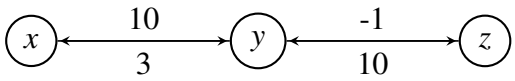


Figure 2.3: A weighted argument graph

- Rule 4: If L(a) = out and L(b) = in and for any possible attacker c of a where $L(c) \neq in$:

$$(a,b) \leftarrow (a,b) - 1; \quad (b,a) \leftarrow (b,a) + 1$$

A merit of the algorithm's design is that it can be interrupted at any time and will return a weighted graph. A further merit comes in the use of the *off* label, which allows arguments to be marked as unknown or not recognized. This is translates well into the real world where many agents putting forward their labels for arguments will not be aware of all available arguments and yet still be able to combine their labellings with other agents into a single graph representation.

Example 8. Consider an argument graph with two arguments a and b and the labellings shown in Table 2.2. A labelling is a tuple $(A_{in}, A_{out}, A_{un}, A_{off})$ where A_{in} corresponds to the set of arguments labeled in, A_{out} to those labeled as out and so on. The first labelling $(\{a,b\}, \emptyset, \emptyset, \emptyset)$ would yield a weighted graph $a \rightarrow b$ with credit 0 and $a \leftarrow b$ with credit 0. The second labelling $(\{a\}, \{b\}, \emptyset, \emptyset)$ would use Rule 4 and update the graph attacks $a \rightarrow b$ to weight 1 and $a \leftarrow b$ to weight -1. Finally the last labelling $(\{b\}, \{a\}, \emptyset, \emptyset)$ would update $a \rightarrow b$ to 0 and $a \leftarrow b$ to 0.

The algorithm is however not suited to our problem on account of the following. A weighted graph does not present itself as readily usable to an agent an agent who proposes arguments and would like to use the weighted graph to determine what the attacks would be (query agent). For example take the weighted graph in Figure 2.3. If we now assume that an agent who is aware of arguments *x, y* wanted to use this graph to understand what the most likely graph structure relevant to his/her case was, then a starting point would be to look at the subgraph containing arguments *x, y* in isolation from the rest of the graph. However the problem still remains in finding an extension of this graph with the associated weights. In this case do we consider the imbalance in weights as an indication of argument *y*'s superiority over

a	out	in	out
b	out	out	in

Table 2.2: Three labellings of arguments a and b

x? If the weights were not so imbalanced does this indicate that there is a mutual attack? These current issues make approach the unusable as a classifier of query agents. The paper does offer the idea of a threshold value for the weights to be considered acceptable, yet the problem remains as this does not consider the notion of degrees of inclusion and exclusion of attacks.

A second notable paper is [47]. The paper offers a novel Bayesian network model which can estimate probability of attacks for a given set of input arguments labellings. The approach works by producing a series of posterior probability distributions and then connecting the corresponding variables associated with each distribution using a Bayesian network model.

The paper, as stated by the author, is designed for applications where the labellings are an $\{in, out, un\}$ of arguments. Much like the previous paper reviews, this causes incompatibility issues with my dataset in that there is not a single input sequence of labellings.

It is also worth mentioning [48] which proposes the use of Beta distributions to quantify the level of uncertainty of an argument. This assumes the epistemic approach to argumentation where one is attempting to model belief in an argument. By using Beta distributions one is able to determine beliefs of sub populations of agents. It is also possible to understand what influence there is in beliefs in one arguments if another argument is retracted or posited. For this thesis however since I have opted for the constellations approach the proposal is not compatible.

2.4 Other Machine Learning Approaches in Argumentation

There has been recent interest in understanding the use of machine learning approaches to help with learning problems in argumentation. An example is neuro-argumentative systems which is an emerging field of argumentation that seeks to

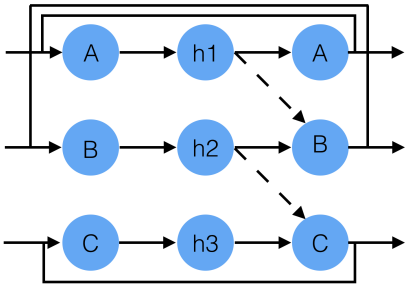


Figure 2.4: Example of a

integrate abstract argument frameworks with neural network architectures [49] [50].

Garcia was the first proponent of the idea of neuro-argumentative systems [50]. He proposed a simple two layer neural network architecture which was able to capture attacks between arguments in an abstract argument framework. He proposed the number of nodes in the input layer should correspond to the number of arguments present in the framework. The same number of nodes should also exist at both the hidden and output layers. The attacks between arguments are to be captured between the hidden layer and the output layer where negative weights indicate an attack and positive weights a support. As an example consider the example in Figure 2.4. The attacks between $A \rightarrow B$ and $B \rightarrow C$ are denoted as dashed lines. The forward pass of the algorithm works by taking input values at time t_0 where input is fed forward through the network to the output layer. Activation at node occurs when the input exceeds a threshold value. Garcia's work proposes an exciting bridge between the rapidly changing landscape of connections models and argumentation.

More recently there was also an investigation into using Boltzmann machines [51], which are often referred to as shallow neural networks, to capture probabilities of attacks given an input sequence of labellings. The approach allows one to relax the assumption of independence between arguments mutually attacking the same argument on account of the hidden nodes.

Whilst the existing literature proposes interesting insight into the potential of this upcoming field I find my use case is not explicitly compatible with these approaches yet. A problem we will see in Chapter 4 is that some reviews produce more

labels and potentially more training data than other reviews and could therefore bias the training data if presented to a neuro-argumentative system.

In summary there are a number of novel approaches seeking extend/incorporate abstract argumentation, e.g. [52], to fit within modern machine learning models of learning.

In this section I have introduced the relevant background literature that is common to most chapters and works presented in this thesis. I chose to review the remaining literature on a chapter by chapter basis.

Chapter 3

Argument Extraction

Evidence-based medicine stipulates that patients are offered medication and treatment based on scientific evidence published in the medical literature. Whilst patients may find it difficult to relate to medical statistics they are keen to understand benefits, potential side effects and implications on their life and life style. Drug reviews, much like other product reviews on the internet, provide useful insights into the performance and acceptance of the drug amongst patients who have experience of it [15]. Drug review websites contrast with traditional medical resources by providing access to an interesting set of arguments based on personal experiences of the patients. Whilst this reflects the subjective experience of individuals we propose to view the review process as users providing arguments and counter arguments about the drug in question.

If such arguments can be retrieved from drug review websites, it is possible to arrange them using existing argument-theoretic frameworks such as Dung's argument graph [31]. The generation of a Dung graph to represent the arguments in a single drug review, enables one to elicit the overall assessment of the drug based on the evaluation of the argument graph; such evaluations can be achieved using Dung's extensions. In order to validate this assessment it is possible to exploit the rating function provided by drug review websites, which enables users to numerically score the drug. I propose that by correlating the rating, produced by the argument extraction and analysis system, against the numerical rating data given by the drug review author I can ascertain a general measure as to how accurate the

analysis was.

The purpose of this chapter is to demonstrate that it is possible to extract arguments from drug reviews and arrange them into argument graphs. It is a proof-of-concept approach to tackling the argument mining and argument reasoning pipe in the pipeline. This experiment was useful in understanding how arguments and argument graphs can be represented in reviews; the lessons from this chapter go on to inform the more substantial work done in the subsequent chapters on argument extraction and reasoning.

I believe this work is a novel contribution because it shows how Dung's approach to analysing arguments is reflected in the way drug review authors evaluate conflicting arguments within a single drug review. This suggests that one could extend the application of this method to those drug review sites that do not have user provided ratings in order to generate analogous ratings. Furthermore the argument-based analysis could provide structured information to patients who are trying to garner an understanding of how the drug was received by previous users. I expect that this tool will provide patients with supplementary reasons for and against the treatment.

Note my method of extracting arguments is not meant as a contribution to argument mining, rather it is a simple method to automate the process of instantiating argument graphs and could potentially be improved by harnessing more advanced argument mining techniques such as those reviewed in [53].

3.1 Argument Extraction

Argument extraction is the process of extracting argument components, primarily premises and conclusions, from text. A motivation for extracting arguments is to provide structured data for use within computational models of arguments or reasoning engines [10]. Applications have been developed for various corpora including legal texts [12], medical articles [8] and user generated content [1]. The most prevalent technique for extracting arguments, first introduced by Teufel [54], is to identify argumentative components within a body of text before establishing rela-

tions between those components; a process named argument zoning. These steps have more recently been recognized as the key steps to building argument mining systems [53]. This approach is useful for a fine grain extraction of all argumentative components within a text but does not provide a context for the arguments within the text. Furthermore often arguments proposed in a corpus will be arguments for or against a common topic. The current argument mining pipeline is not intended to isolate argument types which could be found by a wide study of the corpus whilst our work requires us to aggregate common argument types.

Textual corpora of a specific domain will often present recurrent themes which manifest themselves through the use of phrases and words specific to that domain. Take the example of second-hand car reviews on the internet in which the majority of comments will contain vocabulary pertaining to themes of discussion such as mileage, price and brand. The frequency with which these phrases and words appear give an indication as to the dominant themes within the text. With knowledge of this vocabulary it is therefore possible to mine information for each of the dominant themes. Furthermore, for each unit of information retrieved, its polarity can be assessed, i.e. whether the information was of a negative or positive sentiment. The association of this polarized information with its theme can be seen to exhibit the qualities of a classic argument pair in a (premise, claim) structure. Extending our example of the second-hand car, if we find a sentence speaking positively about the brand, this can be taken as an argument in favour of the car; the premise being the sentence and the claim being the car is a good car. The polarity of the information thus determines whether the pair is to be an argument for or against that theme.

If this process is applied to a group of texts sharing the same domain it is possible to obtain arguments for and against each theme, which in turn enables one to build up an argumentative representation of the texts. This process allows one to glean the overall sentiment of the corpus by substantiating the assessment through the collection and evaluation of arguments. This paper exploits this process by applying it to patient experiences with medical treatments as expressed on the social web.

In the following, I show how simple rule-based information extraction techniques can be harnessed to extract arguments. The implemented system has been written in Python, and makes use of the natural language processing toolkit NLTK. [1]. The code and datasets are available on Github. [2].

I take reviews from two medical websites (Drugs.com and Webmd.com). Drug reviews on these websites, much like other products tend to focus on a core set of features of the product. I identify a set of common features found across the various reviews. The recurrent themes tend to be centred around the side effects experienced, the overall success of the drug and the general experience with the drug. Take for example the following review:

> *"I get achy$_{\textit{side effect}}$ in the hands and feet, have gained weight$_{\textit{side effect}}$ (20)lbs. and hate$_{\textit{negative experience}}$ the hunger it seems to give me cravings for calorie laden foods."*

As can be seen in review above the user's focus is on the side effects of the drug, whilst some words such as 'hate' would indicate that the user had a negative experience with the drug. Similar observations were made when reading a range of different drug reviews. With these observations in mind we identified the following core themes which we use to extract arguments for/against a number of drugs:

- Presence of side effects;
- Severity of the side effects;
- Polarity of experience with the drug;
- Whether or not supplementary drugs can be taken for side effects from the primary drug.

Each theme is identified through the appearance of keywords. Using the example of the theme *presence of side effects*, statements pertaining to this theme are identifiable when a side effect is mentioned; vocabulary for which can be sourced

[1] http://www.nltk.org/
[2] https://github.com/robienoor/NLTKForumScraper

from medical literature. Furthermore each theme can be assessed for polarity, so continuing the example of the *presence of side effects* theme we say that the resulting argument types are *the absence of a side effect* and *the presence of a side effect*. These argument types thus either favour or oppose the use of that particular drug. Using this approach I formalised 10 classification rules based on the themes mentioned above.

I make the assumption that each argument is presented in a single sentence. A sentence may convey multiple arguments but no argument requires multiple sentences to convey it. This is a simplifying assumption that we do not further investigate in this paper. The role of the classification rules is to identify the types of argument present in each sentence.

In order to determine the sentences in a drug review I use the NLTK library which provides a standard sentence tokenisation function that splits text into sentences based on the presence of common delimiters such as full-stops, large line breaks etc. The sentence tokenisation function takes as input a full text (string) and returns a list of strings; hence each sentence is treated as a string. Further to this the sentence is subsequently tokenised into words using the standard work tokenisation function available in NLTK. A sentence is therefore represented as a list of words.

In order to define the classification rules we compiled a number of keyword lists namely `Symptoms`, `Drugs`, `Diseases`, `PosWords`, `NegWords`, `Inverters` and `SideEffects`. The list `SideEffects` contains the term *side effect* in various forms eg: *symptoms*, *side-effects* etc. The list `Inverters` contains a list of negating words eg: *no*, *not*, *none* etc. These lists serve the purpose of providing quick access to medical and sentiment terminology.

The classification rules below are formalised using first-order logic. Below is a list of predicates that are common across the classification rules.

- `Occur`(`sentence`, `wordlist`, `position`) which holds when there is a word in `wordlist` that occurs at the point `position` in `sentence`
- `ImmediatelyBefore`(`string1`, `string2`) which holds when `string1` is the sentence immediately before `string2`.

- `Contains(sentence,wordlist)` which holds when at least one of the words in `wordlist` is in `sentence`.

- `ArgumentType(sentence,type)` which holds when the `sentence` is of type `type`.

I also introduce an auxiliary function that is shared between a number of the classification rules. The function Score checks to see the number of times a particular set of keywords appears in a sentence.

- `Score(sentence,wordlist)` is a function that returns the number of words in `wordlist` that occur in `sentence`

With the common predicates/function defined above I proceed to define all of the individual classification rules. Essentially each rule classifies a sentence to be of a particular type if the conditions of the rule are met for the sentence. A sentence may be classified to be of more than one type (though in practice this is infrequent).

1. **NoSideEffectsI**: This rule looks for an inverter word immediately followed by a side effect string.

   eg: *I have* $\underline{no_{inverter}}$ $\underline{side\ effects_{sideEffect}}$

   $\forall$`sentence, string1, string2`
   `Contains(string1,Inverters)` $\wedge$ `Contains(string2,SideEffects)`
   $\wedge$ `ImmediatelyBefore(string1,string2)`
   $\rightarrow$ `ArgumentType(sentence,noSideEffectsType1)`

2. **NoSideEffectsII**: This looks for an inverter word before a side effect string irrespective of its position in the sentence.

   eg: *During the time I took the medication I did* $\underline{not}_{inverter}$ *experience any* $\underline{side\ effects_{sideEffect}}$ *at all*

$\forall$sentence, position1, position2
Occur(sentence,Inverters,position1)
$\wedge$ Occur(sentence,SideEffects,position2)
$\wedge$ position1 $<$ position2
$\rightarrow$ ArgumentType(sentence,noSideEffectsType2)

3. **SideEffectsI**. This looks for a side effect string with no inverter words in the preceding words.

eg: *The side effects*$_{sideEffect}$ *outweighed the good*

$\forall$sentence, position1
Occur(sentence,SideEffects,position1)
$\wedge \neg\exists$position2(
Occur(sentence,Inverters,position2)
$\wedge$ position1 $>$ position2)
$\rightarrow$ ArgumentType(sentence,sideEffectsPresentType1)

4. **SideEffectsII**. This searches for a symptom within a sentence.

eg: *The side effects were gradual at first but now they are full blown...fatigue*$_{symptom}$ *and joint pain*$_{symptom}$

$\forall$sentence
Contains(sentence,Symptoms)
$\wedge \neg$Contains(sentence,PosWords) $\wedge \neg$Contains(sentence,NegWords)
$\rightarrow$ ArgumentType(sentence,sideEffectsPresentType2)

5. **BearableSideEffects**. If a side effect and positive word are mentioned we interpret this as meaning that the side effect is present but bearable.

eg: *So far my joint pain* $_{symptom}$ *is better*$_{positiveWord}$ *and my energy and motivation had noticeably improved*$_{positiveWord}$

```
∀sentence
Contains(sentence,Symptoms)
∧ Score(sentence,Poswords) > Score(sentence,Negwords)
→ ArgumentType(sentence,bearableSideEffects)
```

6. **UnbearableSideEffects**. If a side effect and a negative word are mentioned we interpret this as meaning that the side effect is present and unbearable.

   eg: *I had several* $\underline{fevers_{symptom}}$ *and* $\underline{bone\ pain_{symptom}}$ *making it very* $\underline{difficult_{negativeWord}}$ *to get up*

```
∀sentence
Contains(sentence,Symptoms)
∧ Score(sentence,Negwords) > Score(sentence,Poswords)
→ ArgumentType(sentence,unbearableSideEffectsType1)
```

7. **UnbearableSideEffectsII**. If a side effect is mentioned in a sentence whose sentiment score is neutral we interpret this as meaning that the side effect is present and unbearable.

   eg: *The constant nightly* $\underline{hot\ flashes_{symptomWord}}$ *and* $\underline{joint\ pain_{symptom}}$ *are* $\underline{irritating_{negativeWord}}$ *but yet I'm still* $\underline{hopeful_{positiveWord}}$

```
∀sentence
Contains(sentence,Symptoms)
∧ Score(sentence,Negwords) = Score(sentence,Poswords)
→ ArgumentType(sentence,unbearableSideEffectsType2)
```

8. **PositiveExperience**. The presence of only positive words is interpreted as meaning a positive experience.

   eg: *I felt much* $\underline{better_{positiveWord}}$ *on it*

$\forall$sentence
$\neg$Contains(sentence,Symptoms)
$\wedge$ Score(sentence,Poswords) > Score(sentence,Negwords)
$\rightarrow$ ArgumentType(sentence,positiveExperience)

9. **NegativeExperience**. The presence of only negative words is interpreted as meaning a negative experience.

   eg: $\underline{Terrible_{negativeWord}}$ $\underline{terrible_{negativeWord}}$ *drug*

$\forall$sentence
$\neg$Contains(sentence,Symptoms)
$\wedge$ Score(sentence,Negwords) > Score(sentence,Poswords)
$\rightarrow$ ArgumentType(sentence,negativeExperience)

10. **SuppDrugAvailable**. A sentence containing a symptom and another drug, which is not the drug being reviewed, is taken to mean that the patient is taking a supplementary drug. The predicate mainDrug(drug) holds when drug, which the drug being reviewed, is not mentioned in the sentence.

    eg: *I have* $\underline{anxiety_{symptom}}$ *added* $\underline{Ativan_{suppplementaryDrug}}$ *to my drugs...*

$\forall$sentence,drug
$\neg$Contains(sentence,Symptoms) $\wedge$ Contains(sentence, Drugs)
$\wedge \neg$mainDrug(drug)
$\rightarrow$ ArgumentType(sentence,supplementaryDrugs)

In this section I have formalised 10 classification rules that are used to extract arguments from medical drug reviews. I show in the next section that the classification rules, albeit simple, yield a reasonable performance. The rules could further

Rule	No. Arguments Extracted	No. T	% T	%F	%NA
PositiveExperience	368	182	49.46	17.12	33.42
NegativeExperience	446	294	65.92	3.81	30.27
NoSideEffectsI	18	18	100	0	0
NoSideEffectsII	31	17	54.84	22.58	22.58
SideEffectsI	142	114	80.28	7.74	11.97
SideEffectsII	61	52	85.25	4.92	9.84
BearableSideEffects	22	11	50	31.82	18.18
UnbearableSideEffectsI	93	81	87.10	4.30	8.60
UnbearableSideEffectsII	320	261	81.56	7.50	10.94
SuppDrugAvailable	180	21	11.67	2.7	85.56

Table 3.1: Accuracy of all arguments pulled out per classification rule

be improved by harnessing argument mining techniques and natural language processing.

3.2 Evaluation of Extracted Arguments

The rules mentioned in the previous section were tested against a set of 570 reviews concerning 4 drugs. The drugs tested include two cancer drugs, an anti-depressant and an over-the-counter painkiller. This variation of drug types is useful since it allows us to measure how well argument types hold across the various drugs. I acknowledge that the quality of arguments from two drug types may be vary. As an example a cancer patient may rate a drug highly on the basis they are alive although the reason for it could be surgery or other treatments. This is a line of enquiry that I intend to explore in future works.

In order to validate the performance of each of these rules, the extracted arguments were manually checked by a single human annotator (myself) to see if they had been classed correctly. Each extracted argument was marked as being either *T* (true - the argument was classified correctly), *F* (false - the argument was classified in the opposite class and could in fact be used as a counter argument) and *NA* (irrelevant - argument extracted has no relation with its intended class).

The results in Table 3.1 demonstrate that using my classification rules, it is possible to extract relevant arguments regarding treatments. The rules exhibited

different precisions (where precision = No. T/No. of Arguments Extracted). For example the rules *NoSideEffectsI* and *PositiveExperience* achieved precisions of 100% and 49.46% respectively. This variability is expected as some of the rules, such as the *PositiveExperience* rule, search for context independent words whereas others search for the occurrence of medical terminology. I also recorded lower precisions when comparing positive sentiment rules to negative ones, eg: *BearableSideEffects* vs. *UnbearableSideEffectsI*. I attributed this to my observation that patients rarely mention a side effect without the intent of complaint.

Alongside these difficulties, I encountered a number of natural language challenges, the majority of which were attributed to the casual nature with which authors wrote their reviews. The difficulties encompassed spelling mistakes, adoption of new terms, abbreviations and general violations of English grammar. Another challenge was the use of non-standard terminology to describe side effects. The quote below highlights this kind of issue.

> *"...my vision seems to be getting weak."*

Discerning a loss of vision from the use of the word *weak* is non-trivial, and is not easily captured using lookup data. Going forward I would seek to improve the classification rules by adopting better natural language processing techniques such as custom named entity recognition for medical records [55] for extracting standard medical terminology.

Lastly another common phenomena was that the when the patient wanted to express a negative sentiment many times they would employ sarcasm to do so. Take the example below:

> *"This drug has done me wonders given that since starting it I have developed acne and redness. I am amazed that this is legal to prescribe..."*

In the example above it is clear that the medication has not worked as expected and the use of the positive language is to express sarcasm. This level of detail is difficult to distinguish using a keyword based approach. Sarcasm thus influences

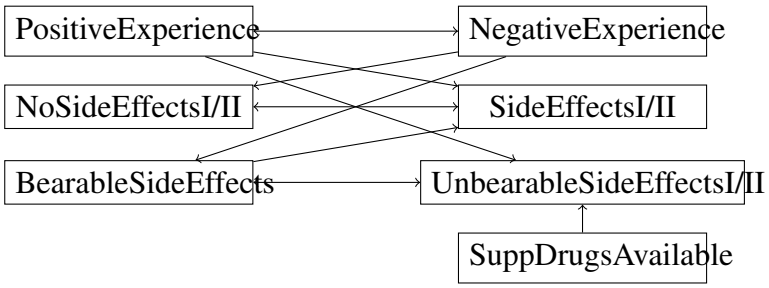


Figure 3.1: Argument graph capturing attack relation between the various classification rules.

I am thankful and consider myself lucky to now be a survivor.$_{\textbf{PositiveExperience}}$

↓

I have experienced major hot flashes day and night.$_{\textbf{SideEffectII}}$

↕

No other side effects at this time.$_{\textbf{NoSideEffectsII}}$

Original Post: *"I have experienced major hot flashes day and night..No other side effects at this time..I am thankful and consider myself lucky to now be a survivor"*
Drug: Tamoxifen
User Rating: 8
Grounded Extension: {*PositiveExperience*, *NoSideEffectsII*}
Argument Evaluation: Positive

Figure 3.2: A review for the drug Tamoxifen. Three arguments were extracted. The grounded extension contains only positive arguments and so the argument-based rating is positive.

the performance for any rule which uses a combination of positive words and/or a symptom, e.g *PostiveExperience*, *UnbearableSideEffects* as reflected in Table 3.1.

3.3 Evaluation of Argument Graphs

In this section I investigate instantiating an argument graph for each drug review with the arguments extracted from it, I then use Dung's grounded semantics to derive a rating for the drug. I validate these argument-based ratings by correlating them with the numerical ratings given by the reviewers at the end of their drug reviews.

In order to instantiate the argument graph for a drug review we require a de-

fined set of attack relations for all argument types. To simplify the graph I have chosen to group together arguments that have the same semantic meaning, i.e. **NegativeExperienceI, NegativeExperienceII** since the underlying rules used are designed to retrieve an argument with the same underlying semantic theme. I refer to this 'final grouping' as argument types throughout the rest of this thesis. In Figure 3.1 I specify these attack relations based on the analysis of a large number of reviews focused on understanding how each argument type influences the numerical ratings provided by the user; more specifically I model the competing levels of influence that the argument types have over the rating with respect to one another. I note that the arguments on the left of the graph are positive (i.e. contributing positively in some way to the rating) and those on the right are negative.

The analysis focused on observing, in the context of a full review, which arguments typically appeared to contribute to the final rating. It was found that in general if a review contained a statement that was highly positive/negative (i.e. those captured by the rules *Positive/Negative Experiences* always had a corresponding rating with a similar polarity. However if a positive argument was mentioned alongside a negative argument then in general positive arguments seemed to win; this was because the positive statement was typically the concluding remark (e.g. 'overall very pleased with drug' etc).

For other relationships it was observed that if a side effect was mentioned and there was mention of it being bearable, it was phrased such that the severity of the side effects were tolerable and hence not contributing negatively to the rating. For this reason I placed an attack between the *BearableSideEffect* and the *SideEffects* arguments. For every other argument in the framework I chose to have them as bidirectionally attacking each other since no other patterns emerged between the respective arguments in the analysis.

A consequence of this choice of attack relation is that the grounded extensions of Figure 3.1 and any of its subgraphs constitute either entirely positive arguments, negative arguments or an empty set. These three possible sets correspond to three polarities (positive, negative and neutral) and serve as our argument-based ratings.

Rating	Negative	Neutral	Positive
1-4	0.531	0.443	0.262
5-7	0.198	0.216	0.172
8-10	0.270	0.340	0.566

Table 3.2: Dung Assesment vs. User Rating using all posts

Rating	Negative	Neutral	Positive
1-4	0.624	0.417	0.129
5-7	0.206	0.202	0.178
8-10	0.170	0.380	0.693

Table 3.3: Dung Assessment vs. User Rating using only validated sentences

In order to validate these ratings I correlate the polarity of a drug review to the numerical value provided by the user. In facilitating this correlation the numerical scale was split into three ranges based on analysis of the reviews which focused on seeing how the overall sentiment of the text corresponded to the rating scale. I assume that a drug review with a rating less than 4 to be a negative rating, a drug review with a rating between 5 and 7 to be neutral and any drug review with a rating greater than 7 to be positive. An example of the system in practice, from argument extraction through to analysis, can be seen in Figure 3.2.

I ran the experiment using two sets of arguments. In the first set, I used all of the arguments extracted using the classification rules. This was to evaluate the performance of the entire automated process, from argument extraction through to analysis of arguments. In the second set of arguments, we utilised only those extracted arguments which have been annotated as being of type *'T'*. By comparing the correlation matrices for both argument sets we are able to measure the effect of inaccuracies in my classification rules on the argument-based ratings.

The results of the experiment in Tables 3.2 and 3.3 indicate a positive correlation in the positive and negative classes. It can be seen that there is a notable improvement in correlation in Table 3.3, given that here we use only validated arguments. The neutral class appears comparatively less correlated with classifications distributed across the ratings scale. What can be seen is that reviews whose con-

stituent arguments predominantly shared the same polarity tended to have a numerical score consistent with this polarity. Drug reviews that have neutral numerical ratings often contained predominantly negative or positive arguments causing us to derive a non-neutral argument-based rating. In other cases it was seen that the author would provide positive and negative statements within a single drug review, and whilst the majority of content was homogeneous in its polarity, one statement may have caused the user to rate otherwise.

3.4 Discussion and Literature Review

In this chapter I have presented an argument-based framework for analysing medical drug reviews to be used by patients who are choosing between multiple treatment options. I have shown how simple domain-specific techniques can be used to extract arguments, but this is only so that we have the necessary input for argument-based analysis. Whilst this work is not intended to be a contribution to argument mining, whose motivation is the automated the extraction of argument components, primarily premises and conclusions, from text [56][54][57], I acknowledge that techniques from argument mining could be employed to improve the system.

This work resembles[58] which proposes the use of lookup data in conjunction with argument schemes to mine user generated arguments from online camera reviews. Whilst that paper successfully mines arguments for a specific product, it does not provide an evaluation of arguments mined using any argument solver, whereas evaluating arguments is the primary aim of my experiment.

My approach was to identify a small set of argument types common across all drug reviews and then construct classification rules to extract those argument types. This is in contrast to a manual annotation approach as in [59] which extracted arguments from a set of reviews and put them together in a single argument graph. My approach enabled us to fully automate the entire system, from extraction through to analysis. It also ensured I had to only construct a single set of attack relations which I imposed on all of the drug reviews.

Going forward I will seek to extend the evaluation of the arguments by mak-

ing use of the quantity of arguments populated for a given argument class. I will also consider using preference-based frameworks [60], probabilistic frameworks [44][45] and social abstract argumentation [61] to allow us to model argument types that are more frequent and yield greater influence over the overall patient ratings. I will also investigate the possibility of learning the attack relations by analysing the numerical rating of a drug review and attempting to construct an argument graph such that it is possible to maximise the correlation between the argument-analysis rating and the numerical rating.

Chapter 4

Argument Extraction with Machine Learning

In the previous chapter I demonstrated that a simple rule based approach can be used, with moderate success, to extract arguments from drug reviews. I also introduced the concept of modelling reviews as abstract argument graphs. I demonstrated that through manual analysis of the reviews, it is possible to identify patterns of how the different argument types (theme of the argument, e.g. is it a negative argument, was drug bearable etc) interact with one another and that this knowledge can be harnessed in order to model each review as an abstract argument graph.

There are however certain issues with such a rule based approach for argument extraction. Firstly the approach inevitably suffers from scalability issues when faced with large numbers of reviews which cover a variety of topics and arguments. This is because as the size of the dataset grows it is difficult to construct and maintain a set of rules that can comprehensively capture all of the various argument types and all of the ways in which these argument types are expressed by the reviewers. A much more suitable mechanism in such cases would be to have an mechanism that is capable of learning how such arguments are expressed in the text.

A second issue is related to the method used to produce the abstract argument graphs for each review. In the chapter I presupposed, as a simplification assumption, the attack relations between the argument types was determined based on manual analysis of the reviews (see Figure 3.1). Whilst this approach may work for small

datasets, where it is possible for a human to read through the reviews and identify such patterns of attack between the various argument types, this does not scale as the number of reviews and argument types may increase. This is expected since once we begin to read through reviews of new drugs for diseases that we had not previously read new themes may emerge. For example for we may encounter reviews complaining that drugs are too expensive or hard to find which is something we had not previously encountered. Additionally the previous approach will not reflect the reality in which arguments will exist in various configurations of attack. In other words a static argument framework may not hold true all the time. I address this problem in the next chapter using probabilistic argumentation. In the remainder of this chapter the focus will be on a proposal for a better argument extraction system.

In order to address these issues I decided to use an approach that uses machine learning. Machine learning (ML) approaches are attractive because they can, through sufficient exposure to data, identify latent features in the data that one could not otherwise see nor be able to represent trivially using a rule based approach. The primary constraint with ML approaches however is that they do require significant amounts of labelled training data (when using a supervised learning approach) and this can in general be quite expensive to acquire in that it requires human input and validation.

In the previous chapter I predefined the list of argument types that I was interested in analysing. This list was compiled by manually analysing a number of reviews and observing topics and themes of discussion that were common amongst the reviews. This analysis then enabled me to narrow down a list of topics for which I then wrote a number of rules to identify arguments related to that topic. In order to conduct a larger scale experiment in this chapter however I required a dataset that had more reviews and I did not want to limit the types of arguments I was searching for. Hence I started afresh with a new set of drug reviews.

4.1 Text Classification with Pretrained Models

I frame the machine learning problem as a text classification task. More precisely I would like the model to take as input a review and provide as output the argument types that exist in that review. In this case, since I am starting with an unannotated dataset, for which I do not know what the document classes (argument types) are in advance nor in fact how many document classes there are in total. For a standard text classification task, one knows in advance in classes that can be assigned to the task. The annotators are thus assigning labels to the documents based on this predefined set of classes. In this case however, since the argument types and not known in advance, I expect to discover the argument types as the annotation exercise unfolds. Hence in order to produce labelled data for supervised training the annotation task here is thus to firstly identify all arguments in the training dataset and label them with appropriate labels.

A major bottleneck with text classification problems that has been known for some time is the the annotation process which is the process of acquiring labelled training data. The annotation process almost always requires extensive human validation and is as a result costly. This cost scales with the complexity of the classification task and dataset. Moreover in cases dealing with multiple document classes it can become prohibitively expensive to acquire sufficient labelled data for each class to train a reasonable performing ML model.

To address this there has been significant work done in the literature to reduce the burden of annotation. Recently we have seen the emergence of few-shot learning algorithms which are algorithms that make use of ML models that rely upon transfer learning to reduce the need for labelled data. Few-shot learning approaches typically rely upon large pretrained language models that can then be fine tuned with little supervised data on the document classification task at hand. Whilst these models appear promising the number of labelled data required is still significant enough to be costly.

In order to achieve this I arranged for an annotator to go through 520 drug reviews and highlight parts of the review which they felt constituted an argument in

favour of or against the underlying drug. In addition to this they would also have to assign some sort of label to the highlighted text that indicated the type of argument being captured. The motivation for labelling parts of the text is to build up a lexicon of keywords/phrases for each argument type and subsequently use these in order to faster identify arguments in other reviews containing similar phrases.

An alternative approach that has been around for some time is to provide or collect keywords and phrases for each document class in advance of beginning the annotation process. This list of phrases is then used to pre-emptively label documents with the corresponding class if it contains the same phrase. The list of phrases can be generated in a number of ways. In the first approach during annotation process in which the annotator labels a document with a document class and also records the phrase/keyword within the document related to the class. Another way relies upon having an annotator who is a domain expert in the classification problem. If this is the case the annotator can provide a list of keyword/phrases for each document class before the annotation process begins.

In this case we begin with the assumption that we do not know the document classes for the dataset. It follows then that, of the aforementioned solutions to reducing the annotation burden, the only viable option is to iteratively ask the annotator to jointly provide the argument type label as well as providing the keyword/phrase in the document related to the argument label.

4.2 Dataset

The dataset I chose to use for this chapter is much larger than the one used in the previous chapter. I chose to use an open source drug review dataset [23] compiled from reviews taken from two popular patient based drug review websites called Drugs.com and Drugslib.com which covers over 6886 medications. By reading through a number of reviews it was apparent that the arguments within the reviews typically dealt with issues that were particular to the condition that the medication was treating. Hence reviews for over-the-counter drugs for treating cold and flu were similar in nature and likewise drugs treating acne were similar in content.

This is depicted in the two reviews below. Consider the first review which is for a chemotherapy drug for breast cancer.

> *"I've been on this drug for nine miserable months now. I had said I would try it for one year and I'm not sure I can make that. It appears that quality of life is not something my doctor cares about. It's almost as if he's being paid by the tamoxifen people because he won't even listen to anything negative about the drug. My hot flashes are unbelievable. My joint pain makes it difficult to walk. I haven't slept in nearly a year. This drug makes life not worth living"* - drug review for breast cancer chemotherapy drugs

Medications for serious illnesses and/or terminal illnesses tend to contain significant information of the patient's personal journey and history with the condition. In contrast less serious conditions tend to focus on the absence/presence of a few symptoms and success of the drug.

> *"I took doxycycline with food and a large glass of water both morning and night. I had no side effects whatsoever- no nausea, vomiting, headache, etc. I felt symptom relief after just 1 day. While I'm sure others may have had different experiences- don't let that deter you from trying!"* - drug review for urinary tract infection medication

In order to ensure that these reviews could be successfully annotated I chose to limit the drugs to those relating to a single condition. This meant that I could be confident that I would not be dealing with an intractably large number of argument types and could have a reasonable number of labelled data points for each argument type.

In my case I chose to focus on drugs related to the condition Acne. I chose Acne as the number of available reviews for it was substantially larger than other conditions and furthermore there was a considerable number of drugs that had been reviewed for this condition. In addition to this it was important that I chose a drug whose reviews provided a good balance of positive arguments and negative arguments. I found that the there was a good distribution of numerical ratings for the

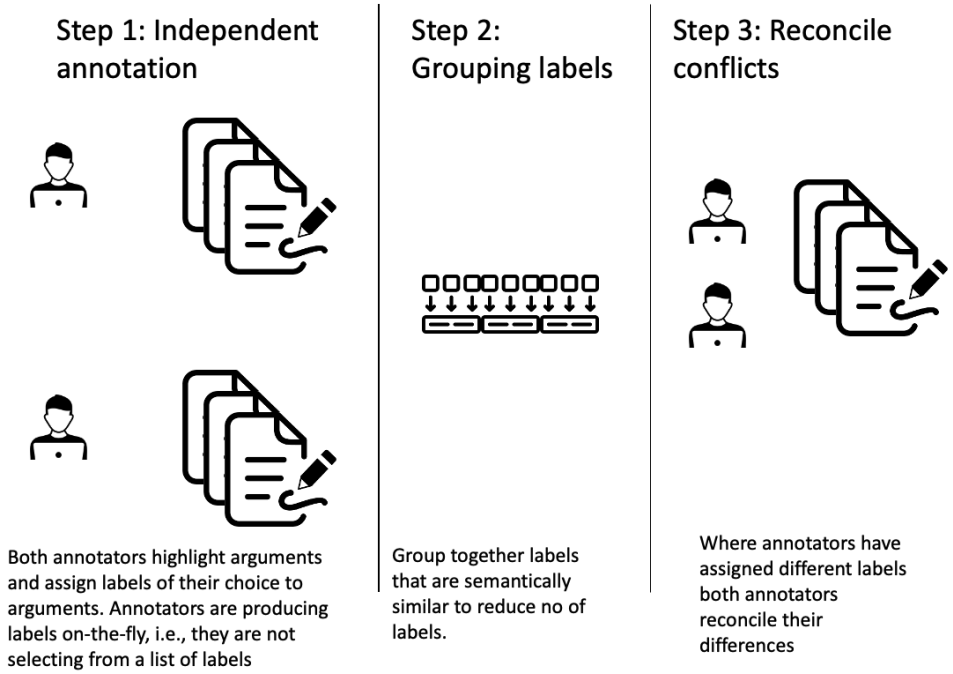


Figure 4.1: Diagram depicting steps to produce annotated dataset

Acne related drug reviews; this is in contrast to some other medications such as those used in chemotherapy to treat cancer.

I sampled a total of 620 reviews pertaining to drugs relate to Acne. Each of the reviews had a numerical rating between 1-10. In order to facilitate the annotation process I developed an annotation interface through which the annotator could input an argument type (document class) as well as highlight the portion of text within the review corresponding to the argument type. The interface, which can be seen in Figure 4.2 is a full stack web application implemented in Python [1]. I chose to develop a custom interface for this as none of the existing open source annotation tools [2] provide the user with the ability to create document class labels on the fly and moreover none of them provide the option of highlighting and associating part of the document to the document class.

The annotation task was to read through the review and highlight portions of the review that the annotator constituted an argument in favour of or against the drug. Once highlighted the annotator was asked to provide a textual description

[1] https://github.com/kawsarnoor/documentannotationtool
[2] BRat, Doccano

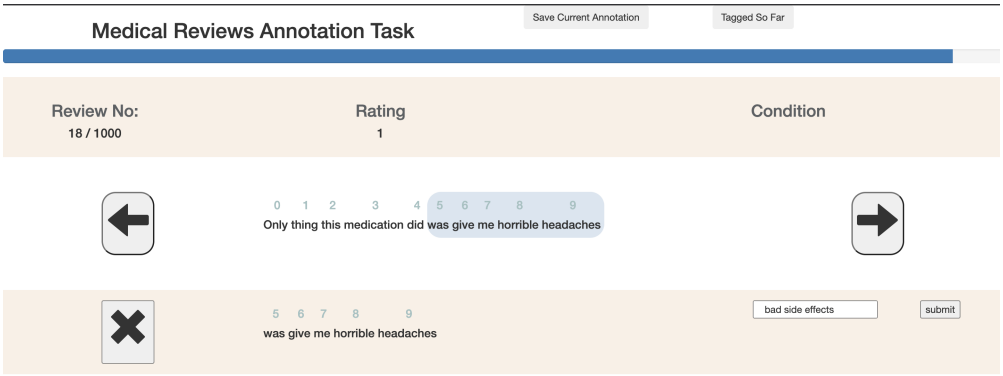


Figure 4.2: Annotation tool used to collect argument types

for the highlighted text; this will become the argument type for that highlighted argument. For example if the annotator highlighted the phrase *'too costly'* then the corresponding argument type they assign might be *'drug expensive'*.

Once both annotators had finished annotating the dataset they had generated 41 unique argument labels between themselves; this was through annotating a total of 2028 spans of text. In total the annotators identified 41 unique argument types. I found there to be a imbalance in distributions of argument labels within the reviews. In order to reduce the imbalance I chose to group together some of the argument labels such that they could be used to describe a single argument type. Examples of this included the labels *'drug worked eventually'* and *'drug worked'*. In total of 33 argument types were produced.

Once these labels were determined there was a final round of annotation in which both annotators sat down and reconciled their differences. A difference was considered to have occurred if (1) Both annotators highlighted text in the same sentence in the same review but assigned different labels (2) One annotator highlighted text that the other did not. I reported an inter-annotator agreement using the kappa score [62] score measure:

$$k = \frac{P_{observed} - P_{chance}}{1 - P_{chance}} \tag{4.1}$$

I reported of a $k = 0.49$. An interpretation of kappa score has been provided

LabelNo	Types	Examples
1	No side effects, minimal side effects	'Never any noticeable side affects'
2	drug not doing job, no change	'It didn t do anything for me'
3	drug worked, drug worked eventually	'Started working after 1week'
4	benefit, some benefit long term benefit	'skin completely cleared'
5	unbearable effect, unexpected side effect	gained 25pounds
6	positive experience good experience	'I m so happy with the results'
7	negative experience	'I would not recommend this pill to anyone'
8	cheap	'Not expensive'
9	side effects go away	'Luckily after a few days the swelling and redness disappeared,
10	side effects	'swollen lymphnodes'
11	bad effect initially, initially bad effects	'I noticed my breakouts have been slightly worse'
12	no change	'have seen no change in my skin'
13	made illness worse	'My acne has become worse'
14	hope effects get better	'hang in there friends'
15	worse when off the drug	'when I stopped my acne issue came back'
16	managing medication	'taking it in the morning with food seemed to eliminate the nausea'
17	some improvement	'I m starting to see improvements'
18	unsure	'Dont know'
19	stopped using	'had to stop using the medications'
20	dangerous side effect	'fainted multiple times'
21	having to remedy side effects	'Moisturiser is a must with this gel'
22	worked very well	'My acne was virtually gone after 2 weeks'
23	unhappy with results	'I wasn t satisfied with the results'
24	not full treatment	'however I still break out quite frequently'
25	bearable side effect	'only minor side effects I noticed were increased thirst and slight dryness'
26	benefit does not last	'I saw no lasting effects'
27	expected more benefit	'not 100 gone'
28	drug worked	'Started working after 1week'
29	scared to stop drug	'I m worried about going off it now what it ll do to my skin'
30	drug not doing job	'On my 3rd month of this medicine and nothing'
31	expensive	'it s not cheap'
32	bad knock on effect	'dried my skin out so badly'
33	short term relief	'some initial improvement in skin'

Table 4.1: Annotation types and examples

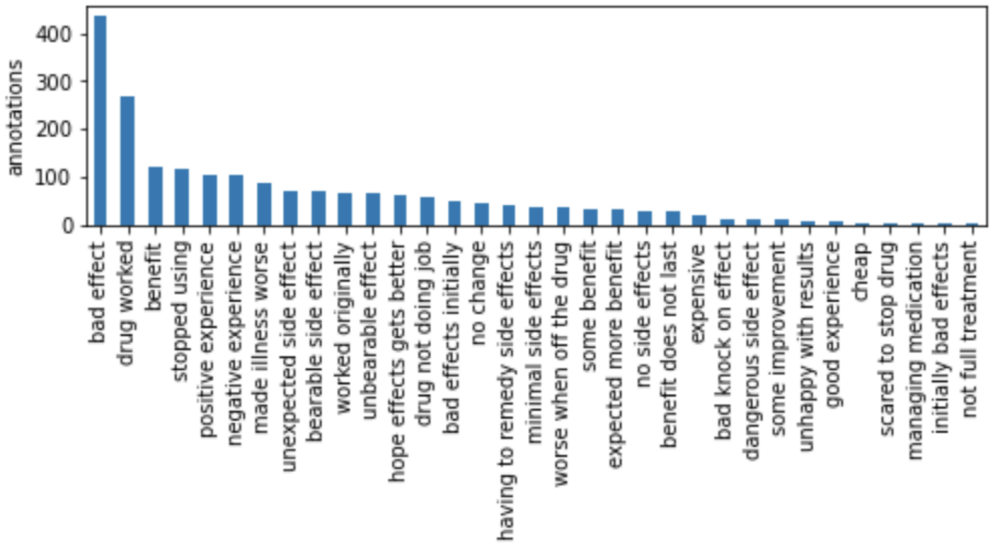


Figure 4.3: distribution of annotations over argument types

in [62]. They suggest that 0.01–0.20 as none to slight agreement, 0.21–0.40 as fair, 0.41– 0.60 as moderate, 0.61–0.80 as substantial, and 0.81–1.00 as almost perfect agreement. In this case there is moderate agreement on annotations. The final dataset is available online [3].

4.3 Training

Following the annotation process I noted that in all the cases where an argument type was identified in a review the corresponding highlighted text was contained within a single sentence. More precisely I found that the arguments did not straddle multiple sentences. This being the case I felt it was more appropriate to treat this as a sentence classification problem. A sentence classification problem is a text classification problem where the unit of text being classified is an individual sentence as oppose to the entire document. I accordingly tokenised each review into its sentences. I followed standard NLP procedures and cleaned common spelling mistakes and replaced punctuation mistakes using an open source NLP toolkit called spaCy [4].

Whilst there are many ways to approach this text classification problem I recog-

[3] https://github.com/robienoor/mlargminingreviews
[4] https://spacy.io/

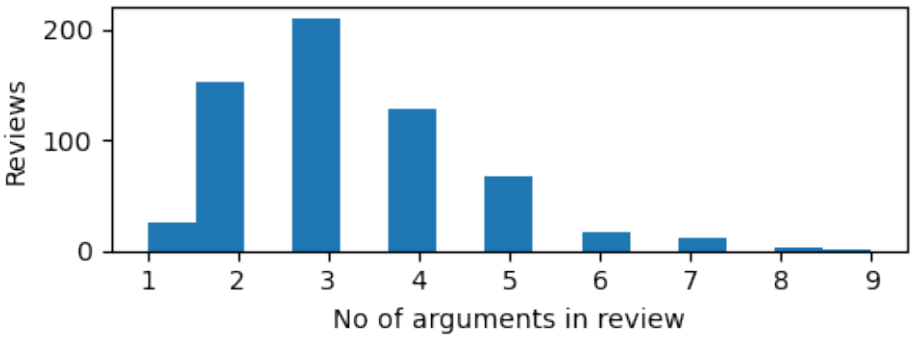


Figure 4.4: frequency plot showing number of arguments in the review

nised that the size of the training dataset was not amenable to simpler supervised machine learning approaches such as Naive Bayes or support vector machines. This is because I did not produce a significant amount of training data through the annotation process. In light of this I focused my attention on recent developments in transfer learning approaches within natural language processing.

4.3.1 Pretrained Text Classification Models

In a traditional supervised learning setting it is assumed that there is some labelled data to do some particular classification or prediction task. The normal procedure is then to partition the labelled data into a training set and testing set. The model is trained to perform well on the training set and the expectation is the the model will have a similar performance on the test set as well. It is however common in the real world, as it is in my case, to be constrained by the amount of labelled data available. To tackle this problem there has been much interest recently in using transfer learning to minimise the labelled data requirement.

Transfer learning is a deep learning technique where a deep learning model is trained on a large dataset (base model) and is then fine-tuned to perform a similar task on a different dataset to produce the final model (fine-tuned model). In NLP a common approach is to use a language model as the base model. A language model is a 'token prediction' model which is trained to predict missing tokens/words in a document. For example given a training sentence '*the cat sat on the mat*' the model would receive as input the string '*the cat sat on the [MASK]*' and be

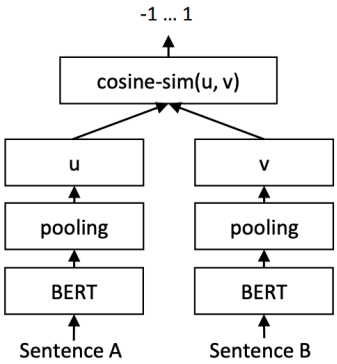


Figure 4.5: SBERT architecture

expected to predict the masked words and hence reproduce the original sentence. The intuition is that if the model can perform this task reasonably well for a wide range of sequences the model parameters have captured certain latent features of the language. his model can then be used in turn to build task specific models that can do things like document classification, sentiment analysis etc. One obvious advantage to using language models as base models for training task specific models is that it does not require any human annotated data; it is an unsupervised learning task.

For sentence-classification I use a SBert model [5]. Whilst more recently the performance of BERT has been surpassed by much larger models (e.g. GPT-3 [63]) they are trainable on consumer harder with a CPU easily hence I opted for BERT. The model architecture, depicted in Figure 4.5 is essentially comprised of two seperate BERT models that take as input a sentence (as a string) and produce an embedding (300 dimensional vector) for each sentence in the pair of sentences being compared. The outputs of both of these two BERT models are then passed through a cosine similarity measure in order to predict the correct similarity score. In the pretrained model I are using the underlying BERT model used in the embedding layer is based on the language model [6] whilst the full SBert model is trained on a combination of a number of larg sentence similarity datasets [64] [65]. In order to fine tune the model for this dataset I needed to produce my own sentence similarity

[5]https://huggingface.co/sentence-transformers/distilbert-base-nli-stsb-quora-ranking
[6]https://huggingface.co/microsoft/mpnet-base

argument label	prec	rec	f1
bad effect	0.72	0.69	0.70
drug worked	0.80	0.71	0.74
benefit	0.75	0.76	0.76
stopped using	0.75	0.76	0.75
positive experience	0.75	0.83	0.78
negative experience	0.73	0.85	0.78
made illness worse	0.7	0.88	0.78
unexpected side effect	0.67	0.9	0.76
bearable side effect	0.66	0.91	0.76
worked originally	0.66	0.91	0.76
unbearable effect	0.66	0.91	0.76
hope effects gets better	0.66	0.92	0.76
drug not doing job	0.65	0.9	0.74
bad effects initially	0.64	0.92	0.74
no change	0.63	0.93	0.74
having to remedy side effects	0.64	0.94	0.75
minimal side effects	0.63	0.93	0.74
worse when off the drug	0.63	0.93	0.74
expected more benefit	0.62	0.93	0.74
some benefit	0.63	0.93	0.74
no side effects	0.63	0.94	0.74
benefit does not last	0.63	0.94	0.75
expensive	0.65	0.94	0.76

Table 4.2: Results from SBert for argument extraction per argument type

dataset where each datapoint in this dataset is comprised of two sentences and a score representing the semantic similarity between the two sentences; in this case a cosine similarity score.

Definition 9. Given some annotated sentences $S = ((s_1, l_1), .., (s_n, l_n))$ we compute the sentence similarity cosine score using the following calculation

$$\mathsf{sentencesimilarity}(s_a, s_b) = \begin{cases} 1 & \text{if} \quad l_a = l_b \\ -1 & \text{otherwise} \end{cases}$$

Example 9. Given the following three labelled sentence s_0= (('I did not experience any side effects', no side effects), s_1= (('it made my acne worse', made illness worse) and s_2= (('no side effects to report', no side effects) we

have the following sentence similarity scores $\mathsf{sentencesimilarity}(s_0, s_1) = -1$, $\mathsf{sentencesimilarity}(s_0, s_2) = 1$ and $\mathsf{sentencesimilarity}(s_1, s_2) = -1$

In my case since I had an imbalance in the distribution of labels per argument type I chose to restrict myself to sample a maximum of 37 sentences per class which I computed based on the median number of labels per argument types. I chose this over the mean as, is visible in Fig 4.4 there is a significant amount of standard deviation. To produce the dataset I then produced all pairs possible from this sample and removed and pair which contained the same original sentence. This resulted in a total of 1,489,620 sentence pairs to train from with an average of 45,140 pairs of sentence pairs for each label.

In order to train the model I split the dataset off into 80:20 training:testing split and conducted 3-fold validation experiment. The model was trained using the sentence pairs that could be constructed using the training dataset. Once the model was trained I computed the labels for the test dataset by computing the cosine similarity between the sentence and each of the sentences in the training dataset.

Definition 10. Given a sentence t, a set of training sentences S and a trained SBert Model M the label of t is given by $\text{argmax}_{(s,l)\in S} M(t,s)$.

Example 10. Given a trained model M, a test sentence t and two labelled training sentences $(s_1, l_1), (s_2, l_2)$ say the model computes the following cosine similarities, $M(t, s_1) = 0.78$ and $M(t, s_2) = 0.89$. The label assigned to t will therefore be l_2.

4.4 Results

Table 4.2 summarises the findings of the experiment. I recorded a macro-recall of 0.87 and a macro-precision of 0.66. Although it is challenging to compare this to existing works, we can look to analogous experiments that achieve macro-F1 scores in the area of 0.85 [66], [67] but had larger annotated datasets (3k-10k). With this in mind and recalling the distribution of annotations per argument type in Figure 5.2 it appears that the performance is inline with the aforementioned works.

Peering into the examples we find that for certain classes such as 'no side effects' the labels were very similar if not exactly the same in many cases hence

being easy for the model to identify. In contrast many of the argument types scored much lower precisions and hence F1 scores. In some cases this was due to the size of the training dataset but moreover some of the argument types were not different enough in order to warrant having a full -1 cosine difference. In future work I intend to investigate how Ican either group together similar argument labels together, e.g. unbearable effect and unexpected side effect.

Figure 4.6 shows a TSne plot produced using the computed embeddings for the annotated sentences. The underlying BERT model encodes using 768 dimensional vector. The TSNe plot is a feature reduction algorithm which allows us to visualise the embeddings to 2 dimensions. TSne plots are used to provide a general sense of similarity between the plotted entities. It may be the case that a lot of information is lost during the feature reduction process but none the less the plot gives us a general sense of similarity between the argument labels.

For each of the argument labels I computed the average embeddings using all sentences in the annotated dataset that have that label. I then plotted these embeddings in the two dimensional plot. Considering that the underlying BERT model did not have any exposure to the actual dataset I still see that it is performs a reasonable job of computing sentence embeddings for sentences that are of a similar nature. I have circled some of the noticeable clusters that form in the plot. It shows that arguments that are similar in nature have some tendency to be closer together in the feature space.

To dig a bit deeper consider Table 4.3. It shows, for each cluster, some examples of the annotated sentences for those labels. I can see that there is some semantic similarity between these sentences and this explains the proximity between them in the TSne plot.

4.5 Relevant Literature

As mentioned at the start of this chapter, I have framed this pipe of the full pipeline as an argument extraction process. In particular I am not extracting argument components in a conventional argument mining sense, where arguments and relations

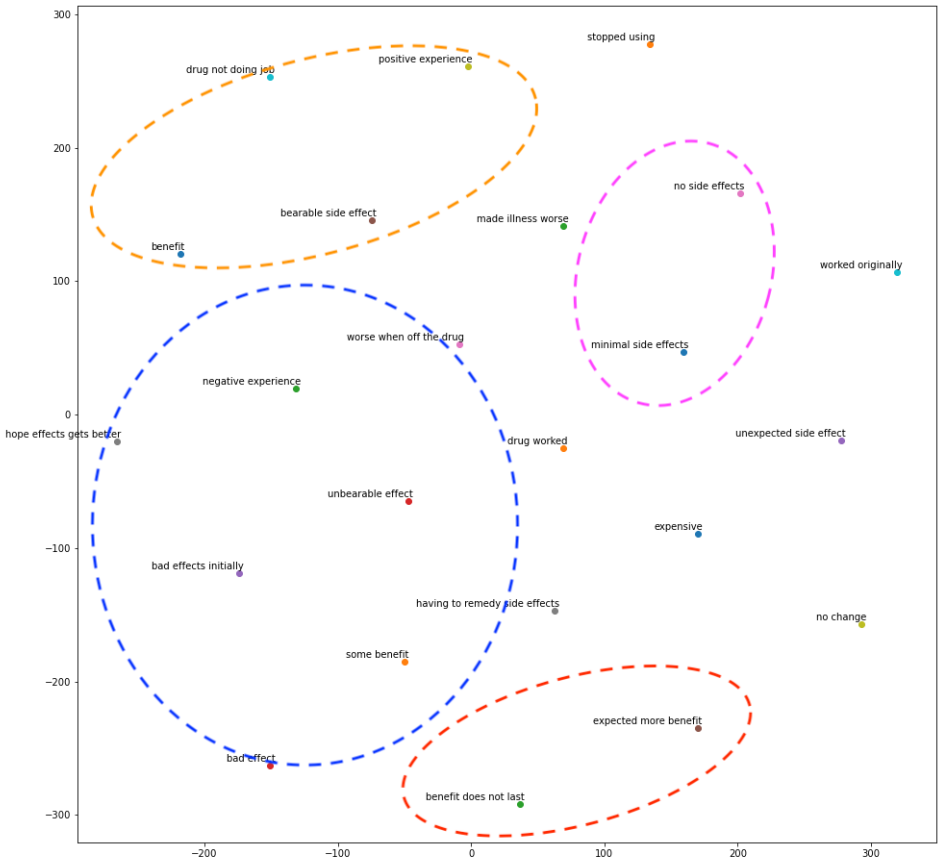


Figure 4.6: TSne plot showing average embeddings for each of the argument labels

are being extracted [68]. Instead I have chosen to frame this as a sentence classification task.

The work presented in this Chapter is thus better described as apsect-based sentiment analysis (ABSA) [69] which is an NLP task concerned with the extraction of 'themes'/'aspects' in text (in particular reviews) and assessing how these 'aspects' contribute to the sentiment of the text. Much work has been done in this and notably deep learning approaches have been a popular direction [70].

ABSA can be broken down into two sub classification tasks. The first is identifying the aspects which in [71] defined as being of two types; explicit in which the

cluster	concept	example
1	no side effects	*'no side effects'*, *'Never any noticeable side affects'*
	minimal side effects	*'i have not had many problems'*, *'my side effects were low'*
2	negative experience	*'Absolutely terrible this pill'*, *'never take this pill'*
	unbearable effect	*'my anxiety is unbearable'*, *'my face got progressively worse'*
3	benefit	*'my skin started to clear'*, *'skin completely cleared'*
	positive experience	*'My skin really cleared up tremendously'*, *'So happy I took this along'*
4	expected more benefit	*'I expected to see better results'*, *'I still have acne scars left'*
	benefit does not last	*'acne began to come back'*, *'my skin is not smooth'*

Table 4.3: Examples of sentences from the annotated dataset. Sentences from the cluster can in many cases have similar semantic meanings

aspect is explicitly mentioned and implicit in which it is implied. In my use case I was dealing with a combination of both. The second step is classifying how the classification task influences the sentiment of the reviewer.

In short ABSA can be viewed as a sentence classification task and hence I make use of the latest models for document/sentence classification [72].

4.6 Conclusion and Future Directions

In this chapter I trained a sentence level argument classification model. I started by using a much larger annotated dataset than in the previous chapter and this time I did not predetermine the argument types and instead allowed the annotator to freely label the arguments during the annotation process. The annotation task involved highlighting the identified argument in review and then assigning the highlighted text with an argument label.

Ifound that most arguments were contained within a small phrase or even just a sentence. This enabled us to phrase this a sentence classification problem. As the dataset I had collected was not very large I wanted to make use of advances in

transfer learning to be able to train a reasonably performing model albeit the data constraints. I chose to use a pretrained SBErt model which is a model that computes sentence similarity based on an underyling BERT model.

I produced a sentence similarity dataset by taking pairs of sentences from the annotated dataset and assigning a similarity score based on the labels assigned. I recorded a macro-F1 of 0.74.

In future I believe that the sentence similarity measure I produced is very simple and can be improved upon. For example in some of the labels, e.g. no side effect and minimal side effect, are quite similar and so the similarity score I assign to these labels should reflect that. On the contrary other labels, e.g. positive experience and unbearable effect, should be maximally different.

It would also be desirable to be able to train my own base BERT model. The underlying a large dataset scraped from the internet. Medical drug reviews present unique semantic features that make it different to other domains. For example side effects are a big factor here and large BERT models may not be able to capture these concepts appropriately. For example *'acne has decreased'* is a benefit and is a good thing in the case of reviews in contrast to *'libido has gone down'*. Training a BERT model from scratch would allow us to ensure the model is able to better capture this level of detail.

Chapter 5

Argument Reasoning using Unsupervised Learning

5.1 Introduction

In this chapter I look closer at the drug reviews and in particular I investigate ways in which I can exploit the relationship between the rating of a review and its constituent arguments. We hypothesise that the rating provided in a review is indicative of the winning set of arguments in a review and propose a method for harnessing this relationship in order to predict the argument graph for that review.

Product reviews are user provided feedback usually presented in the form of a textual review and a numerical rating. They are useful to prospective users of the product seeking information to supplement their decision making process. Take for example a person looking to purchase a specific mobile device. This person will formulate an opinion on the mobile by reading technical articles, marketing material and product reviews. Product reviews differ however from other sources in that they provide first hand accounts of experiences with the product, each effectively serving as a case study into the product's performance with the added advantage of often providing insight into lesser known features experienced only through repetitive use. Furthermore by reading many reviews the user is able to identify popular merits and demerits of the product. We can see that through this process the reader is aggregating arguments and counterarguments for the product.

5.1.1 Reviews as Argument Graphs

A textual review can be seen as a set of arguments, each of which is either for or against the product in question. We call this textual review a collection of polarized arguments. Furthermore whether or not a reviewer is overall in favor of, neutral or against the product can be inferred by the numerical rating. Viewed this way the rating can be seen as a sort of conclusion inferred from the arguments in the textual component of the review.

With an understanding of the arguments that can occur in a review and polarity of the review I have proposed in [27] that reviews can be instantiated into argument graphs. In this chapter I continue this investigation in which I work within the context of medical drug reviews. I propose using medical drug reviews to generate a classifier that can be used to classify reviews/comments that do not come with rating as being either positive, negative or neutral (i.e. extract polarity from reviews).

If we consider each drug review to be a collection of arguments instantiated into an argument graph, we can assume then that the arguments that 'win' in the graph are those that contribute most to the rating. If we think in terms of extensions we might say that, for example the grounded extension of this graph should correlate to the review's numerical rating since this extension captures the conclusion/final sentiment of the review.

Put in other words, the arguments in the textual review whose polarity correlates to the review's numerical rating are those arguments that win and are hence members of the grounded extension. With this in mind I propose that it is possible, given a review whose arguments have been extracted and assessed for polarity, to instantiate all the argument graphs whose grounded extension matches the polarity of the review's rating. By using these argument graphs we are able to generate a probability distribution across them for each review. This idea is formalized later in this paper.

Drug review websites such as Drugs.com, Mayoclinic provide a platform in which medical patients can express their experiences with different medications and treatments. These websites behave much like consumer websites, such as Amazon

and Airbnb, in that they request reviewers to provide a textual review and a numerical rating. Patients who are seeking to use a specific drug may reading these reviews and based on their investigation may chose to act on their findings, or if the medication they are taking is part of a serious treatment, i.e. in the case of cancer, the patient is at least more aware of the benefits and potential harms other users have experienced. I believe such reading also benefits doctors and drug companies who may be seeking for data outside of clinical trials to better understand the performance of a drug. For example drug reviews may help capture sensitivity to side effects or may enlighten certain benefits of the drug that may not have been as visible in controlled trials of the drug.

Upon analysis, it can be seen that the textual reviews of a given drug tend to comment on a specific number of topics. Take for example the reviews of a number of different painkillers. In this case the reviews may discuss topics, such as reason for taking the painkiller, its effectiveness and its side effects. We can think of this as users providing arguments related to a limited set of *argument topics*. More generally we say an *argument topic* is as a theme/topic of discussion. For each *argument topic* arguments can be made to support the topic and by extension support/oppose the drug depending on the topic. As an example if we take the example of side effects from the painkiller, we can imagine arguments around this topic being that there were no side effects, there were many side effects or even there were side effect but they were bearable. Each of these arguments have a polarity in that they either support the drug or oppose it to some degree. I refer to these polar arguments as *'types of arguments'*. The complete collection of these *argument types* for all *argument topics* I define as the set of *argument types* (ATs).

Each review can hence be seen as a collection of different *argument types*. A user may provide conflicting *argument types* in a single review. The user may then go on to rate the product positively with a high numerical score. What we would intuit in this case is that, overall the positive arguments were of higher importance to the user than the negative arguments thus explaining the high numerical rating.

By analyzing many reviews of the same product/ type of product, we begin

to develop an understanding as to how the argument types interact, based on an aggregation of their interactions as observed within individual reviews. I begin by introducing background definitions and then move onto formalising our approach in Section 5.3. Following this I undertake an experiment using a set of annotated reviews.

5.2 Modelling a Review with Probabilistic Argumentation

5.2.1 Probabilistic Argumentation

I introduced the definitions for probabilistic argumentation in Section 2.2. Let us recall some of the extensions that can be ascertained once we have identified an acceptable set of arguments in an argument graph. Given an acceptable set there are the following extensions that can be attained are:

1. **Complete extension** (co) - A set Γ is a complete extension *iff* Γ is defended
2. **Grounded extension** (gr) - A set Γ is a grounded extension *iff* it is the minimal (w.r.t set inclusion) complete extension
3. **Preferred extension** (pr)- A set Γ is a grounded extension *iff* it is the maximal (w.r.t set inclusion) complete extension
4. **Stable extension** (st)- A set Γ is a stable extension *iff* it attacks every argument not in its set

I use the notation Ext_σ where $\sigma \in \{\mathsf{co}, \mathsf{gr}, \mathsf{pr}, \mathsf{st}\}$ to denote the arguments in extension σ.

Let us recall that the constellations approach to probabilsitic argumentation introduced in [45] which builds upon foundations laid out in [44] [73]. The constellations approach considers a probability distribution over a set of subgraphs comprising of arguments A. I recall Definition 7 for distinguishing between the two types of subgraphs that can occur.

Definition 11. An argument framework $G = (A,R)$ can be decomposed into a subgraph $G' = (A',R')$ where $G' \sqsubseteq G$ *iff* $A' \subseteq A$ and $R' \subseteq (A' \times A') \cap R$. A set of subgraphs of G can be denoted $\mathsf{Sub}(G) = \{G' | G' \sqsubseteq G\}$. The subset G' is considered **full** *iff* $A' \subseteq A$ and $R' = (A' \times A') \cap R$. A subgraph G' is considered **spanning** *iff* $A' = A$ and $R' \sqsubseteq A$.

Using Definition 11 we provide some auxiliary definitions to help define the set of all spanning subgraphs for a graph G and similarly the set of all full subgraphs.

Definition 12. Given a graph $G = (A,R)$ we denote the set of all spanning subgraphs for G as $\mathsf{Spanning}(G)$ and the set of all full graphs for G as $\mathsf{Full}(G)$.

Definition 13. A subgraph distribution is a function $P^C : \mathsf{Sub}(G) \rightarrow [0,1]$ where the sum of all probabilities for each graph in $G' \in \mathsf{Sub}(G) = 1$, i.e. $\sum_{G' \sqsubseteq Sub(G)} P^C(G') = 1$. Using Definition 11 we attain a **full subgraph distribution** when P^C is applied the set of full subgraphs of G. Likewise a **spanning subgraph distribution** when P^C is applied to the set of spanning subgraphs of G.

Given a subgraph distribution it is possible to determine the probability with which a a set of arguments is in an extension $\sigma \in \{\text{pr,co,st,gr}\}$.

Definition 14. For $E \subseteq A$ the probability that $L : E \rightarrow \{\text{in,out,un}\}$ is a σ set is given by the equation:

$$P_\sigma(L) = \sum_{G' \in Sub(G) s.t. L \in \sigma(G')} P^C(G') \tag{5.1}$$

Example 11. Table 5.1 shows the full subgraph for argument graphs containing two arguments (graphs $G_1, G_2, G_3, G_4, G_5, G_6, G_7$). The spanning subgraphs are G_1, G_2, G_3.

The constellations approach is relevant to our task as we have a collection of reviews where the arguments are known and whereas the attacks between arguments are not. In the following section I outline my approach to dealing with this problem.

In what follows I propose a method for generating a probability distribution over the space of possible arguments graphs assignable to a review; I refer to this

Graph	Structure	Probability
G_1	x $\leftarrow$ y	1/6
G_2	x $\rightarrow$ y	1/6
G_3	x $\leftrightarrow$ y	1/6
G_4	x y	1/6
G_5	x	1/6
G_6	y	1/6
G_7	$\emptyset$	1/6

Table 5.1: A uniform full subgraph distribution for two conflicting arguments

space of possible graphs as the **graph space**. More specifically I assume a relationship between the rating provided and the arguments that exist in the grounded extension of the review. Using this relationship we construct a (prior) distribution over the graph space from which we can then sample a graph and assign it to the review. These probability distributions provide us an aggregated representation of the set of reviews we are looking at, and can be used in other downstream reasoning tasks, as well as be used to assign graphs to reviews.

Further to this I investigate extracting additional information about arguments in a set of reviews. In particular I evaluate the overall impact of an argument by considering the effect it has had on the reviews in which it has appeared. This additional information can then be used to update the probability distribution over the space of graphs.

I conclude by demonstrating that my setup is flexible and can be used in conjunction with off the shelf machine learning algorithms to build predictive models for assigning argument graphs to reviews that do not have ratings.

My proposal is not limited to product reviews and can indeed be used in any situation in which I have the potential to model statements as a set of arguments where each statement is accompanied by a proxy measure indicating which arguments win in the tuple. For example consider a television debate being watched by a number of viewers. Each viewer, over the course of the debate, accumulates a number of arguments from both parties, and makes an overall assessment indicating which party they felt won the debate. My approach could thus be used to build probability distributions over the set of arguments and can thus be used to assign

graphs to each viewer.

To summarise I make primarily two contributions with this chapter. The first contribution is providing a methodology for identifying probability distributions over constellations of argument graphs based on a set of arguments and a proxy measure which I use to infer which arguments are mostly likely to be acceptable.

The second contribution is refining this distribution using real data. More specifically I provide a method for incorporating statistical data about the arguments in a set of reviews in order to modify the probability distribution such that is better reflects the reviews that exist in the dataset.

5.2.2 Analysing the Polarity of an Argument Graph

I start by defining the set of all possible argument graphs that can be constructed given a set of positive and negative arguments; i.e the graph space. I make the assumption that a review will only be able to have graphs assigned to it that contain all of the arguments in the review; in other words only spanning subgraphs that can be constructed using the arguments in the review. What this entails is choosing only the set of bipartite spanning subgraphs that can be produced using the arguments found in the review. This is important to note because when constructing the graph space we can reduce our computational cost by only considering the graphs needed in order to analyse a review.

Definition 15. Let $\mathscr{A}^+$ be the set of all positive arguments and $\mathscr{A}^-$ the set of all negative arguments. Given a subset of positive arguments $A^+ \subseteq \mathscr{A}^+$ and negative arguments $A^- \subseteq \mathscr{A}^-$ an argument graph space S is the set of all bipartite spanning subgraphs, i.e: $S = \{(A^+ \cup A^-, R) | R \in \mathscr{P}(A^+ \times A^- \cup A^- \times A^+)\}$.

Proposition 1. Given a set of positive and negative arguments A^+ and A^- let $m = |A^+|$ and $n = |A^-|$. The size of the space of graphs is then 2^{2mn}.

Proof. This number of graphs in the graph space is given by the number of unique combinations of attacks we can between positive arguments ($m = A^+$) and negative arguments ($n = A^-$) excluding self attacks. This is given by the power set 2^{2mn}. □

Once we have the space of argument graphs we can then begin to analyse each graph in the graph space in terms of its grounded extension. Although other semantics can be considered I have chosen to focus on the grounded extension as a simplfication assumption and intend to revisit other extensions in future works. Below I define a number of auxiliary functions needed to conduct analysis on this.

Definition 16. Let A^+ be a set of positive arguments and A^- a set of negative arguments and S be the corresponding graph space. For each graph $G \in S$ we define the following sets $\mathsf{groundArgs}^+(g) = \{a \in \mathsf{gr}(G) | a \in A^+\}$ and $\mathsf{groundArgs}^-(g) = \{a \in \mathsf{gr}(G) | a \in A^-\}$. Using these sets we can then define the overall polarity of a graphs that have the grounded extension g using the function below.

$$\mathsf{Pol}(G) = |\mathsf{groundArgs}^+(G)| - |\mathsf{groundArgs}^-(G)|$$

The function Pol provides a simple mechanism for ordering the graph space by considering the overall polarity of the grounded extension of a graph. It allows is to represent the graph space as a sequence containing all graphs in S s.t. for any two graphs $G_i, G_j \in S$ it holds that $\mathsf{Pol}(G_i) \geq \mathsf{Pol}(G_j)$.

Example 12. Consider a situation where we have a positive argument a and a negative argument b. The number of graphs in graph space S is 2^2 according to Proposition 1. The value assigned to each graph in S by Pol is depicted in Table 5.2. We can see that the graphs are sorted their Pol score s.t $\mathsf{Pol}(G_1) \geq \mathsf{Pol}(G_2)$ etc.

Proposition 2. $\mathsf{Pol}(G) \in \mathbb{Z}$ and $|A^-| \leq \mathsf{Pol}(G) \leq |A^+|$

Proof. The value for $\mathsf{Pol}(G)$ is dependent on the two terms $|\mathsf{groundArgs}^+(G)|$ and $|\mathsf{groundArgs}^-(G)|$. The largest value each term can take is $|A^+|$ and $|A^-|$ respectively given that those are the maximum number of arguments that can end up in the grounded extension for each respective polarity. It therefore follows that $\mathsf{Pol}(G) \in \mathbb{Z}$ and $|A^-| \leq \mathsf{Pol}(G) \leq |A^+|$. □

I stipulate that the rating scale for the reviews are to be partitioned into three categories; namely positive, neutral and negative. In my case I consider a rating

No	G	Pol(G)	P(G)		
			Pos(s)	Ntl(S)	Neg(S)
G_1	a $\rightarrow$ b	1	1	0	0
G_2	a b	0	0	0.5	0
G_3	a $\leftrightarrow$ b	0	0	0.5	0
G_4	a $\leftarrow$ b	-1	0	0	1

Table 5.2: The probability distribution using the function Candidates on a graph space S where $A = \{a, b\}$ where a is a positive argument and b a negative argument

scale between 1 and 10. In the case that the ratings do not follow a 10-point scale, normalisation can be used to bring the values within the 1-10 scale.

With these partitions in mind we partition the graph space based on the overall polarity of the grounded extensions. In other words we say that graphs that have more positive arguments in its grounded extension than negative are considered positive and so on.

Definition 17. Given a graph space S with grounded extension $\mathsf{Gr}(S)$ the sets of positive $\mathsf{Pos}(S)$, neutral $\mathsf{Ntl}(S)$ and negative $\mathsf{Neg}(S)$ grounded extensions are:

$$\mathsf{Pos}(S) = \{G \in S | \mathsf{Pol}(\mathsf{gr}(G)) > 0\}$$

$$\mathsf{Ntl}(S) = \{G \in S | \mathsf{Pol}(\mathsf{gr}(G)) = 0\}$$

$$\mathsf{Neg}(S) = \{G \in S | \mathsf{Pol}(\mathsf{gr}(G)) < 0\}$$

In order to relate the polarity of the grounded extension of the graph to the numerical rating I propose a simple function that assigns a review a set of graphs based on the polarity of its rating. We say the polarity of a rating r is negative if $r \leq 4$, neutral if $5 \leq r \leq 7$ and positive if $r \geq 8$. These boundaries are a continuation of those seen in Chapter 3.

Definition 18. Let A^+ be a set of positive arguments, A^- a set of negative arguments and S the corresponding graph space. Given a review (A, r) where $A = A^+ \cup A^-$ and r is a rating s.t $1 \leq r \leq 10$ the set of candidate graphs for the review is given by the function:

$$\mathsf{Candidates}(S,r) = \begin{cases} \mathsf{Pos}(S) & \text{if} \quad r \geq 8, \\ \mathsf{Ntl}(S) & \text{if} \quad 5 \leq r \leq 7, \\ \mathsf{Neg}(S) & \text{if} \quad r \leq 4, \end{cases}$$

With the rating scale partitioned into the three polarities we can then begin to assign probabilities to each graph.

Definition 19. Given a review (A,r) and a set of candidate graphs $\mathsf{Candidates}(S,r)$ a probability distribution over $\mathsf{Candidates}(S,r)$ is a function $P : \mathsf{Candidates}(S,r) \to [0,1]$

At this point we have no reason to believe one candidate graph is more probable than another and hence we can distribute probability mass over each graph uniformly.

Definition 20. Given a review (A,r) and a set of candidate graphs $\mathsf{Candidates}(S,r)$ a uniform probability for a graphs $G \in \mathsf{Candidates}(S,r)$ is given by the function:

$$P(S,G,r) = \begin{cases} \dfrac{1}{|\mathsf{Candidates}(S,r)|} & \text{if} \quad G \in \mathsf{Candidates}(S,r) \\ 0 & \text{otherwise} \end{cases}$$

Example 13. Continuing with the example from Table 5.2, we can see the three possible probability distributions corresponding to the three possible polarities.

Proposition 3. Given a graph space S and a rating r, it holds that $\sum_{G \in S} P(S,G,r) = 1$.

Proof. The three possible resulting sets $(\mathsf{Pos}(S), \mathsf{Neg}(S), \mathsf{Ntl}(S))$ from $\mathsf{Candidates}(S,r)$ are disjoint sets. It follows then that $P(S,G,r)$ places non-negative mass only on the set $\mathsf{Candidates}(S,r)$ with mass uniformly distributed across each graph in that set. Uniform distributions are valid probability distributions. □

If we now wish to assign a graph to a review based on its rating r we can retrieve the set of candidate graphs using Candidates and use the probability mass

function P in order to identify a probability distribution; the graph can then be assigned by sampling from this (multinomial) distribution.

Proposition 4. Any graph space in which there are m positive arguments and n negative arguments there exists an asymmetry in the size of the sets $\mathsf{Pos}(S)$ and $\mathsf{Neg}(S)$ s.t $|\mathsf{Pos}(S)| > |\mathsf{Neg}(S)|$ when $m > n$ and $|\mathsf{Neg}(S)| > |\mathsf{Pos}(S)|$ when $n > m$.

Proof. The size of $|\mathsf{Pos}(S)|, |\mathsf{Neg}(S)|$ vary in accordance to how many graphs in the graph space have grounded extension with a corresponding polarity. When the number of positive graphs, m, is greater than the number of negative graphs, n, there are more ways that a positive argument can end up in the grounded extension therefore a larger proportion of the graphs in the graph space S have a positive arguments in their grounded extension. This is where the asymmetry occurs. □

The consequence of this proposition is that when such an asymmetry exists I distribute the probability mass thinly over the larger polarity set in contrast to the smaller polarity set. In this case if we compare the larger set to the smaller set, because both are uniform distributions, we are more likely to make the correct graph assignment when the probability mass is spread over then smaller set as there are fewer graphs to sample from.

Up until this stage we have relied upon the simple intuition that the grounded extension of a graph in a graph space has an inherent polarity and that this polarity can be aligned with the polarity of the rating provided in a review. We are yet to discriminate between argument graphs in a set of polarity argument graphs, e.g the set $\mathsf{Pos}(S)$. Although the polarity of the grounded extension is a good starting point for determining the overall polarity of a graph, we can go further and analyse the graph structure to make more granular distinctions in the degrees of polarity between the different graphs.

5.2.3 Ranking Argument Graphs

Classical semantics, such as the grounded extension, provide acceptability statuses of arguments by analysing the acceptability of the arguments in subsets of all arguments in a graph. These classical semantics provide an assessment of the accept-

ability of arguments in absolute terms and whilst this may be sufficient for some applications, in many cases we require a finer grained interpretation of an argument's acceptability. To illustrate in some applications when an argument attacks another we may wish to model the attacked argument as being weakened as opposed to being completely defeated, in other words we wish to quantify the degree of acceptability of arguments.

Graded semantics were introduced to bridge this gap [40]. A graded semantics is a function that assigns arguments in an argument framework a numerical value indicating the degree of its acceptability.

For our purposes we are only interested in making distinctions between arguments in a set of candidate graphs; the starting point for this is to understand the acceptability of an argument in an argument graph. In principle any graded semantics could be used; for simplicity I assumed a very simple function based on the number of attacks an argument receives and inflicts.

Definition 21. Given an argument graph $G = (A,R)$ and an argument $a \in A$, the number of attacks a inflicts is $\mathsf{att}_G(a) = |(x,y) \in R | x = a\}|$ and the number of attacks it receives is $\mathsf{def}_G(a) = |(x,y) \in R | y = a\}|$. Using this I define a combined score Grade below:

$$\mathsf{Grade}(G,a) = \mathsf{att}_G(a) - \mathsf{def}_G(a)$$

Example 14. Consider an argument graph $\{\{a,b\},\{(a,b)\}\}$. In this case $\mathsf{Grade}(G,a) = 1$ and $\mathsf{Grade}(G,b) = -1$.

Proposition 5. Let $\mathsf{Space}(A^+,A^-)$ be a graph space. Given $B \in \{A^+,A^-\}$ and $a \in B$ then the maximum value for a can be assigned for *Grade* w.r.t the graph space is given by $\mathsf{max}_{G \in \mathsf{Space}(A^+,A^-)}\mathsf{Grade}(G,a) = |A^+ \cup A^- \setminus B|$ and the minimum by $\mathsf{min}_{G \in \mathsf{Space}(A^+,A^-)}\mathsf{Grade}(G,a) = -|A^+ \cup A^- \setminus B|$.

Proof. The value for $Grade(G,a)$ is maximal if a attacks all its opponents and is not attacked in return. If considered in the context of a graph space it means a is attacking all the opponents in the set of arguments with the opposite polarity. The opposite is true for minimum value for *Grade*. □

Proposition 6. Given an argument graph G it holds that $\sum_{a\in A^+} \mathsf{Grade}(a,G) + \sum_{b\in A^-} \mathsf{Grade}(b,G) = 0$.

Proof. For ease of notation I specify that for a set of arguments $B \in \{A^+, A^-\}$ the following sets exists: $B_{\mathsf{att}} = \{\mathsf{att}_{\mathsf{G}}(a) | a \in B\}$ and $B_{\mathsf{def}} = \{\mathsf{def}_{\mathsf{G}}(a) | a \in B\}$. I then make the following observations $\sum_{a\in A^+} \mathsf{Grade}(G,a) = |A^+_{\mathsf{att}}| - |A^+_{\mathsf{def}}|$ and $\sum_{a\in A^-} \mathsf{Grade}(G,a) = |A^-_{\mathsf{att}}| - |A^-_{\mathsf{def}}|$. Given that G is a bipartite graphs it follows, given that $G = (A,R)$ then $\forall (a,b) \in R$ if $a \in A^+ \rightarrow b \in A^-$ and likewise if $a \in A^- \rightarrow b \in A^+$. From this we see that the following is true $|A^+_{\mathsf{att}}| = |A^-_{\mathsf{def}}|$ and $|A^-_{\mathsf{att}}| = |A^+_{\mathsf{def}}|$. If we make the substitutions it follows that $\sum_{a\in A^+} \mathsf{Grade}(G,a) + \sum_{b\in A^-} \mathsf{Grade}(G,b) = |A^+_{\mathsf{att}}| + |A^+_{\mathsf{def}}| - |A^-_{\mathsf{att}}| - |A^-_{\mathsf{def}}| = 0$ □

Whilst graded semantics assigns numerical values to arguments in the context of single argument graph we are interested in something different; namely scoring an argument in an argument graph relative to all the scores it has been assigned in the rest of the graph space. My aim is to create a function that allows us to ultimately compare graphs against each other.

Consider graphs G_1 and G_2 in Table 5.4. We can see that in G_1 both of the positive arguments are maximally attacking their opponent b and are not facing any retaliation; this can be interpreted as meaning that G_1 is most positive graph in the graph space. In contrast argument a in G_2 is not engaging in any conflict and therefore, although it is equally as acceptable under a grounded semantics, we wish to make the draw out that relative to its performance in G_1 it is under performing.

In the following I formalise this concept.

Definition 22. Given a set of arguments A and a graph space S the smallest and largest value that can be assigned to an argument $a \in A$ is given by the functions:

$$\mathsf{min}(a,S) = \min_{G\in\mathscr{S}} \mathsf{Grade}(G,a) \quad \mathsf{max}(a,S) = \max_{G\in\mathscr{S}} \mathsf{Grade}(G,a)$$

The min-max normalised graded function is then a function $\mathsf{NormGrade} : a \rightarrow [0,1]$ which we define as follows:

$$\mathsf{NormGrade}(G,S,a) = \frac{\mathsf{Grade}(G,a) - \mathsf{min}(a,S)}{\mathsf{max}(a,S) - \mathsf{min}(a,S)}$$

Proposition 7. $0 \leq \mathsf{NormGrade}(G,S,a) \leq 1$

Proof. When $\mathsf{Grade}(G,a) = \mathsf{min}(a,S)$ the function $\mathsf{NormGrade}(G,S,a) = 0$ and likewise when $\mathsf{Grade}(G,a) = \mathsf{max}(a,S)$ the function $\mathsf{NormGrade}(G,S,a) = 1$ returns 0 □

Example 15. Table 5.3 depicts the graph space of two arguments a and b. The maximum value achievable by either argument is in attacking the other whilst not being attacked. Hence we see that $\mathsf{min}(a,S) = \mathsf{min}(b,S)$. The opposite is true for the minimum score assignable in which case $\mathsf{min}(a,S) = \mathsf{min}(b,S) = -1$. The Table then depicts the normalisation of each argument based on the Grad score assigned to it in a graph w.r.t to the aforementioned minimum and maximum values.

Given this we can now define a function which assigns the entire graph a score based on the graded semantics of its constituent arguments

Definition 23. Given a set of arguments $A = A^+ \cup A^-$, where $A^+ \cap A^- = \emptyset$, and a corresponding graph space S the aggregate graded score for a graph $G \in S$ we say that the aggregate score for positive arguments in graph G is $AttackScore^+ = \sum_{a \in A^+} \mathsf{NormGrade}(G,S,a)$ and the aggregate score for negative arguments in graph G is $AttackScore^- = \sum_{a \in A^-} \mathsf{NormGrade}(G,S,a)$. The aggregate polarity score for the graph is then given by:

$$\mathsf{AttackScore}(S,G) = AttackScore^+ - AttackScore^-$$

Example 16. Table 5.3 depicts the aggregate score of each graph assigned using AttackScore. In this case the outcomes are aligned with the polarity of the grounded extensions and we see that the aggregate scores differ change with the polarity of the grounded extension of the graphs.

A nice consequence of the function AttackScore is that if we consider the ordered set of attack scores assigned to graphs in a graph space we find that the difference between them is a constant which I refer to as ΔAtt.

Proposition 8. Let $AttackScores = (AttackScore_0, .., AttackScore_{m-1})$ be a sequence of all the attack scores in the set $\{\mathsf{AttackScore}(G) | G \in S\}$ s.t for any two values in the set $AttackScore_i > AttackScore_{i+1}$. It then holds that the pairwise difference between any two consecutive values in the sequence is a constant ΔAtt i.e. $\Delta Att = AttackScore_i - AttackScore_{i+1} = AttackScore_{i+1} - AttackScore_{i+2}$.

Proof. Let $g = \sum_{a \in A^+} \mathsf{Grade}(a, G)$. Using this in conjunction with the equation in Proposition 6 we find that $\sum_{a \in A^-} \mathsf{Grade}(a, G) = -g$. We then make the following substitution using the definition for NormGrad and AttackScore.

$$AttackScore_i = \frac{g - i - p(min^+)}{max^+ - min^+} + \frac{g - i + n(min^-)}{max^- - min^-}$$

□

If we then take the difference between two consecutive attack scores we find that $\Delta Att = AttackScore_i - AttackScore_{i-1}$ and we substitute the values for $max^+, min^+, max^-, min^-$ from Proposition 6, we find the equation reduces to the constant:

$$\Delta Att = \frac{1}{2n} + \frac{1}{2p}$$

Example 17. To illustrate the result in Proposition 8 consider the situation depicted in Table 5.3 where $n = p = 1$. We find that $AttackScores = (1, 0, 0, -1)$. The $\Delta Att = 1$ in this case. If we consider the setting in Table 5.4 where $n = 1, p = 2$ we see that $AttackScores = (2, 1.25, 0.5, -0.25, -1)$ an accordingly the $\Delta Att = 0.75$.

The score assigned to each graph by AttackScore is sufficient to give a basic ordering over the graphs in a graph space. However in certain cases, such as if the graph space contains many arguments or there are more positive than negative arguments, we see that it assigns the same score to some graphs even if they have different polarities. Take for example Table 5.4 in which graphs $\mathsf{AttackScore}(G_7) = \mathsf{AttackScore}(G_{11}) = 0.5$ yet if we look at the grounded extension we see that $\mathsf{Gr}(G_7)$ is positive as can be whilst $\mathsf{Gr}(G_{10})$ is neutral.

No	Graph	Grade		AttackScore
		a	**b**	
G_1	a $\rightarrow$ b	1	0	1
G_2	a b	0.5	0.5	0
G_3	a $\leftrightarrow$ b	0.5	0.5	0
G_4	a $\leftarrow$ b	0	1	-1

Table 5.3: Graded scores for arguments a, b and aggregate scores for each graph

In order to make this distinction I incorporate the Pol function to give us a final aggregate score for each argument graphs as follow. I start by sorting the graph space by AttackScore and then Pol.

I then consider each subset of graphs in the set $\{\mathsf{Pos}(S), \mathsf{Ntl}(S), \mathsf{Neg}(S)\}$. I order the graphs in a polarity set assessing the score obtained using the pol function.

Definition 24. Given arguments $A = A^+ \cup A^-$ where $A^+ \cap A^- = \emptyset$, a corresponding graph space S, and set of polarity argument sets $Polarity = \{\mathsf{Pos}(S), \mathsf{Ntl}(S), \mathsf{Neg}(S)\}$. For each polarity set $polarity \in Polarity$ we define a sequence $(Gr_1 ... Gr_N)$ of all grounded extensions in $\{\mathsf{gr}(G) | G \in polarity\}$ where $N = |\{\mathsf{gr}(G) | G \in polarity\}|$ s.t for any two grounded extensions Gr_i, Gr_{i+1} it holds that $\mathsf{Pol}(Gr_i) \geq \mathsf{Pol}(Gr_{i+1})$.

This first stage of sorting ensures that graphs in a polarity set are arranged according to the polarity of the grounded extension. I choose to sort first by pol rather than AttackScore as the AttackScore values are non-unique and do not alone differentiate between graphs which have a high pol score.

Example 18. Consider the set of positive graphs in Table 5.4. I see that graphs $G6$ - $G10$ all share the same Agg values although the polarity of the graphs differ. A polarity based ordering ensures that graphs $\{G6, G7\}$ are ranked higher than $\{G8, G9, G10\}$.

I then sort the resulting sequence w.r.t to grounded extension using the AttackScore score.

Definition 25. Given arguments $A = A^+ \cup A^-$ where $A^+ \cap A^- = \emptyset$, a corresponding graph space S and the set of possible grounded extensions $\mathsf{Gr}(S)$ let $\mathsf{gr}_{Gr}(S) = \{G \in$

$S|\mathsf{gr}(G) = Gr\}$ be the set of graphs for a grounded extension $Gr \in \mathsf{Gr}(S)$. We then say that $(G_1...G_M)$ is a sequence of all graphs in $\mathsf{gr}_{Gr}(S)$ where $M = |\mathsf{gr}_{Gr}(S)|$ s.t for any two graphs G_i, G_{i+1} it holds that $\mathsf{AttackScore}(S, G_i) \geq \mathsf{AttackScore}(S, G_{i+1})$.

Example 19. Continuing on from Example 18 we see that there are two sets of Pol scores: graphs with pol score of 2 are $\{G1..G5\}$ and the graphs with score of 1 are $\{G8, G9, G10\}$. For graphs with score of 1 we see that $G1$ is ranked highest with Agg score of 1 and the rest are equally ranked.

With this ordering over the graph space we then proceed to compute an aggregate score for each argument graphs which combines both the Pol and AttackScore.

Definition 26. Let S be a graph space and $(G_1, .., G_m)$ be a sequence of all the graphs in sorted by AttackScore and then Pol. We define that for any two graphs G_i, G_{i+1} that $\mathsf{similar}(G_i, G_{i+1})$ holds iff $(\mathsf{AttackScore}(\mathsf{S}, \mathsf{G_i}) = \mathsf{AttackScore}(\mathsf{S}, \mathsf{G_i})) \wedge (\mathsf{Pol}(\mathsf{G_i}) = \mathsf{Pol}(\mathsf{G_i}))$ otherwise $\neg\mathsf{similar}(G_i, G_{i+1})$. We then define the aggregate score of a graph below where we assume $\mathsf{Agg}(G_0) = \mathsf{AttackScore}(S, G_1)$:

$$\mathsf{Agg}(G_i) = \begin{cases} \mathsf{Agg}(G_{i-1}) & \text{if} \quad \mathsf{similar}(G_i, G_{i+1}) \\ \mathsf{Agg}(G_{i-1}) + \Delta Att & \text{if} \quad \neg\mathsf{similar}(G_i, G_{i+1}) \end{cases}$$

Example 20. Consider graphs G_6, G_8 from the example in Table 5.4. The attack scores are all the same i.e $\mathsf{AttackScore}(S, G_6) = \mathsf{AttackScore}(S, G_8) = 0.5$. If we look carefully at the grounded extension of each we see that $\mathsf{Pol}(G_6) > \mathsf{Pol}(G_8)$. The Agg function picks up on this and lowers the score it assigns to G_8.

The next step is to align the ordered graph space to the numerical rating; so far we have assumed a 1-10 rating scale. We do this by assigning each of the polar sets $\mathsf{Pos}(S), \mathsf{Ntl}(S), \mathsf{Neg}(S)$ upper and lower bounds within the rating scale, i.e: positive graphs are those which have ratings from 8-10. We can then analyse the graph space to see what are the corresponding values assigned by the function Agg for these bounds. With the rating value and the corresponding aggregate score we identify a second order polynomial function enabling us to smoothly map between any value in the range [0,10] and an Agg value.

Definition 27. We define a six member tuple of coordinates called $AggCoordinates = ((10, \max_{G \in \mathsf{Pos}(S)}(G)), (8, \max_{G \in \mathsf{Pos}(S)}(G)), (7, \max_{G \in \mathsf{Ntl}(S)}(G))$ $, (5, \max_{G \in \mathsf{Ntl}(S)}(G)), (4, \max_{G \in \mathsf{Neg}(S)}(G)), (1, \min_{G \in \mathsf{Neg}(S)}(G)))$. Furthermore we stipulate that iff $\mathsf{Pos}(S)$ is a singleton set then the first member of the tuple is $((10, \max_{G \in \mathsf{Pos}(S)}(G) + \Delta\mathsf{Agg})$. Likewise iff $\mathsf{Neg}(S)$ is a singleton set then the last member of the tuple is $((1, \min_{G \in \mathsf{Pos}(S)}(G) - \Delta\mathsf{Agg})$.

Example 21. If we consider the set of arguments proposed in Table 5.2 we see that we attain the coordinates $AggCoordinates = ((10, 1.5), (8, 1), (7, 0.5), (5, 0.5),$ $(4, -1), (1, -1.5))$. Note that given that in this case $|\mathsf{Pos}(S)| = |\mathsf{Neg}(S)| = 1$ we adjusted the first and last members of the tuple. Now consider the larger example in Table 5.4. Here we attain the coordinates $AggCoordinates = ((10, 2), (8, -0.25), (7, -1), (5, -1.75), (4, -2.5),$ $(1, -3.25))$.

With the coordinates *AggCoordinates* we can then fit our polynomial function which will enable us to go from a rating to a corresponding Agg value. This is a useful feature for models that do not predict integer values and instead predict continuous values.

Definition 28. Given the coordinates *AggCoordinates* and a rating $r \in [0, 10]$ I define function $\mathsf{ratingToAgg} : r \to \mathbb{R}$ which is a second-order polynomial function fitted to the coordinates *AggCoordinates* using the least-squares method:

$$\mathsf{ratingToAgg}(r) = c_0 r^2 + c_1 r + c_2$$

Where $c_0, c_1, c_2 \in \mathbb{R}$ are the coefficients of the polynomial.

Example 22. Continuing on from the previous example, if we consider the setting in which we have 1 positive arguments and 1 negative arguments we see that the polynomial almost reduces to a linear function ($c_0 = 0.003, c_1 = 0.28, c_2 = -1.5$). Figure 5.1 shows the function. In contrast the setting in which we have 2 positive arguments and 1 negative argument produces a more curved function ($c_0 = 0.05, c_1 = -0.04, c_2 = -3.2$) and is visible in Figure 5.1.

No	Graph	Gr(G)	Grad			Attack Score	Agg	P						
			a	b	c			10	9	8	7	6	5	4,3,2,1
G_1	a $\rightarrow$ b $\leftarrow$ c	a,c	1	0	1	2	2	0.16	0.06	0.01	0	0	0	0
G_2	a b $\leftarrow$ c	a,c	0.5	0.25	1	1.25	1.25	0.13	0.11	0.04	0.01	0.01	0.01	0.01
G_3	a $\rightarrow$ b c	a,c	1	0.25	0.5	1.25	1.25	0.13	0.11	0.04	0.01	0.01	0.01	0.01
G_4	a $\rightarrow$ b $\leftrightarrow$ c	a,c	1	0.25	0.5	1.25	1.25	0.13	0.11	0.04	0.01	0.01	0.01	0.01
G_5	a $\leftrightarrow$ b $\leftarrow$ c	a,c	0.5	0.25	1	1.25	1.25	0.13	0.11	0.04	0.01	0.01	0.01	0.01
G_6	a $\rightarrow$ b $\rightarrow$ c	a,c	1	0.5	0	0.5	0.5	0.08	0.12	0.11	0.05	0.03	0.02	0.02
G_7	a $\leftarrow$ b $\leftarrow$ c	a,c	0	0.5	1	0.5	0.5	0.08	0.12	0.11	0.05	0.03	0.02	0.02
G_8	a b $\leftrightarrow$ c	a	0.5	0.5	0.5	0.5	-0.25	0.05	0.07	0.15	0.11	0.06	0.05	0.05
G_9	a $\leftrightarrow$ b c	c	0.5	0.5	0.5	0.5	-0.25	0.05	0.07	0.15	0.11	0.06	0.05	0.05
G_{10}	a b c	a,b,c	0.5	0.5	0.5	0.5	-0.25	0.05	0.07	0.15	0.11	0.06	0.05	0.05
G_{11}	a $\leftrightarrow$ b $\leftrightarrow$ c		0.5	0.5	0.5	0.5	-1	0.02	0.03	0.07	0.15	0.11	0.09	0.08
G_{12}	a b $\rightarrow$ c	a,b	0.5	0.75	0	-0.25	-1.75	0.01	0.01	0.02	0.08	0.13	0.14	0.13
G_{13}	a $\leftarrow$ b c	b,c	0	0.75	0.5	-0.25	-1.75	0.01	0.01	0.02	0.08	0.13	0.14	0.13
G_{14}	a $\leftrightarrow$ b $\rightarrow$ c		0.5	0.75	0	-0.25	-1.75	0.01	0.01	0.02	0.08	0.13	0.14	0.13
G_{15}	a $\leftarrow$ b $\leftrightarrow$ c		0	0.75	0.5	-0.25	-1.75	0.01	0.01	0.02	0.08	0.13	0.14	0.13
G_{16}	a $\leftarrow$ b $\rightarrow$ c	b	0	1	0	-1	-2.5	0	0	0	0.03	0.01	0.13	0.19

Table 5.4: Breakdown of probability distribution and aggregate graded scores for each graph in a graph with 2 positive arguments and one negative

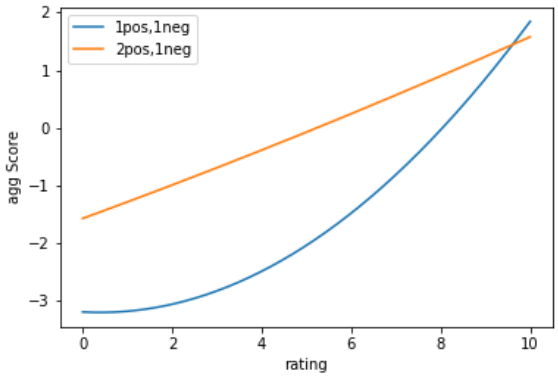


Figure 5.1: Fitted ratingToAgg functions for two sets of arguments

In order to assign a probability distribution we simply say that the mass assigned to a graph for a given rating is inversely proportional to its distance to $\mathsf{ratingToAgg}(r)$. Furthermore we take the squared distance to emphasise that graphs far away from the rating's aggregate value are least likely.

Definition 29. Given a function ratingToAgg and a rating $r \in \{1,..,10\}$ we define a distance function $\mathsf{AggDist}(G,r) = \frac{1}{1+|\mathsf{Agg}(G)-\mathsf{ratingToAgg}(r)|^2}$. I then define a probability mass function:

$$P^*(G,r) = \frac{\mathsf{AggDist}(G,r)}{\sum_{G \in S} \mathsf{AggDist}(G,r)}$$

Example 23. The right-hand side of Table 5.4 depicts the probability distribution of the ratings over the graphs. We see that given the graph space is asymmetric the highers ratings are more spread out across the positive graphs in comparison to the lower ratings which distribute almost exclusively to the lower end of the graph space.

Using Argument Impacts

In the previous section we have only looked at generating probability distributions over the constellations of argument graphs based on the rating alone. We have not considered using additional information that can be extracted from real data for which our distribution is to be used. In this section I propose a method for extracting

information about individual arguments and how they impact the overall rating from the reviews before using this information to update our probability distribution.

A set of reviews contains some information which allows us to ascertain an approximate understanding of the influence an argument has in a review when it appears. By influence I mean that across the set of reviews, if an argument appears in a review the rating of the review moves with its polarity as well.

In order to make this assessment I look at the reviews in which the argument appears in and measure how much the ratings where in line with its polarity. I define this as being the degree of impact the argument has within the set of reviews.

Definition 30. Given a set of reviews Rev and an argument a, the set of reviews the argument appears in is given by $\mathsf{App}(a,Rev) = \{rev \in Rev | rev = (A,r) \,\&\, a \in A\}$. I denote the number of reviews it appears in as $N = |\mathsf{App}(a,Rev)|$. The sum of the ratings is then $\mathsf{sum}(a,Rev) = \sum_{(a,r)\in \mathsf{App(a,Rev)}} r$. The impact of the argument is then given by:

$$\mathsf{Impact}(a,Rev) = \begin{cases} \dfrac{\mathsf{sum}(a,Rev)}{10N} & \text{if} \quad a \in A^+ \\ 1 - \dfrac{\mathsf{sum}(a,Rev)}{10N} & \text{if} \quad a \in A^- \end{cases}$$

Where the coefficient $10N$ is used as I have assumed a 0-10 scale for review ratings.

Proposition 9. Assuming a rating scale of 0-10 an argument a, and a set of reviews Rev, then $0 \leq \mathsf{Impact}(a,Rev) \leq 1$.

Proof. If we assume that $a \in A^+$ then the smallest value for $\mathsf{sum}(a,Rev)$ occurs if all the reviews have a rating of 1. In this case the $Impact(a,Rev) = 0$. If we assume $a \in A^-$ then the best value it can be assigned in all reviews is 0 in which case $Impact(a,Rev) = 0$. If we consider the contrary then we see that $a \in A^+$ and all reviews have rating of 10 then $Impact(a,Rev) = 1$ and vice versa when $a \in A^-$. □

Example 24. Take a case in which we have four reviews $Rev = \{(\{a,b,c\},9), (\{a,b,c\},8),(\{a,d\},7),(\{b,c\},2)\}$. where $A^+ = \{a,c\}$ and $A^- = \{b,d\}$. The im-

pacts are then $\mathsf{Impact}(a,Rev) = 0.8$, $\mathsf{Impact}(b,Rev) = 0.63$, $\mathsf{Impact}(c,Rev) = 0.36$ and $\mathsf{Impact}(d,Rev) = 0.3$

Given an impact we can then modify our probability distribution

Definition 31. Given a set of reviews Rev, a review $(A,r) \in Rev$, the corresponding graph space S for the review and a graph $G \in S$ the distance between the impacts of the arguments A and their grades in graph G is given by:

$$\mathsf{dist}(A,Rev,G) = \sqrt{\sum_{a \in A} (\mathsf{Impact}(a,Rev) - \mathsf{NormGrade}(G,a))^2}$$

Proposition 10. The maximum value dist can have is $\sqrt{|A|}$ and the minimum value is 0.

Proof. The lower bound for dist is given when for each argument $a \in A$ the $\mathsf{Impact}(a,Rev) - \mathsf{NormGrade}(G,a) = 0$, i.e. $\mathsf{Impact}(a,Rev) = \mathsf{NormGrade}(G,a)$. In this case the value for $\mathsf{dist} = 0$. The upper bound is given when for each $a \in A$ the following true: $\mathsf{Impact}(a,Rev) - \mathsf{NormGrade}(G,a) = 1$. In this case we see the term $\sqrt{\sum_{a \in A} (\mathsf{Impact}(a,Rev) - \mathsf{NormGrade}(G,a))^2}$ reduces to $|A|$. □

What is meant by Proposition 10 is that when an argument appears alone or in a group of arguments of similar polarity it is then the case that it will be assigned an artificially large impact score. Whilst it may be a good sign that the argument, even in a standalone context, still causes the rating to be inline with its polarity it does not necessarily mean that we can learn any argumentative information, i.e. in the case when it appears along side counter arguments.

The distance measure can then be used to skew the distribution across the graph space towards one which incorporates information of each argument's impact. There is a natural correspondence between impact and graded score as they both are indicators to the degree of importance an argument plays in a graph/review.

Definition 32. Let (A,r) *Revs* be a review, and S a graph space. Given a graph $G \in S$ we say that $d_G = \frac{1}{\mathsf{dist}(A,Rev,G))}$. The update weight associated with graph G is then.

$$\mathsf{Weight}(G,r) = \frac{d_G P(G,r)}{\sum_{F \in S} d_F P(F,r)}$$

The weight assigned to each graph is thus the product of the probability of the graph and the inverse distance of the graph's grades to the argument's impacts. The normalising constant at the bottom ensures that the distribution of weights across the graph space is a probability distribution.

The weight update is akin to Bayesian inference where we wish to incorporate some additional beliefs we have about the graphs (in this case the impacts) into our probability distribution.

Example 25. Continuing from Example 24 if we now consider a review $(a,b,c,9)$ we find that the largest weights are $\mathsf{Weight}(G_6,9) = 0.3$; this makes sense in this graph a has the highest grade followed by b and then c. We also see that $\mathsf{Weight}(G_6,3) = \mathsf{Weight}(G_6,3) = 0.10$ and that $\mathsf{Weight}(G_2,3) = \mathsf{Weight}(G_5,3) = 0.05$.

In some applications it may however be the case that we do not wish to completely switch our probability distribution across the candidate graphs to the distribution given by the weights; and instead we wish to cautiously incorporate the weights. In this case I propose a parameterised update as follows.

Whilst the impact values of each argument may be useful, there may be cases in which the dataset is non-ideal and has a degree of noise in it. In this cause we may wish to exercise caution when incorporating impact into our probability distribution; to this end I propose a parameterised update to the probability distribution which enables us to set the degree to which the weights will change the distribution.

Definition 33. Let (A,r) be a review, S a graph space, $G \in S$ a graph, $P(G,r)$ a probability mass assigned to G, $\mathsf{Weight}(G,r)$ an update for G and $\gamma \in [0,1]$ be an update parameter. The update to $P(G,r)$ is then given by:

$$P^*(G,r) = P(G,r) + \gamma(\mathsf{Weight}(G,r) - P(G,r))$$

We can see that when $\gamma = 0$ no contribution is made by the update weight and the probability remains the same. When $\gamma = 1$ we see that $P(G) = \mathsf{Weight}_r(G, r)$. The γ value thus allows us to control the contribution of the update weights to the probability distribution.

Example 26. Continuing on from Example 25 we see that the probabilities are set to their weights values, e.g: $\gamma = 1$ are $P(G_6) = \mathsf{Weight}(G_6, 9) = 0.3$. When however we set $\gamma = 0.5$ we see that $P(G_9) = \mathsf{Weight}(G_6, 9) = 0.2$ so we have updated the distribution more cautiously.

5.3 Experiment

In this section I demonstrate my argument graph learning model using a dataset of reviews taken form the Drug.com corpus. I illustrate the pipeline of steps needed to be able to integrate a standard machine learning classifier that enables one to assign graphs to reviews without ratings.

More specifically this is a two part process. In the first part I train a standard machine learning classifier to predict the rating of a review given a tuple of arguments. I demonstrate, using an annotated dataset that this is possible. The second step is then to take the predicted ratings and use this to build a probability distribution across the constellation of possible graphs and then sample from this distribution in order to assign a graph to the review. I test each section in isolation to illustrate how the pipeline can be adapted to sets of reviews with ratings.

The dataset used in this experiment is the same dataset used in the previous chapter, Chapter 5.

5.3.1 Predicting Ratings for Reviews

In the following I discuss the details of my model as well as its performance on the dataset

Model Architecture I trained a 2-layer feed-forward multi-layer neural network to predict ratings given a binary vector of arguments. My architecture consisted of 250 neurons in the hidden layers and a single neuron in the output layer. For the hidden

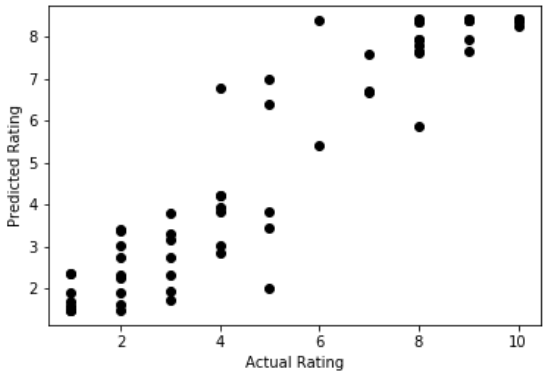


Figure 5.2: Predicted vs. Actual Ratings for trained model on validation set

layers I chose a softmax activation function whilst for the output layer I chose a linear activation function. The model was training using standard backpropogation with a mean-square-error (MSE) loss function. I used the off-the-shelf Python deep-learning library Keras.

I split my data set into training:validation split of 80:20.

Performance After 150 iterations of training the neural network I achieved a mean absolute percentage error (MAPE) of 30.86 %. MAPE is a standard loss measurement when training regression models; it is defined as:

$$\text{MAPE} = \frac{100\%}{n} \sum_{t-1}^{n} \left| \frac{A_t - F_t}{A_t} \right|$$

where A_t is the ground truth, F_t the predicted value and n the number of datapoints.

The results demonstrated that using a simple neural network the reviews tended to obey the three polarity categories. The hardest ratings to predict were the ratings between 4 and 6. This I believe is partly due to the quality of the original reviews; a number of times the annotators noted that there was no understandable correlation between the arguments proposed in a review and the rating provided.

5.3.2 Predicting Graphs for Reviews

In this section I discuss the process of assigning graphs to reviews. In order to evaluate my model I gave two annotators (neither were authors) 29 reviews, in which

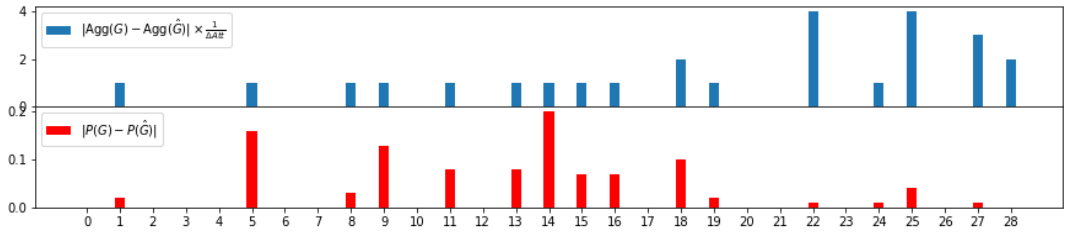


Figure 5.3: Results showing the probability distances and Agg distances between the predicted graphs and actual graphs

the arguments where identified, and asked them to assign each review an argument graph. Each annotator was given a set of guidelines in which he/she was instructed to assign attacks between the identified arguments if they felt there was a legitimate attack between them. Attacks were interpreted to mean that the attacking argument was stronger than the attacked one. If this was not immediately obvious based on the content of the arguments then they were instructed to see if the polarity of the rating coincided with the arguments they felt were more acceptable. In order to compare my predicted graphs to the ground truths provided by the annotators I constructed two measurements.

The first is grounded distance which measures the degree of overlap between my predicted graph's grounded extension and the grounded extension of the ground truth graph. Formally I define the grounded similarity as follows:

Definition 34. Given a sequence of ground truth graphs $(G_1,..,G_m)$ and corresponding predicted graphs $(\hat{G}_1,..,\hat{G}_m)$ the extension performance is given by the function:

$$\mathsf{GroundedDist} = \frac{|\mathsf{gr}(G) \setminus \mathsf{gr}(\hat{G})| + |\mathsf{gr}(\hat{G}) \setminus \mathsf{gr}(G)|}{|\mathsf{gr}(G)| + |\mathsf{gr}(\hat{G})|}$$

The function GroundedDist is 0 when the both graphs have exactly the same extension and 1 when they share no arguments in common.

The Agg distance measures the amount of error between the ground truth graph's Agg score and the predicted graph's Agg score. For a given graph space S the maximum that any two graphs $G_1, G_2 \in S$ can differ by is ΔAgg. I use this understanding to define the distance measure below.

Definition 35. Given a graph space S and set $Aggs = \{\mathsf{Agg}(G)|G \in S\}$ and a ground truth graph G and predicted graph $\hat{G}$ s.t $G, \hat{G} \in S$ I define an aggregate distance function:

$$\mathsf{AggDist}(G, \hat{G}) = \frac{|\mathsf{Agg}(G) - \mathsf{Agg}(\hat{G})|}{\mathsf{max}(Aggs) - \mathsf{min}(Aggs)}$$

Example 27. Consider the example in Table 5.4. Assume we have a ground truth graph G_1 and a predicted graph G_{10}. The denominator of $\mathsf{AggDist}(G_1, G_{10})$ is $\mathsf{max}(Aggs) - \mathsf{min}(Aggs) = 2 - -2.5 = 4.5$. The final value is then $\mathsf{AggDist}(G_1, G_{10}) = (2 - -0.25)/4.5 = 0.5$

I note if we divide the numerator of the function AggDist by ΔAgg we can treat this quantity as an equivalence class to tell us, in terms of ΔAgg how many classes separate the predicted graph from the actual graph.

In selecting a graph from the graph space I select the graph with the highest probability. In the case the have multiple graphs with the same probability I randomly sample from this set.

In terms of results Figure 5.3 shows the error in terms of aggregate scores and the probability scores. The upper graph shows how many equivalence classes in the predicted graph was away from the actual graph. What can be seen is that in most cases the predicted graphs were only 1 equivalence class.

I found that the average GroundedDist to be 0.30. I found this to be the case as most reviews were consistent in that if the rating was of a particular polarity it generally meant that the grounded extension was also of the same polarity. In the few cases where we produced the wrong extension we were either adding an additional argument to the extension or removing it; in other words the extension we provided was not far from the actual extension.

Figure 5.4 depicts a review, in which three argument types where identified, and the attacks where assigned using my probabilistic model.

What I have demonstrated in this section is the ability to take tuples of arguments with no argumentative relations annotated and use the proxy measure of the rating to assign the closest matching graph. The model can also be easily adapted

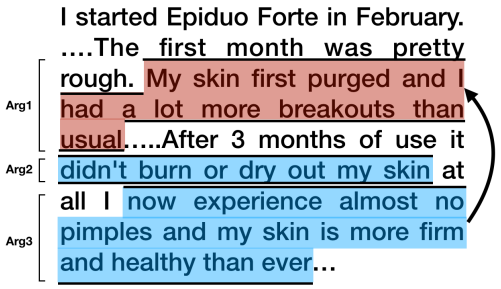


Figure 5.4: A shortened review for the acne drug Epiduo with arguments annotated. Three arguments were identified. The graph sampled from the corresponding graph space is depicted above with Arg3 attacking Arg1

to reviews which provide no ratings simply by fitting a classifier that learns the relation between tuples of arguments and ratings. The graphs assignment task is then handled by the probabilistic argumentation pipeline put forward in this paper.

5.4 Related Works

Generating probability distributions over constellations of argument graphs was proposed in [74] where it is assumed that an agent specifies a belief in the acceptability status of arguments. Using this data the paper proposed methods for aggregating, combining and summarising these beliefs. Whilst related to this paper, I have a different starting point which is that I do not have access to such beliefs directly rather I have access to ratings which I process to produce a distribution over a set of argument graphs. There have been a few proposals for argument graphs learning algorithms when in/out/un labellings are provided by agents. In [46] a learning algorithm is proposed which takes as input a probability distribution over a set of in/un/out labellings. The algorithm is an on-the-fly algorithm to aggregate these labellings into a weighted argumentation graph. In my case I deal with a setting in which I do not have access to such labellings and furthermore I produce a distribution over a constellation of argument graphs. Likewise [47] makes a similar starting assumption in that the algorithm begins which a set of labellings for each argument. A Bayesian approach is proposed in order to learn from these labellings a posterior distribution for a set of arguments being in an extension. Both of these papers differ from my approach in that I do not assume I have such labelled data

with regards to which arguments are in and out. Another proposal in [75] provides a method for extracting bipolar argument frameworks from a set of movie reviews. Each review contains a textual review and a binary rating indicating whether the reviewer thought the movie was good or bad. The proposed algorithm produces a quantitative bipolar argument per review which differs from my probabilistic output. Various proposals for capturing and aggregating views taken from the social web have also been made [61][27]. These proposals use judgement aggregation and voting mechanisms to produce the aggregation which differs from my approach which produces probabilistic interpretation of views.

In summary my proposal differs primarily from the existing literature in that it is driven by my interpretation of ratings. The notion of rating is not dealt with explicitly in the literature and certainly not in a probabilistic context.

Discussion

In this chapter I have proposed a methodology for creating a probability distribution of arguments graphs in a space of arguments graphs. I consider a situation in which I deal with bipartite argument graphs but do feel this can be generalised to handle multipartite graphs.

I further provide a method for utilising information extracted from a corpus of arguments which can be used to enrich the probability distribution. I demonstrate through my experiment that by using off-the shelf machine learning models it is possible to learn a mapping between from tuples of arguments to argument graphs.

In terms of computational costs involved in predicting the graphs I found in my experiment that the largest graph space I had to compute was for a review that contained 3 positive arguments and 3 negative arguments giving a total of 263,144 argument graphs. This is still manageable to compute. The total number of argument graphs in the graph space grows exponentially according to Proposition 1 and when the number of arguments and counter arguments grows to 4 each the size of the graph space grows to 43×10^9 which is not trivially computed. I do believe that a better understanding of the combinatorics of the various grounded extension can push my method to handle more arguments.

To conclude what I have proposed in this chapter is the use of the constellations approach to probabilistic argumentation to models the uncertainty over the set of possible graphs that can be assigned to an argument. As we will see in the next chapter, modelling it as such assists in further developing the system into a supervised learning process; a process that is much more amenable to a probabilistic approach. Having said that the functions used to compute this probability distribution rely on methods more closely related to work done on graded semantics.

In future I wish to experiment with other graded semantics to see the influence on the resulting distribution. Likewise I wish to explore the use of additional acceptability semantics in order in order to enrich the function for partitioning the graph space based on polarity. I also acknowledge that what I have presented is conceivable for smaller sets of arguments and in order to scale my techniques I will require a better understanding of the combinatorics involved in generating graphs for a particular grounded extension.

Chapter 6

Argument Reasoning using Supervised Learning

In the previous chapter we proposed a method for identifying a probability distribution over the set of possible graphs that the reviewer may have had in mind. The method is built upon the assumption that if a review has a positive rating it implies any positive argument(s) in the graph will likely be 'winning' arguments and vice versa. The method takes as input a review, which is a set of arguments and a rating, and outputs a probability distribution over a set of arguments graphs for that review. Since no external human input is used to inform or verify the resulting probability distribution, i.e. no annotator provided data was used in producing the distribution, we can describe this method as a type of unsupervised learning process in that we are able to arrive at a output without any labelled input. In Bayesian terms we can also describe the method as providing us with a prior distribution as this distribution encapsulates our starting assumption about the probability distribution for that review before having considered any real world evidence.

In this chapter we consider the question of how this 'prior' probability distribution can be updated if we begin to consider human provided data. The human provided data in this case would be a correct argument graph for a review. Our proposal is the use of Baye's rule to update the prior probability distribution for a review using annotator provided data. In this chapter we describe this proposal and conduct a number of simulations to verify the framework works.

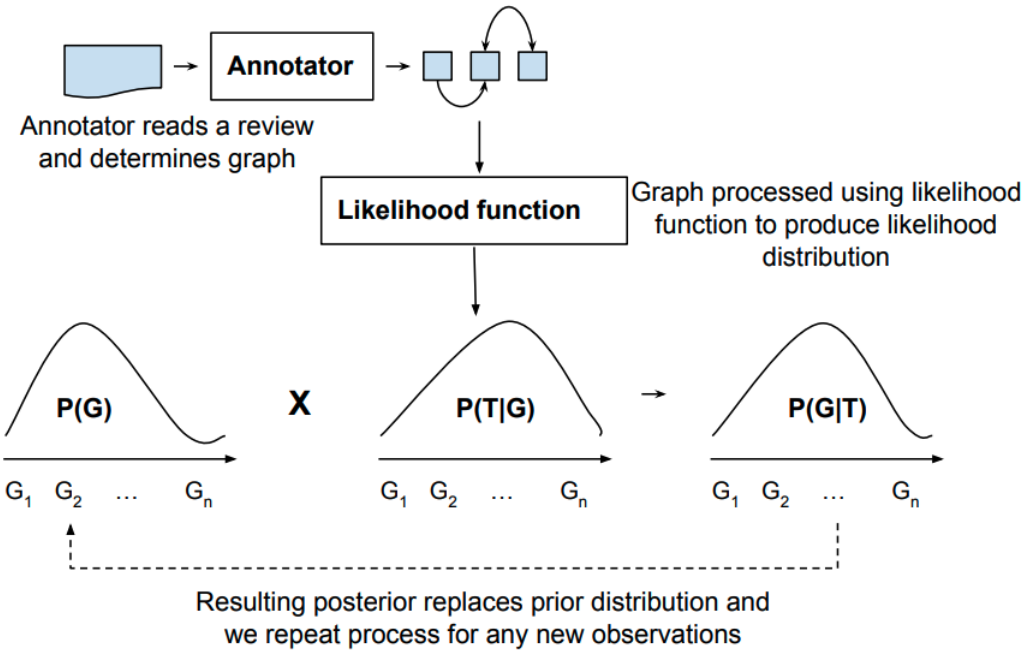


Figure 6.1: Bayesian framework for updating probability distribution when new labelled data is collected.

Whilst in the the previous chapter we described the learning process as an unsupervised learning process we in this chapter we consider our proposal as a supervised learning process since we are refining our 'model' through real world data.

6.1 Bayesian Framework for Updating Probability Distribution

In this section we describe our supervised learning process at a high level. The data used as part of this process is assumed to be provided by annotators and/or can even be crowd-sourced. In particular what the annotator is to provide is an argument graph for a review. The graph they provide captures the annotator's interpretation of the conflicts between the various arguments in the review. Using Bayesian terminology we refer to this annotation as an observation. Formally we say that an observation is the assignment of a graph to a review.

Definition 36. Let $v = (A, b)$ be a view where $A \subseteq A^+ \cup A^-$. An **observation** is a

tuple $t = (G, b)$ where $G \in \mathsf{Space}(A^+, A^-)$.

Example 28. Continuing on with the arguments in Figure 6.2 and rating . Consider a view $(\{a,b,c\},9)$. An example observation would then be $((\{a,b,c\},\{(a,b)\}),9)$

If multiple observations are submitted we then have a set of observations, T, for a set of reviews. We recall that in the previous chapter that we proposed an unsupervised method for determining the prior distribution for a review, i.e. $P(G)$. If we consider this prior in conjunction with the set of observations we thus have all of the necessary components to be able to update our prior using these observations and thus produce a new (posterior) distribution that incorporates these new observations using Baye's rule.

Definition 37. Given a set of positive and negative arguments A^+ and A^-, a view $v = (A, b)$ s.t $A \subseteq A^+ \cup A^-$, a graph space $S = \mathsf{Space}(A^+, A^-)$ and a set of of observations T we say that our updated belief is given by the function:

$$P(G|T) = \frac{P(T|G)P(G)}{\sum_{G \in S} P(T|G)P(G)} \tag{6.1}$$

In the formula above the variable $P(G)$ is our prior distribution and represents our initial belief that graph $G \in S$ is the intended graph for view v before we have considered the observations. We acquire our prior distribution using the methods proposed in the previous chapter.

The term $P(T|G)$ is referred to as the **likelihood function** and tells us, if we assume the intended graph for v is G, what the probability (likelihood) of observing the observations T is. The multiple of the prior and the likelihood, $P(T|G)P(G)$, is proportional to our new 'updated' belief about what the correct graph should be for a review, $P(G|T)$, and is referred to as the **posterior distribution**.

The denominator is called the normalising constant and ensures the resultant posterior distribution is a probability distribution; in other words that $\sum_{G \in S} P(G|T) = 1$.

The process of using this formula is depicted in Figure 6.1. What can be seen is that the process is run each time a new observation is made. In each 'iteration' of this process we incorporate the observation into our existing belief (prior) via the likelihood function to produce our new (posterior) distribution. When a new observation is made we then proceed to take the previously computed posterior as our starting prior. This 'loop' allows us to continually incorporate new observations when they are available.

The very first time the process is run we use the unsupervised method in the previous section to produce our starting prior distribution. As more and more observations are added this distribution will change to reflect the new labelled data. The key component that needs to be defined here is therefore the likelihood function which we do in the next section.

6.2 Likelihood Functions

In this section we define the likelihood function that will allow us to update our prior distribution using labelled data and allow us to obtain our updated posterior distribution $P(G|T)$.

The likelihood function enables us to rank/score argument graphs based on their ability to explain the labelled data/observations provided. Intuitively the likelihood function should assign higher probabilities to graphs that are in some sense 'similar' to the graph provided in the observation and assign low probabilities to graphs that are by the same standard 'dissimilar' to the graph provided. This ensures that, as per Equation 37, we will ultimately assign a higher posterior probability, $P(G|T)$, when the likelihood is high for that graph and vice versa.

The key requirements here is to thus formalise the relationship between observation and likelihood distribution. This requires that we understand the types of observations that we can expect an annotator to provide and use this understanding in turn to define the likelihood function.

We recall that an observation is the assignment of a graph by an annotator to a review. Let us take an example in which we have two reviews, v_1 and v_2 and that we

are trying to identify a probability distribution, $P(G)$, for v_1. Assume an annotator provides a observation for v_2. A question arises then of whether it is possible to use the provided observation for v_2 to update our probability distribution of v_1.

Example 29. Assume we have the argument graph shown in Example 6.2 for a review about an electric bike; let us refer to this observation as v_1. There is now a separate review v_2 which contain arguments a, b and a new negative argument, d, labelled as *'product requires costly on-going maintenance'*. We want to identify a probability distribution for the possible argument graphs for v_2. Is it possible to use the portion of the argument graph for v_1 containing arguments a and b to help us identify our distribution $P(G)$ for review v_2?

We propose that if there is some overlap between the arguments and attacks in v_1 and v_2 then it follows that there might be some information that we can harness to inform our probability distribution for v_1. To determine how to harness this information we first establish the various ways in which this overlap may occur.

In the first instance let us assume that there are no overlapping arguments between the two reviews; in other words the two reviews have completely different sets of arguments. In this case there is no information to be learnt from the observation and hence we expect that our prior distribution to remain unchanged.

In the second instance let us consider that there is an overlap of exactly 1 argument. The presence of a standalone argument, without any attacks, does not inform us of anything useful except that the argument occurred. No information can thus be used to update the prior therefore, as in the first case, we expect our prior to remain unchanged.

Similarly let us consider a third case where there is an overlap of more than 1 argument but that the overlap is such that the overlapping arguments are all of the same polarity. For example consider Figure 6.2 and let us assume we have an observation from another review which contains arguments a, c and a third negative argument d. Since we are trying to model attacks between arguments using our probability distribution and there are no attacks that can occur in this new observation that will help us how to model attacks in Figure 6.2 we can learn nothing from

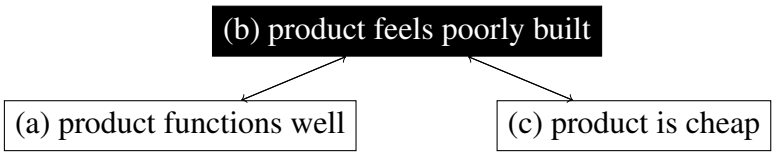


Figure 6.2: An example of an argument graph containing two positive arguments in favour of a product (a,c) and one negative argument against it (b)

this observation. This means as in the first two cases we expect our prior to remain unchanged.

What this leaves us with is the conclusion that the number of overlapping arguments needs to be greater than or equal to 2 and needs to contain at least one positive argument and one negative argument. If this is the case we can use the attacks (or absence of attacks) between the positive and negative arguments in the observation to be able to update and inform us which graphs are more likely in our likelihood distribution. We refer to such observations as **relevant**. Formally an observation $t = ((A,R),b)$ is relevant to a view $v = (A_v,b_v)$ iff $|\{A \cap A_v\}| \geq 2$. In the rest of this chapter we only consider relevant observations and so we use the terms observation and relevant observation interchangeably.

If we now consider only relevant observations we find that relevant observations can be of two types:

- $A = A_v$ (**Matching Observation**)
- $A \neq A_v$ (**Similar Observation**)

A matching observation is a relevant observation that has the exact same set of arguments as the review we are producing a probability distribution for. What this implies is that the graph provided in the matching observation can be found in the graph space for the graph we are trying to model. Since we have already defined a similarity measure for graphs in the same graph space in the previous chapter, AggDist, we can therefore use this similarity measure to help us define our likelihood function for when we encounter a matching observation.

A similar observation on the other hand is a relevant observation that is not a matching observation. In other words which there are some additional arguments

in either the observation or the view. In this case the view and the observation do not share the same graph space and a slightly modified approach is required to determine the likelihood distribution for the view.

Example 30. Consider a view $(\{a,b,c\},10)$ and two observations $t_1 = ((\{a,b,c\},\{\}),10)$ and $t_2 = ((\{a,b\},\{\}),10)$. In this case t_1 is a matching observation and t_2 is a similar observation.

In encountering both a matching and similar observation we would expect the update to the probability distribution to adhere to certain rationality postulates which we discuss below. We propose the following for a view v with likelihood function $P(T|G)$ and an observation (G,b). For ease of notation we use $\mathsf{Args}(G)$ to denote the set of arguments in the graph G.

(RPR) Rating Proportionality: Given two observations $t_1 = (G,b_1)$ and $t_2 = (G,b_2)$ and a view $(\mathsf{Args}(G),b)$ s.t $|b-b_1| > |b-b_2|$ then it should follow that $\sum_{G\in S}(P(G|t_1)-P(G))^2 > \sum_{G\in S}(P(G|t_2)-P(G))^2$. RPR states that for two observations that have the same graph but different ratings, the one that has the rating most similar to the rating of the view will impact the posterior most.

Example 31. Consider two observations o_1 with argument graph $a \leftarrow b$ and rating of $o_1 = 8$ and o_2 with argument graph $a \leftarrow b$ and rating of $o_2 = 5$. Let us assume we are trying to identify the probability distribution for a review with arguments a,b and rating 8. In this case we make the assumption that the graph in o_1 should be inform our distribution more than the graph in o_2 since its rating is more similar to our review.

(APR) Argument proportionality: Given two observations $t_1 = (G_1,b_1)$ and $t_2 = (G_2,b_2)$ s.t $|\mathsf{Args}(G_1)\cap\mathsf{Args}(G)| \geq |\mathsf{Args}(G_2)\cap\mathsf{Args}(G)|$ and $b_1 = b_2$ it should follows that $\sum_{G\in S}(P(G|t_1)-P(G))^2 \geq \sum_{G\in S}(P(G|t_2)-P(G))^2$. APR states that given two relevant observations, the degree to which the observation alter the posterior distribution is proportional to the degree of overlap in arguments with the view.

Example 32. For example consider a view with arguments $A^+ = \{a,b,c\}$ and $A^- = \{d\}$. We would expect in this case to learn more from observation which have arguments $\{a,b,d\}$ than from observations which have arguments $\{a,d\}$ only. This is because there is a greater overlap in both arguments and attacks, and these can be used to update the posterior.

In what follows we define our likelihood function for both a matching observation and a similar observation.

6.2.1 Updates with Matching Observations

Recall that the set of graphs that can be assigned to a view $v = (A_v, b_v)$ is $\mathsf{Space}(A^+ \cap A_v, A^- \cap A_v)$. For a matching observation $t = (G, b)$ we find that the graph G in the observation is a member of this graph space, i.e. $G \in \mathsf{Space}(A^+ \cap A_v, A^- \cap A_v)$. What this means is that we can make use of the aggregate score defined in the previous section when determining the likelihood distribution.

In essence what we would like from this function is to assign nearly all of its mass on graphs in which the observed graph and graph in the graph space are similar and very little mass when the opposite is true.

Definition 38. Given a view $v = (A_v, b_v)$ and a matching observation $t = (G_t, r_t)$ we define the relative **attack distance** between a graph $G \in S$ and observed graph G_t as $\mathsf{dist}_{\mathsf{Att}}(G, G_t) = 1$ iff $G_t = G$ and $\mathsf{dist}_{\mathsf{Att}}(G, G_t) = \frac{1}{2+|\mathsf{Agg}(G) - \mathsf{Agg}(G_t)|}$ otherwise. We then say that the normalised distance is $\mathsf{normdist}_{\mathsf{Att}}(G, G_t) = \frac{\mathsf{dist}_{\mathsf{Att}}(G,G_t)}{\sum_{F \in S} \mathsf{dist}_{\mathsf{Att}}(F,G_t)}$.

What can be seen is that when the graphs are the same, i.e. observed graph is same as a graph in the graph space we are assigning the largest mass possible, 1. In other cases I the mass drops off and this process is controlled by the denominator in $\mathsf{dist}_{\mathsf{Att}}$. The constant 2 allows us to minimise mass assigned when graphs are dissimilar.

Example 33. Consider a matching observation with argument graph $a \to b$. If we are trying to compute similarity of observation to a review with arguments a, b then the similarity measures we get are seen below

G_i	G	$\mathsf{dist}_{\mathsf{Att}}(G, a \to b)$	$\mathsf{normdist}_{\mathsf{Att}}(G, a \to b)$
G_1	$a \to b$	1	0.52
G_2	$a \leftrightarrow b$	0.33	0.17
G_3	$a \quad b$	0.33	0.17
G_4	$a \leftarrow b$	0.25	0.13

What we can see in Example 33 that by using the $dist_{Att}$ to measure similarity between the observation and the graphs in the graph space we are able to place the majority of the mass of the distribution onto the graph in the graph space that matches the observation. The normalisation step that is computed using $\mathsf{norm}_{\mathsf{dist}}$ ensures that the resulting distribution is a probability distribution.

We now consider how the rating of an observation influences the resulting likelihood distribution. As described in the previous subsection the basic idea is that observations that have ratings similar to the view should produce a likelihood distribution that influences the posterior more than those that are less similar. To illustrate consider a view $(A, 10)$ and two observations $t_1 = (G, 9)$ and $t_2 = (G, 1)$. In this case we would give greater priority to t_1 as the observation is very similar to the view's rating. What is thus required is a likelihood function which controls how much an observation can contribute to a probability distribution. The function should operate such that the contribution the observation makes to the resulting posterior distribution should be proportional to how similar its rating is to the review. Hence when the ratings are exactly the same the contribution is maximal and when they are maximally different the contributions are minimal/negligible (e.g. using a rating scale 0-10 if we have an observation with rating 1 and we are calculating probability distribution for review with rating 10).

To start we need a parameter which allows us to measure similarity between the observation's rating and the rating of the the review we are modelling. For this we propose the following:

$$\Delta b = \frac{b^+_{max} - b^-_{min} - |b - b_t|}{b^+_{max} - b^-_{min}} \tag{6.2}$$

Where b_{max}, b_{min} are the highest and lowest possible rating a review can receive

respectively, b is the rating of the review we are modelling and b_t is the rating of the observation.

Example 34. Consider $b_{max} = 10, b_{min} = 0$ and a review with a rating $b = 8$ for which we are trying to identify a probability distribution for. Some examples of Δb are when $b_t = 1$ we find $\Delta b = 0.22$, $b_t = 8$ we find $\Delta b = 1$, $b_t = 10$ we find $\Delta b = 0.77$ and $b_t = 6$ we find $\Delta b = 0.77$. We see that that the closer b_t is to b the more it tends towards 1 and the further it is the more it tends to 0.

Next what we need is a distribution that we can use in conjunction with $Deltab$ to control how much the posterior distribution is influenced/changed by the observation. We know that the uniform distribution is effectively a non-informative prior. This is to say that when a uniform distribution is multiplied with a distribution and then normalised the resulting distribution is exactly the same as the starting distribution; in other words $P(G|T) = P(G)$. This **uninformative distribution** we know be the uniform distribution which we define as having values equal to $\frac{1}{|S|}$ where $|S|$ is the number of graphs in the graph space.

We propose using the uniform distribution alongside $normdist_{Att}$ to control how much an observation can influence the posterior distribution based on its rating. What we desire is a resulting likelihood distribution that behaves proportionally to Δb. Consequently we expect that when ratings are maximally different, i.e. $\Delta b = 0$, the likelihood distribution reduces to a uniform distribution and the resulting posterior is equal to the prior distribution. Likewise when $\Delta b = 1$ we expect the likelihood distribution to reduce to the distribution provided by $normdist_{Att}$. To achieve this we propose the following formulae.

$$P(T|G) = \frac{1}{|S|} - \Delta b(\frac{1}{|S|} - \mathsf{normdist}_{Att}(G, G_t)) \tag{6.3}$$

What we can see is that the parameter Δb controls to what degree the likelihood distribution resembles the uniform distribution or the distribution provided by the function $\mathsf{normdist}_{Att}$. We see that when $\Delta b = 1$, i.e. the view and the observations have the same rating, the likelihood function reduces to the distribution provided

G_i	G	$P(T\|G, b_t = 0)$	$P(T\|G, b_t = 5)$	$P(T\|G, b_t = 10)$
G_1	$a \rightarrow b$	0.25	0.39	0.52
G_2	$a \leftrightarrow b$	0.25	0.212	0.17
G_3	$a \quad b$	0.25	0.212	0.17
G_4	$a \leftarrow b$	0.25	0.190	0.13

Table 6.1: Resulting likelihood distributions for three observations with identical graphs, $(a \rightarrow b)$, but different ratings.

by $\mathsf{normdist}_{Att}$. Conversely when $\Delta b = 0$ the distribution reduces to the uniform distribution over the graph space, S, for the review we are modelling. The likelihood distribution exhibits linear behaviour between these two extremes.

Example 35. Continuing on with Example 33 let us now assume we have three observations that have an identical graph $a \rightarrow b$ but the first has a rating of 0, the second 5 and the third 10. Let us also assume we are modelling a review with arguments a, b and a rating of 10. If we compute our likelihood distribution using Equation 6.3 we see the results below in Table 6.1. We can see that when $b_t = 10$, i.e. $\Delta b = 1$, the resulting likelihood function remains the same as is calculated in the table in Example 33. Conversely when $b_t = 0$, i.e. $\Delta b = 0$ the resulting likelihood function is a uniform distribution. When the rating is 5 however we see that the likelihood distribution resembles a flattened version of the the distribution provided by $\mathsf{norm}_{\mathsf{dist}}$.

Proposition 11. The function we have defined for $P(T|G)$ in Equation 6.3 satisfies RPR.

Proof. Let $t_1 = (G, b_1)$ and $t_2 = (G, b_2)$ be observations and $(\mathsf{Args}(G), b))$ a view s.t $|b_1 - r| > |b_2 - b|$. In both cases $\mathsf{normdist}_{Att}(G, G_t)$ is a constant term and the only differing term in the likelihood would be Δb. When Δb is high, i.e t_1, then the likelihood tends to $\frac{1}{|S|}$ which in turn means the posterior tends to $P(G)$ and thus $\sum_{G \in S}(P(G|t_2) - P(G))^2 > \sum_{G \in S}(P(G|t_1) - P(G))^2$ □

Once the likelihood distribution we can then update our prior distribution using it using the update procedure depicted in Figure 6.1. A full example has been pro-

G_i	$P(G)$	$P(T\|G)$	$P(G\|T)$
G_1	0.504	0.522	0.762
G_2	0.199	0.174	0.100
G_3	0.199	0.174	0.100
G_4	0.098	0.130	0.037

(a) Iteration 1

G_i	$P(G)$	$P(T\|G)$	$P(G\|T)$
G_1	0.762	0.522	0.909
G_2	0.100	0.174	0.040
G_3	0.100	0.174	0.040
G_4	0.037	0.130	0.011

(b) Iteration 2

G_i	$P(G)$	$P(T\|G)$	$P(G\|T)$
G_1	0.909	0.522	0.979
G_2	0.040	0.174	0.014
G_3	0.040	0.174	0.014
G_4	0.011	0.132	0.003

(c) Iteration 3

G_i	$P(G)$	$P(T\|G)$	$P(G\|T)$
G_1	0.979	0.522	0.990
G_2	0.014	0.174	0.005
G_3	0.014	0.174	0.005
G_4	0.003	0.132	0.001

(d) Iteration 4

Figure 6.3: Four consecutive updates to the prior distribution of a review $\{(a,b),10\}$ using matching observations. We update the distribution with four identical observations of value $(a \to b, 10)$

.

vided in Figure 6.3 showing how the posterior distribution changes as observation are incorporated into the distribution.

Example 36. Assume we are trying to identify the distribution for a review $\{(a,b),10\}$. To illustrate the update process we chose a simple scenario where we have been provided with four identical matching observations which have a value $(a \to b, 10)$. In other words the matching observations have exactly the same rating as the review we are modelling. The starting prior distribution is given to us using the unsupervised methods described in the previous chapter. Figure 6.3 shows how after each iteration the most likely graph, $a \to b$, increases in probability mass and in fact after four iterations the resulting distribution places almost all of its mass on this observed graph. What we can see is that the G_1 is converging to 1 due to the fact that the data is repeatedly producing observations with G_1 as the graph.

We note that the likelihood function for matching observations trivially satisfies APR as all observations using this likelihood function always have the same set of arguments as the view in consideration.

6.2.2 Updates with Similar Observations

We recall that a similar observation is a relevant observation that does not have the exact same set of arguments as the view we are modelling. In this case we are concerned with learning from the portion of arguments and attacks in the observation that overlap with the arguments in the view. In order to make this comparison we propose a similarity measure that compares the overlapping portion of the observed graph with the graphs in the graph space for the view based on topological structure as well as similarities in grounded extension. Measuring topological structure ensures that we are able to identify similarly structured graphs and measuring the similarity in grounded extension allows us to see how similar the conclusions are that both graphs reach. Further to this we enforce an additional measure which ensures that observations that have more arguments in common with the view's graph space contribute more to the update than observations that do not.

We begin by defining a similarity measure based on topological structure. We start by assessing the similarity between the four possible types of relations (**attack types**) between two arguments. For ease of notation we refer to the four possible attack types between any two arguments a,b given a set of attacks R as follows: we say $\mathsf{attackType}((a,b),R) = a \rightarrow b$ when $(a,b) \in R \,\&\, (b,a) \notin R$, $\mathsf{attackType}((a,b),R) = a \leftarrow b$ when $(b,a) \in R \,\&\, (a,b) \notin R$, $\mathsf{attackType}((a,b),R) = a \leftrightarrow b$ when $(a,b) \in R \,\&\, (b,a) \in R$, $\mathsf{attackType}((a,b),R) = a - b$ when $(a,b) \notin R \,\&\, (b,a) \notin R$. Where there is no confusion we also say that given an $attack \in \{a \leftarrow b, a \rightarrow b, a \leftrightarrow b, a - b\}$ and a graph $G = (A,R)$, $attack \in G$ iff $\mathsf{attackType}((a,b),R) = attack$. With these attack types we build a similarity measure to be able to measure the degree of similarity between two graphs in terms of their attacks.

Definition 39. Let a,b be two arguments and let $X,Y \in \{a \leftarrow b, a \rightarrow b, a \leftrightarrow b, a - b\}$ be two attack types for a,b. Let $|\mathsf{arcs}(X)|$ represent the number of attacks in X. We define $\mathsf{dist}_{\mathsf{att}}(X,Y)$ as:

$$\mathsf{dist}_{\mathsf{att}}(X,Y) = \begin{cases} 0 & \text{if } X = Y \\ 1 & \text{if } |\mathsf{arcs}(X)| \neq |\mathsf{arcs}(Y)| \\ 2 & \text{if } |\mathsf{arcs}(X)| = |\mathsf{arcs}(Y)| \ \ \& \ \ X \neq Y \end{cases}$$

The measure captures the degree of dissimilarity between the attack types. We note that this measure is the same as the Hamming distance except in the case of $\mathsf{dist}_{\mathsf{att}}(a \leftrightarrow b, a - b)$. The reason we give special treatment to these attack types is because we want to ensure that they are treated dissimilar to $\mathsf{dist}_{\mathsf{att}}(a \leftrightarrow b, a \leftarrow b)$ or $\mathsf{dist}_{\mathsf{att}}(a \leftrightarrow b, a \rightarrow b)$. This is because they have the same grounded extension and so in this sense are more similar. A tabulated version of the measure can be seen in Table 6.2.

	$a \rightarrow b$	$a \leftrightarrow b$	$a - b$	$a \leftarrow b$
$a \rightarrow b$	0	1	1	2
$a \leftrightarrow b$	1	0	1	1
$a - b$	1	1	0	1
$a \leftarrow b$	2	1	1	0

Table 6.2: Table with attack distances between the different attack types

To compute the overall similarity between two graphs we simply add the similarity scores for each pair of overlapping arguments as follows:

Definition 40. Given graphs $G_1 = (A_1, R_1)$ and $G_2 = (A_2, R_2)$ the shared positive arguments are $A^+_{1,2} = A_1 \cap A_2 \cap A^+$ and the shared negative arguments are $A^-_{1,2} = A_1 \cap A_2 \cap A^-$. We define a set of possible conflicts as $C = A^+_{1,2} \times A^-_{1,2}$. For a given possible conflict $c \in C$ we define a distance $\mathsf{diff}(c, R_1, R_2) = 2 - \mathsf{dist}_{\mathsf{att}}(\mathsf{attackType}(\mathsf{c}, \mathsf{R}_1), \mathsf{attackType}(\mathsf{c}, \mathsf{R}_2))$. We say that the degree of similarity between both graphs is given by the function $\mathsf{similarity}(G_1, G_2) = \sum_{c \in C} \mathsf{diff}(c, R_1, R_2)$.

Example 37. Consider two graphs $G_1 = a \rightarrow b$, $G_2 = a \leftrightarrow b \rightarrow c$. The set of possible conflicts is $\{(a,b)\}$. From G_1 we find the $\mathsf{attackType}((a,b), \{(a,b)\}) =$

$a \rightarrow b$ and from G_2 we find $\mathsf{attackType}((a,b),\{(a,b),(b,a),(b,c)\}) = a \leftrightarrow b$. The $\mathsf{similarity}(G_1,G_2) = 1$.

Next we consider the degree to which two graphs are related in terms of their grounded extensions.

Definition 41. Given two graphs $G_1 = (A_1,R_1)$ and $G_2 = (A_2,R_2)$ the degree to which they are related in terms of their grounded extension is defined by the proportion of overlapping arguments in their grounded extensions. We say the overlap in the grounded extensions is $\mathsf{dist}_{\mathsf{gr}}(G_1,G_2) = |(\mathsf{gr}(G_1) \setminus \mathsf{gr}(G_2)) \cup (\mathsf{gr}(G_2) \setminus \mathsf{gr}(G_1))|$ and that $\mathsf{dist}_{graph}(G_1,G_2) = \mathsf{similarity}(G_1,G_2) + \mathsf{dist}_{\mathsf{gr}}(G_1,G_2)$. We then define the total distance between graphs as:

$$\mathsf{dist}_{\mathsf{total}}(G_1,G_2) = \frac{\mathsf{dist}_{graph}(G_1,G_2)}{\sum_{G \in S} \mathsf{dist}_{graph}(G,G_2)}$$

What we can see is that the final distance measure, $\mathsf{dist}_{\mathsf{total}}$ is a combined measure taking into consideration similarity in terms of topology of overlapping graphs and similarity of grounded extensions.

Example 38. Consider two graphs $G_1 = a \rightarrow b$ and $G_2 = a \rightarrow b \rightarrow c$. We find that in this case $\mathsf{similarity}(G_1,G_2) = 1$, $\mathsf{dist}_{\mathsf{gr}}(G_1,G_2) = 0.5$. If we now consider the graph space $S = Space(\{a,c\},\{b\})$, s.t. $G_2 \in S$, that $\mathsf{dist}_{\mathsf{total}} = 0.095$.

We finally want to ensure that the difference in rating between view and observation influences the final likelihood distribution (as in the case of the matching observation). We also add an additional constraint which is that we want to make sure that an observation which has a higher degree of overlapping arguments with the graphs space contribute more to the update than those that have a lower degree.

Definition 42. Given an view (A,r) and a similar observation $t = (G_t,r_t)$. We say that $\Delta Args = \frac{|A \cap \mathsf{Args}(G)|}{\mathsf{max}(|A|,|\mathsf{Args}(G)|)}$. We then say that the likelihood function is this given by:

$$P(T|G) = \frac{1}{|S|} - \Delta Args \Delta b(\frac{1}{|S|} - \mathsf{dist}_{\mathsf{total}}(G,G_t)) \tag{6.4}$$

G_i	$P(G)$	$P(T\|G)$	$P(G\|T)$
G_1	0.504	0.221	0.457
G_2	0.199	0.359	0.293
G_3	0.199	0.198	0.162
G_4	0.098	0.221	0.089
Observation = $a \leftrightarrow b \rightarrow c$			

(a) Iteration 1

G_i	$P(G)$	$P(T\|G)$	$P(G\|T)$
G_1	0.457	0.254	0.442
G_2	0.293	0.310	0.347
G_3	0.162	0.225	0.139
G_4	0.089	0.211	0.072
Observation = $a \leftrightarrow b \leftarrow c$			

(b) Iteration 2

G_i	$P(G)$	$P(T\|G)$	$P(G\|T)$
G_1	0.442	0.154	0.251
G_2	0.347	0.462	0.590
G_3	0.139	0.231	0.118
G_4	0.072	0.154	0.041
Observation = $a \leftrightarrow b$			

(c) Iteration 3

G_i	$P(G)$	$P(T\|G)$	$P(G\|T)$
G_1	0.251	0.25	0.251
G_2	0.590	0.25	0.590
G_3	0.118	0.25	0.118
G_4	0.041	0.25	0.041
Observation = $b \rightarrow c$			

(d) Iteration 4

Figure 6.4: Four consecutive updates to the prior distribution of a review $\{(a,b),10\}$ using a mixture of matching, similar and non-relevant observations. In iterations 1 and 2 we see that We update the distribution with four identical observations which in this example we assume have a value of $(a \rightarrow b, 10)$

Example 39. Let a,b be positive arguments, c be a negative argument, $v = (\{a,c\},10)$ a view and $t = ((\{a,b,c\},(a,c)),10)$. In this case $\Delta Args = 2/3$, $\Delta b = 1$.

Proposition 12. The function $P(T|G)$ defined in Equation 6.4 satisfies APR.

Proof. Let $t_1 = (G_1,b)$, $t_2 = (G_2,b)$ be observations where $G_1 = (A_1,R_1)$ and $G_2 = (A_2,R_2)$ s.t $G_1 \sqsubseteq G_2$ and let $v = (A,b')$ be a view. Δb is a constant and $\Delta Args$ is larger for t_1 than for t_2. Therefore $\sum_{G \in S} P(t_1|G)$ tends to $\frac{1}{|S|}$ as $\Delta Args$ decreases and likewise $\sum_{G \in S} P(t_1|G)$ tends to $\mathsf{dist}_{total}(G_1,G)$ as $\Delta Args$ grows. □

We note that our proposal can be adapted to settings in which ratings are not used, i.e where a view is simply a set of arguments and observations are argument graphs. In this case we can specify the uniform distribution as the prior distribution for the view and adapt our proposals for the likelihood function by making $\Delta r = 1$ as by doing so we remove the influence of the rating. Thus for a matching observation we would have a likelihood of $P(T|G) = \mathsf{normdist}_{Att}(G,G_t)$ and for a similar observation we would have $P(T|G) = \frac{1}{|S|} - \Delta Args(\frac{1}{|S|} - \mathsf{dist}_{total}(G,G_t))$.

Example 40. Using the same example as in Example 35 let us assume we are updating the prior distribution using four observations as depicted in Figure 6.4. In the

first and second iteration we see that the resulting likelihood places most of it mass on graph $a \rightarrow b$ and this is reflected in the resulting posterior distribution. In iteration 3 we see that since the observation is a matching observation the likelihood distribution is much less flatter and the mass assigned to graph $a \rightarrow b$ increases greatly in this step. The final iteration uses a non-relevant graph and so a uniform/non-informative distribution is used and this does not influence the resulting posterior.

Simulations

In the absence of labelled data for our reviews we use synthetic data to test our proposal. Our primary motivation in these simulations is to demonstrate that our approach produces reasonable posterior probability distributions that reflect the synthetic data being used to produce them.

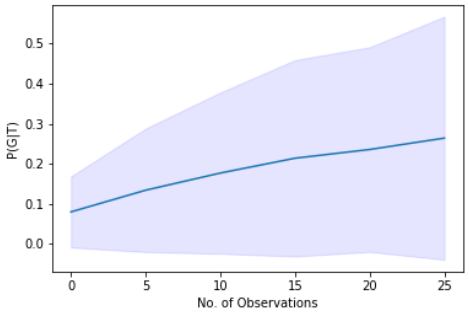
One simple expectation that we would have is that if the set of observations we have contain the same graph many times, e.g. $a \leftarrow b$ appearing in 90% of the observations, we would expect that the resulting posterior increasingly accumulates mass on that particular graph. In order to formally be able to measure this expectation we define the concept of noise relative to an observed graph.

$$\mathsf{noise}(G,T) = 1 - \frac{|\{t \in T | G \sqsubseteq t\}|}{|T|}$$

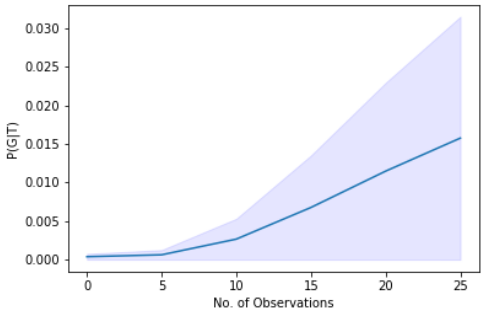
Where for two graph $G = (A,R)$ and $t = (A_t,R_t)$, $G \sqsubseteq t$ iff $A \subseteq A_t$ and $R = \{(a,b) \in R_t | a,b \in A\}$.

Example 41. Given a set of observations $T = \{t_1,t_2\}$ where and $t_1 = (a \rightarrow b, 10)$ and $t_2 = (a \rightarrow b \rightarrow c, 10)$. We see that $t_1 \sqsubseteq t_1$ and $t_1 \sqsubseteq t_2$ since the attack $a \rightarrow b$ exists in both observations. Consequently we find that $\mathsf{noise}(t_1, \{t_1,t_2\}) = 0$.

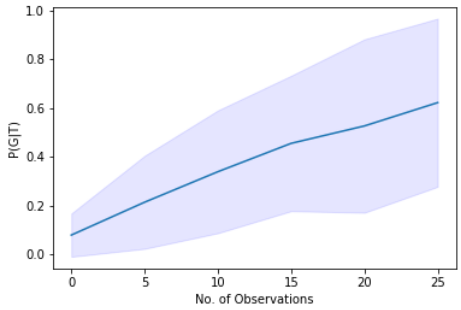
Our expectations is that the probability mass assigned to graph $G \in T$ in the posterior distribution should be inversely proportional $\mathsf{noise}(G,T)$. Moreover we expect that the mass assigned to the graph should increase as the number of observations increase, i.e. $|T|$.



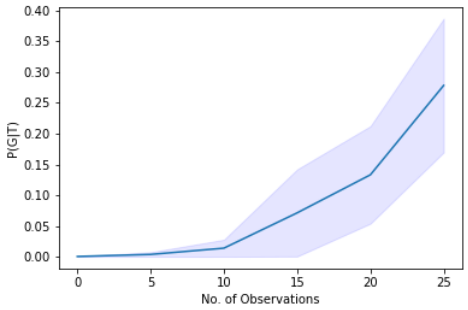
(a) noise = 0.75, no. of args = 3



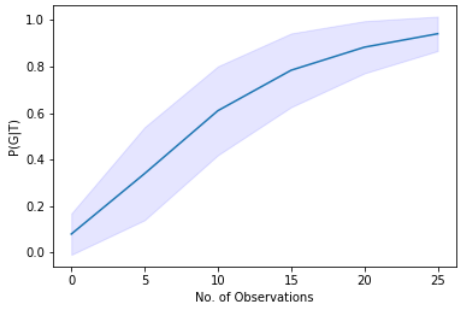
(b) noise = 0.75, no. of args = 5



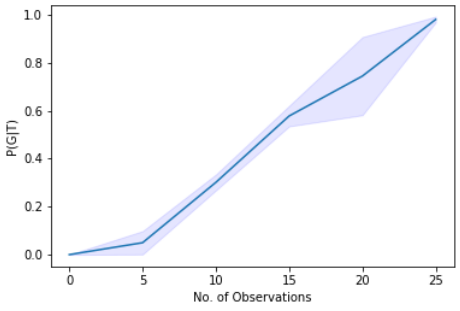
(c) noise = 0.5, no. of args = 3



(d) noise = 0.5, no. of args = 5



(e) noise = 0.25, no. of args = 3



(f) noise = 0.25, no. of args = 5

Figure 6.5: Plots showing $P(G|T)$ against number of observations. The central blue lines Figures (a), (b), (c) show the average posterior probabilities for all $G \in \mathsf{Space}(\{a,b\},\{c\})$ and (d), (e), (f) show them for all $G \in \mathsf{Space}(\{a,b,c\},\{d,e\})$. The blue bands represent the 95% confidence intervals.

6.2.3 Synthetic Data

To observe how the posterior probability of a review's graph G changes w.r.t noise and number of observations we created multiple datasets for different combinations of noise and dataset size. We started by specifying a set of positive arguments $\{a,b,c\}$ and negative arguments $\{d,e\}$. We experimented with 3 levels of noise $(0.25,0.5,0.75)$ and various numbers of observations in the range $5 \leq N \leq 25$. For each possible combination of noise and N we then created 150 synthetic datasets $(T_1,..,T_{150})$ for G where each graph in the set is randomly generated until we achieved the required level of noise and N. To simplify the simulations we the fixed the rating of every observation to 10. We then summarised the posterior probability for that combination of noise and N by computing an average posterior probability using all of the datasets i.e. $P(G|T) = \sum_{i=1}^{150} P(G|T_i)/150$.

6.2.4 Results

Figures 6.5a, 6.5c and 6.5e show how the posterior probability of all graphs in $\mathsf{Space}(\{a,b\},\{d\})$ change with different levels of noise and dataset size. Instead of plotting the posterior probability for each individual graph in the graph space we plot the average of all the graphs in that graph space along with the 95% confidence interval bands denoted by the surrounding blue bands. Figures 6.5b, 6.5d and 6.5f show a similar picture but for a larger graph space of $\mathsf{Space}(\{a,b,c\},\{d,e\})$. Note that for all of the figures when the number of observation is 0 we report the prior probability for the graph.

As can be seen in all cases there is a general trend for the model to assign a higher posterior probability as more observations are added. Furthermore as noise decreases this learning is much faster. As can be seen clearly in Figures 6.5a, 6.5c, a larger starting prior noticeably influences the rate at which the posterior probability grows when noise is high. This is also noticeable in Figure 6.5b even though the graphs start with a prior probabilities that are all very small.

Figure 6.6 shows how using observations with different ratings affects the posterior probability. The purple lines show the average posterior probability for graphs in $\mathsf{Space}(\{a,b\},\{d\})$ when the rating for the view is 10. Each of the three lines use

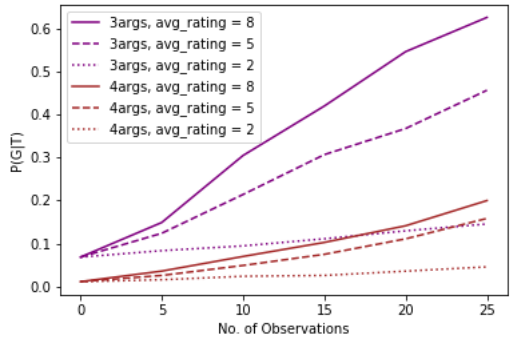


Figure 6.6: Simulations showing the effect of ratings on the posterior probability

observations with average ratings of 2, 5 and 8 respectively. The red lines show a similar picture but for graphs in $\mathsf{Space}(\{a,b\},\{d,e\})$. What can be seen is that observations that use ratings very different to that of the view cause the posterior probability to grow slower than when they are similar; this demonstrates the effect of Δb on the posterior. For more detailed analysis of these simulations please refer to our supplementary results and analysis [1].

6.2.5 Conclusion

In this chapter I have proposed a method for updating our (prior) probability distribution for a review when labelled data becomes available, and by labelled data we mean a review for which some annotator has provided what they believe to be the correct argument graph for. The method uses an iterative process based on Bayesian inference to update the distribution through the use of custom likelihood functions.

I identified two types of labelled graphs (observations) that we might encounter. The first is a matching graph which is a graph which is in the graph space for the review that is being modelled. In this case I propose using a modified version of the graph similarity measures, proposed in the previous chapter, in order to construct our likelihood distribution. The second type of labelled graph is a graph which has a portion of its graph overlapping with graphs in the graphs space for the review we are modelling. In this case I propose a similarity measure that takes into consideration graph topology and number of overlapping arguments. In both cases I

[1] https://github.com/robienoor/bayesianframeworklearningargumentation

also account for how close the rating from the labelled graph is similar to the review we are trying to model.

Evaluating the approach and seeing how well it performs requires large amounts annotated data (each review needs to be assigned a corresponding graph by a human annotator). Since this was not feasible I chose to generate synthetic data and use this as part of a number of simulations. These simulations focused on ensuring the proposal met certain expectations. In particular one expectation was that if, for a given graph, we repeatedly update the graph with the same graph/graphs that are very structurally similar that those graphs that are similar to this observation, over the course of the simulation, acquire all the probability mass.

Chapter 7

Argument Interface

In the previous chapters I developed proposals for how to implement the first two pipes in the pipeline depicted in Figure 1.1. The final pipe is an interface through which the end user can interact with the data mined from the reviews. In this chapter I consider the following:

1. What are the computational requirements for generating and storing probability distributions for a set of reviews?

2. When storing the extracted argument data (arguments + attacks) in a database what would the data model look like?

3. What are the requirements for having a user interface interacting with this extracted data?

The objective of this chapter was to understand what the requirements would be for developing a system based on the proposals made in the previous 3 chapters. In particular the chapter is focused on understanding what the technological requirements for developing a system through which a patient can interact with the extracted argument data. We believe that the system will have to be a "full-stack application" in the sense that we will need both a database that enables us to store and update the arguments and argument graphs produced through the argument pipeline as well as an interface through which the user can interact with this data.

We will start by considering the challenges for generating and storing the probability distributions that we presented in the previous two chapters. We saw in Chap-

ter 5 how for each review we can algorithmically generate the probability distribution over the graph space for that review. Generating the probability distribution for each graph can understandably be computationally demanding. The computational requirements are proportional to the number of arguments in the review as well as the polarities of the arguments. We consider how the computational requirements can be reduced in Section 7.1.

In addition when we introduce labelled (argument graph) data, we then consider the computational costs of producing and storing our posterior distributions. We consider this n Section 7.2 as well.

Once we have produced a method for generating and storing the probability distribution for the graph spaces for the reviews we then have to consider how a system will read from this dataset. What we propose in Section 7.3 is that the end user will not have to sample from this distribution and instead we propose a sampling in advance an intermediate database that will store a single argument graph per review.

Our proposal also contains suggestions on how the data model for this intermediate database should be. Lastly we consider the types of user interactions with the dataset and think about ways in which the data can be explored and interacted with. We consider two types of visualisation. The first is developing a simple ability to view the arguments in the graph as well as the attack relations between them. We propose that these depicted entities should be interactive and the user should have the ability to label them as correct/incorrect. What this would enable the system to do is to collect feedback on how the underlying models have performed; both the argument extraction pipe as well as the probabilistic argumentation. We also briefly consider the emerging field to text2sql models which are models that are capable of interpreting 'natural language queries' into 'structured queries', e.g. 'how many people reported no side effects' $\rightarrow$ select count(*) from arguments where argument.type like "no side effects".

7.1 Storing Probability Distributions from Unsupervised Learning

Recall that the given a review (A, r) the size of the graph space is given by the formulae blah. Using this formula we can calculate the total number of argument graphs that we will be identified for this dataset using the formula:

Definition 43. Assume we have a dataset $\{(R,A,r),..,(R,A,r)\}$ where R is the textual review, A is the set of arguments identified in the review and r is the numerical rating for the review. Let us also assume in this example that the superset of all positive and negative arguments is $\mathscr{A}^+$ and $\mathscr{A}^-$ respectively. The total numer of argument graphs for the dataset is then $\sum_{i=1}^{t} 2^{mn}$.

As we can see as the term mn increases there is an exponential growth in the size of the graph space. Likewise as the size of the dataset grows the total number of graphs that is having to be stored grows too as well. In the case where we have no labelled data however we can exploit the fact that the way in which the probability distribution for a graph space is identified is in fact determined only by three variables; namely the number of positive arguments m, the number of negative arguments n and the rating of the review. What follows from this is then that for a given dataset we only need to compute the graphs for the unique set of m,n,r combinations and the unique set of probability distributions

Proposition 13. Given a set of reviews $\mathscr{R} = \{(R_1,A_1,r_1),..,(R_n,A_n,r_n)\}$ the total number of unique graph spaces for $\mathscr{R}$ is $|\{A|\{A_1,..,A_n\}\}|$ and the total number of unique probability distributions is $|\{(A,r)|\{(A_1,r_1),..,(A_n,r_n)\}\}|$

Proof. Any two sets of arguments, A_1,A_2 yield the same graph space if $A_1 = A_2$ hence if we have collection of reviews the number of unique graph spaces is equal to the number of unique sets of arguments in those reviews, e.g. $|\{A|\{A_1,..,A_n\}\}|$. Similarly, if we assume we are using an unsupervised learning approach, the number of unique probability distributions is $|\{(A,r)|\{(A_1,r_1),..,(A_n,r_n)\}\}|$. □

Example 42. Consider reviews $Rev_1 = (R,(\{A_1,A_2\},4))$, $Rev_2 = (R,(\{A_1,A_2\},4))$ and $Rev_3 = (R,(\{A_1,A_2\},10))$ where $\mathscr{A}^+ = \{A_1\}$ and $\mathscr{A}^- = \{A_2\}$. What we can

see is that since all three reviews share the same arguments they all yield the same graph space. Furthermore since $Rev_1 = Rev_2$ they also yield the same (unsupervised) probability distribution.

What this implies is that a computer will not have to hold in memory $\sum_{i=1}^{t} 2^{mn}$ argument graphs and $\sum_{i=1}^{t} 2^{mn}$ associated probabilities, i.e. the graph space and probability distribution for each individual review in the dataset. This is because in fact a lot of the graph spaces will be duplicates in the dataset.

We can further reduce the number of graphs that we have to hold in memory by abstracting away from (combinations of) individual arguments and focusing on their polarities. Reviews that have the same combination of positive and negative arguments may not have the exact same graph space but they are otherwise structurally identical. What I propose is that we only generate graph spaces for unique combinations of positive and negative arguments (focusing only on polarity). This graph space will serve as the **graph space template** for any review that has the same combination of positive and negative arguments. Thus when the graph space for a particular review is required by the program we can instantiate it on-the-fly by using this 'cached' graph space template.

With this in mind we therefore need only to worry about generating graph spaces for the set of unique combinations of positive and negative arguments in the dataset; i.e. we only need to generate all the necessary graph space templates.

Proposition 14. The number of unique graph space templates $S_{\mathscr{R}}$ for a set of review $\mathscr{R}$ where a review is $(R,(A^+,A^-),r)$ is between $0 \leq S_{\mathscr{R}} \leq |\{((|A^+|,|A^-|)|(R,(A^+,A^-),r) \in \mathscr{R}\}|$.

Proof. The number of unique graph spaces when we consider polarity is given by given by $|\{((|A^+|,|A^-|)|(R,(A^+,A^-),r) \in \mathscr{R}\}|$. This therefore becomes the upper bound for the number of number of unique graph spaces we have to store in memory. □

This upper bound here is given when each review in the dataset has a unique combination of positive and negative arguments. A similar line of reasoning hold

when considering how much memory would be required to hold the associated probability distributions for each review. We will only generate the probability distributions for unique combination of graph spaces templates and ratings; this we refer to **probability distribution template**. Thus, as in the previous case, at run time when we require the probability distribution for a review we only need to lookup the distribution using the combination of rating and positive and negative arguments.

In Figure 7.2 we see a diagram summarising the sequence of operations in generating a probability distribution for a single review. The first step is to generate the graph space template which either generated from scratch or taken from a cache (if that graph space template was previously requested). The probability distribution over this is then generated. If we are using unsupervised learning then it is possible that there is already a cached version of the distribution (if the input arguments and rating are the same of previously requested distribution).

Proposition 15. The number of unique probability distributions templates $P_{\mathscr{R}}$ for a set of reviews $\mathscr{R}$ where a review is $(Rev,(A^+,A^-),r)$ is between to $0 \leq P_{\mathscr{R}} \leq |\{(|(A^+,A^-,r)|(Rev,(A^+,A^-),r) \in \mathscr{R}\}|$.

Proof. The unsupervised learning process produces one unique probability distribution for a given combination of positive arguments, negative arguments and a rating, i.e. $(|A^+|,|A^-|,r)$, based on the Definition 29 which specifies that the final distribution for a review $(|A^+|,|A^-|,r)$ is given by $\frac{\mathsf{AggDist}(G,r)}{\sum_{G \in S} \mathsf{AggDist}(G,r)}$. Therefore, the total number of probability distributions templates the system will have to store is based on the number of positive and negative arguments in the set of reviews, $\mathscr{R}$. If there are no reviews in the set of reviews, $\mathscr{R}$ then the total number of templates is 0. If every review in the set of reviews is different in its combination of positive and negative arguments then this is the upper bound and we will have to store a graph for each review.

□

We now discuss briefly how much physical memory is potentially saved during run time by templates for the graph space and probability distributions. Both

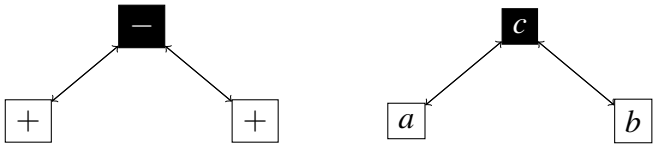


Figure 7.1: The graph on the left is a template for a graph in the graph space for reviews with 2 positive arguments and one negative arguments. The graph on the right is a set of arguments using the template. Templates can be used to generate argument graphs and graph spaces on the fly

arguments and attacks in an argument graph can be denoted by an integer value. In Python this occupies 4bytes of data. Secondly the individual probabilities in the probability distributions can be represented as Float values. Float value in most applications occupy 8 bytes of data. In the example below we describe how much memory is saved by employing templates.

Example 43. Continuing on from Example 42 we find that we have to compute 1 graph space template. This template will be for graphs with 1 positive argument and 1 negative argument. We will also only have to store two probability distributions. The first will be for reviews with 1 positive, 1 negative argument and a rating of 4. reviews with 1 positive, 1 negative argument and a rating of 10. The total amount of data for probability distributions is thus 64 bytes. The amount of data for the graph space is 12bytes.

In the example above we see that the total memory requirement for representing the review is 78 bytes. This is in contrast to 164bytes that would otherwise be necessary had we stored an individual graph space and associated probability distribution. In summary in this section we have demonstrated that through careful implementation of how argument graphs are generated and stored we are able to lower the computational resource requirements for running this in a real world system.

7.2 Storing Probability Distributions from Supervised Learning

In the previous section we focused on how templates for both the graph space and probability distributions could be used in order to improve the memory requirements of the system. This is assuming the dataset we were working with does not contain any labelled data; i.e. observations as per the definitions in Chapter 6. In this section we describe the computational cost that comes when labelled data is available; more precisely the memory requirements of having to store the posterior probability distributions.

Going back to our discussion of templates we noted that the determining factor for the types of graph space templates that we generate is the number of positive and negative arguments in reviews in the dataset. Since the only change that occurs when we use a supervised learning approach when using labelled data is the resulting probability distribution we can continue to use the graphs space templates to store the graphs.

If we recall the supervised learning method it involves computing a similarity score by comparing the (labelled) graph against all of graphs in the review. This process means that for each review we end up with a slightly different distribution depending on the rating and the individual arguments. This means that the probability distribution templates no longer work once labelled data is introduced and the lower bound essentially changes to $|\{(|(A^+,A^-,r)||(Rev,(A^+,A^-),r) \in \mathscr{R}\}|$.

7.3 Data Model For arguments and Counter Arguments

In order to store the arguments and the argument graphs we require a basic data model. In the previous section we explored how the underlying argumentation system can be implemented. We spoke about deconstructing the task of building the probability distributions for the graph space for each review into smaller and more manageable processes. For an illustrated sequence of operations please refer to the

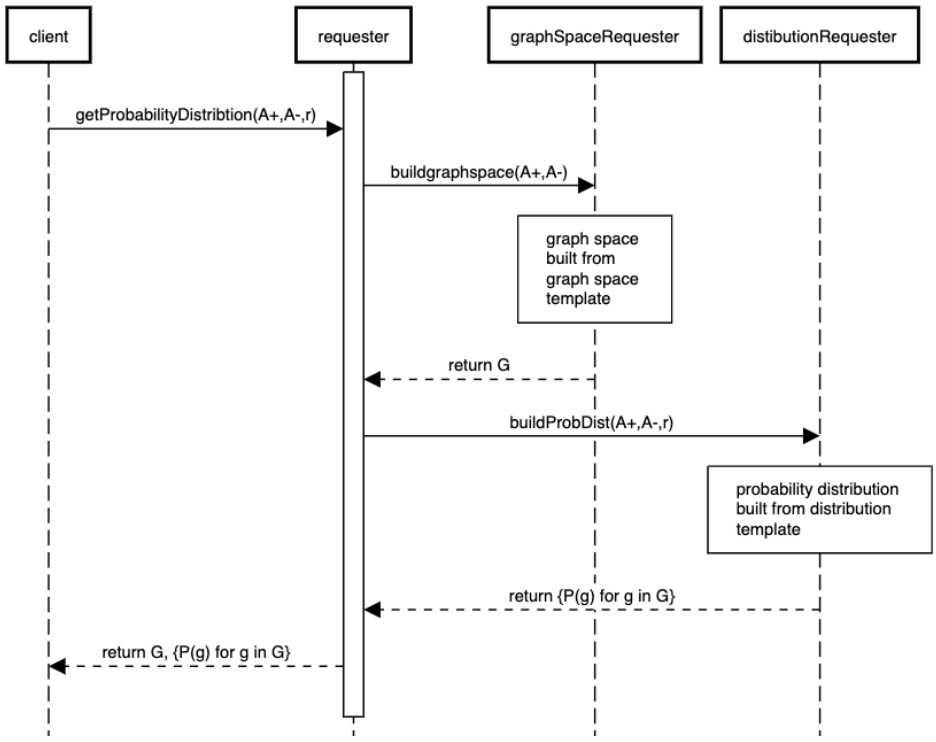


Figure 7.2: Sequence of operations for generating a probability distribution for a review

appendix Figure A.1.

So far our discussion has focused on how best to compute and store the graph spaces and associated probability distributions. Ultimately however our objective is to have a system where we can retrieve argument graphs for each review but also handle queries such as:

- *"In how many reviews has argument a attacked argument b?"*
- *"Show me other examples of where argument a has been countered by argument b"*
- *"On average how many times does argument a defeat argument b"*

For example consider two arguments for acne medication: A - *"drug was inexpensive"* and B - *"drug had undesirable side effects"*. A patient may want to know how often in the reviews has A been more dominant of an argument than B or on average which argument is stronger etc.

To be able to handle such queries we need a data model that can store individual

No	Graph	$P(G)$	Bin
G_1	$a \rightarrow b$	0.25	$0 \leq P(G) < 0.25$
G_2	$a \leftrightarrow b$	0.25	$0.25 \leq P(G) < 0.5$
G_3	$a \quad b$	0.5	$0.5 \leq P(G) < 0.9$
G_4	$a \leftarrow b$	0.1	$0.9 \leq P(G) \leq 1$

Table 7.1: Example of probability bins

attacks between arguments. To do this we propose that, for each review, we identify the probability distribution for the review's graph space. We then predict a graph for the review by sampling from this distribution. We lastly store the details of that graph in a relational database whose schema can be seen in Figure 7.7. If we repeat this procedure for all reviews our relational database can then be used to answer queries such as those mentioned previously.

In order to sample from the probability distribution we essentially wish to take the graph with a high probability. A simple way to do this would be to take the graph that has the highest probability assigned to it, i.e: $\mathsf{max}(P(G)|G \in Space(Review))$.

Another approach would be to sample from the probability distribution as you would from a multinomial distribution. In this approach we allocate each graph in the graph space a bin whose size corresponds to the probability assigned to it. We then produce a random number in range [0,1] and find the first bin whose probability is greater than or equal to the random number. An example of how these bins can be computed is given below.

Example 44. Consider the graphs in the Figure 7.1. If we generate a random number between 0 and 1, say $x = 0.5$ we see that the first bin whose probability is larger than x is G_3 therefore we would sample graph G_3. If $x = 0.8$ then we also sample G_3 again. If $x = 0.4$ then we sample G_2 and so on.

We can see that using this sampling approach that the probability that a graph is selected is directly proportional to how much probability mass it has been assigned. The larger the probability mass the more likely that the random number will fall within the boundaries of its bin and vice versa. Although this approach requires a few more steps (in comparison to just taking the graph with the largest probability),

Figure 7.3: Data model for reviews, arguments and attacks

it does ensure that other graphs in the graph space are not completely ignored and also have a chance to be sampled.

Once a sampling approach has been decided and a graph has been selected for a review we can then store this graph review in our relational database using the schema in Figure 7.7. In this schema the review is stored in a table, the corresponding arguments in a separate table and the attacks in their own tables.

Once the database has been populated with all of the argument graphs one can then proceed to query the database. As an example consider the example query below which has been translated into the corresponding (structured query language) SQL syntax:

question: *What are all of the counter arguments to argument A?*

query:

```
select label from
(select
argument.id as argument_id,
argument.argument_label as argument_label,
attack.id as attack_id,
attack.argument_id,
attack.counter_argument_id
from argument
full join attack on argument.id = attack.attacked_id
where argument.label = 'A'
and attack.attacker_id IS NOT NULL) as attackers
```

```
inner join
argument on argument.id = attackers.attacker_id
```

One can also answer questions such as *'what is the most popular attack on A'* or *'total number of attacks on A'* using the appropriate SQL command such as max, count and avg commands.

Whilst having the ability to make these queries is valuable it is only helpful to users who can construct such queries. In order to make this functionality available to non-technical users (which are mostly patients who will interact with this database) then it we propose using a automated translation service through which the user can enter their query in natural language and the service will convert it into an executable SQL query.

7.4 Natural Language Queries

In recent years there has been an interest in developing algorithms that can translate natural language queries into SQL. Over the past few years a number of machine learning approaches have been proposed for this 'text2sql' task in which the algorithm/model is fed a natural language query such as 'how many people are over 85?' and the model then outputs a corresponding SQL query such as 'select count(*) from table where age $>$ 85' for a targeted database.

This area of research is particularly exciting as it enables individuals to query databases without requiring them to construct the queries themselves. In particular this is appealing to applications whose user-base are incapable of writing such queries. In our case we do not expect that uses will be able to read/write SQL but they may wish to interact with the argument database through natural language. Whilst this is not the focus of the thesis I spent some time exploring the feasibility of fine tuning a pretrained text2sql models to work with our database.

One model in particular that has acquired prominence in recent years is the T5 model [76]. The T5 model is a transformer based model that has been pretrained on a large corpora comprising both unsupervised NLP tasks (masked-token prediction). In terms of fine-tuning this base T5 model then T5 is trained using pairs of 'free text

k	Acc_ex	Acc_lf
1	0.94	0.89
2	0.77	0.77
3	0.88	0.88
4	1	1
5	0.94	0.94
Avg	**0.91**	**0.90**

Table 7.2: Results from 5-fold validation on text2sql dataset

queries' and 'SQL'.

To generate the training data we focus on a simple set of queries that the user may want to ask the database. These queries make use of the SELECT statement and the COUNT statement. In essence we want the user to be able to ask in free text the following queries:

- How many times is arg a attacked by arg b?
- In which reviews is arg a attacking b?

Whilst we acknowledge that the user may want to write free text queries that make use of other aggregate operators such as TOP 10, MAX or AVG due to limited resourcing we did not have scope produce labelled data for these types of queries.

To generate training we programatically produced 100 SQL queries and then for each query a human annotator (myself) provided 3 corresponding natural language queries. The SQL queries were generated by using a template into which we randomly inserted the aggregate type (select, count(*)) and inserted the argument and counter argument. By doing this we generate a dataset of 300 pairs of SQL to natural language queries.

This training data was then split into an 80:20 training split and was used to train the T5 model. The main parameters that we changed was the number of epochs for which we were training for and we found that 5 epochs were sufficient for model training. Since our dataset was so small we could not produce a validation training set to monitor performance during training. We used a 5-fold validation here which means that we 80:20 training:test sample was performed 5 times.

To measure performance for the model we chose to use a performance metric that is commonly used to benchmark text2SQL performances. We report two forms of accuracy. The first is execution accuracy, Acc_{ex}, which is defined as:

$$Acc_{ex} = \frac{N_{ex}}{N}$$

Where N is defined as the number of queries and N_{ex} is defined as the number of queries that can be executed and return the correct rows for that query. For example 'show all examples of where argument C attacks argument B' would be mapped to 'SELECT * FROM attackview where attacker = C and attacked = B'. This query can be directly executed using the schema we have define without any adjustment and will return the correct rows for our query.

The second metric is logical form accuracy, N_{LF}, and is defined as:

$$Acc_{lf} = \frac{N_{lf}}{N}$$

Where N_{lf} is defined as the number of queries in which the predicted sequence matches the target sequence exactly, i.e. the strings are an exact match. Acc_{lf} is desirable to report as Acc_{ex} can sometimes reward predictions where the output query unintentionally returns the correct result, i.e. the query is executable SQL and the rows returned are those desired. For example 'show all examples of where argument C attacks argument B' could be mapped to 'SELECT * FROM attackview where attacker = C LIMIT 1'. In this example output query it might be the case that the first result returned by adding the constraint 'LIMIT 1' will have an attacked = 'B'. The N_{lf} score therefore penalises such results but also penalises results where the predicted query is correct but does not exactly match the target string. For example SELECT * FROM attackview where attacked = B and attacker = C'. In this example the attacker and attacked conditions have been switched but are identical.

The results are shown in Table 7.2. We can see that both Acc_{lf} and Acc_{ex} remain mostly similar throughout each of the 5 experiments. This suggests that any error caused is purely because the returned rows were incorrect. We also see that

text query	translated sql
show reviews where 'cheap medication' attacks poor 'side effects'	select * from table where attacker = 'cheap medication' and attacked ='side effects'
in which reviews is 'side effects' the attacker	select * from table where attacker ='side effects'
number of times where 'no side effects' is attacked by 'minimal benefits'	select count(*) from table where attacker ='minimal benefits' and attacked = 'no side effects'

Table 7.3: Examples of free text queries translated to SQL

the average accuracies reported around or above 0.9 which suggests that there may be room for improvement with the addition of more training data. Some examples of generated SQL queries can be seen in Table 7.3.

7.5 Interface

In this section we explore options for the interface through which the end-to-end argumentation system can be interacted with. In particular in this section we propose designs for how the interface can be used to assist in developing the first two pipes in the pipeline; that is allowing the user to identify arguments and secondly allowing the user to identify attacks.

This type of functionality would allow the user to update and train the underlying models, both for argument detection and graph prediction in real time. This setup would accelerate the learning process of the models particularly if multiple users were to be using the system.

In terms of visualising the predicted graphs to the user we felt it was important that any visualisation did not necessarily interfere with the normal experience and instead the mined information was optionally visible to the user. We suggested that this could be implemented by first highlighting the spans of text which have been identified as an argument. To differentiate between positive and negative arguments we use blue to denote positive and red for negative. In order to visualise the attacks

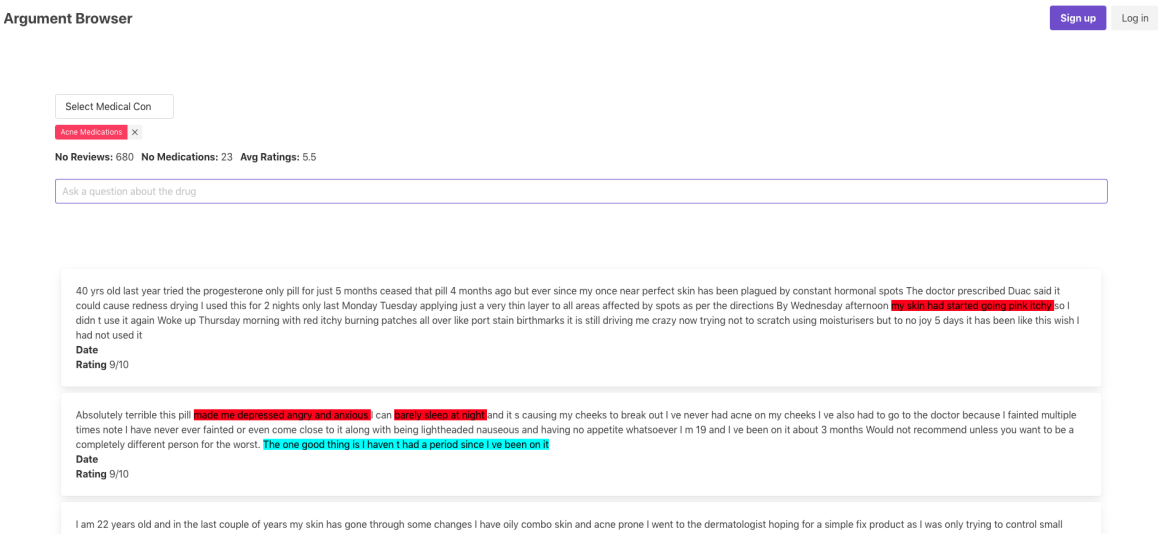


Figure 7.4: Argument Browser. Pro arguments highlighted in blue and counter arguments in red

the user can hover his/her cursor over the argument and an arrow(s) will be shown between the arguments in conflict. In this way the user can experience the review as they normally would but additionally they are able to see easily which of the arguments exist and which ones are in conflict.

As mentioned earlier on one of the stated objectives the system was that it would assist the user in quickly navigating all of the important arguments in that set of reviews. To do this the user should have the option of quickly navigating through the arguments and counter arguments that have been recorded in the system. To achieve this we offer the user various options when hovering over an individual arguments:

- View other reviews containing this type of arguments
- View other counter arguments to this argument
- View basic statistics about this argument, e.g. how many time this argument has appeared in the dataset etc.

We felt that by offering these options only upon hovering over the text the user has an unobstructed view of the text and can therefore read as they normally would and additionally engage with the argumentation system when they wanted.

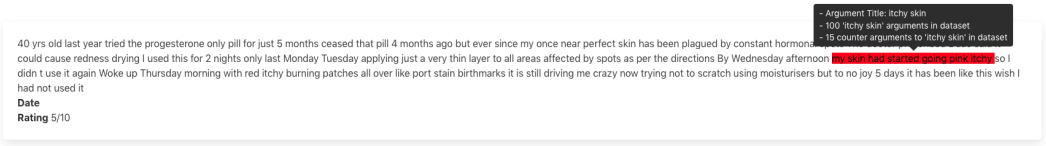


Figure 7.5: Hovering over argument reveals information about the argument and show statistics about counter arguments

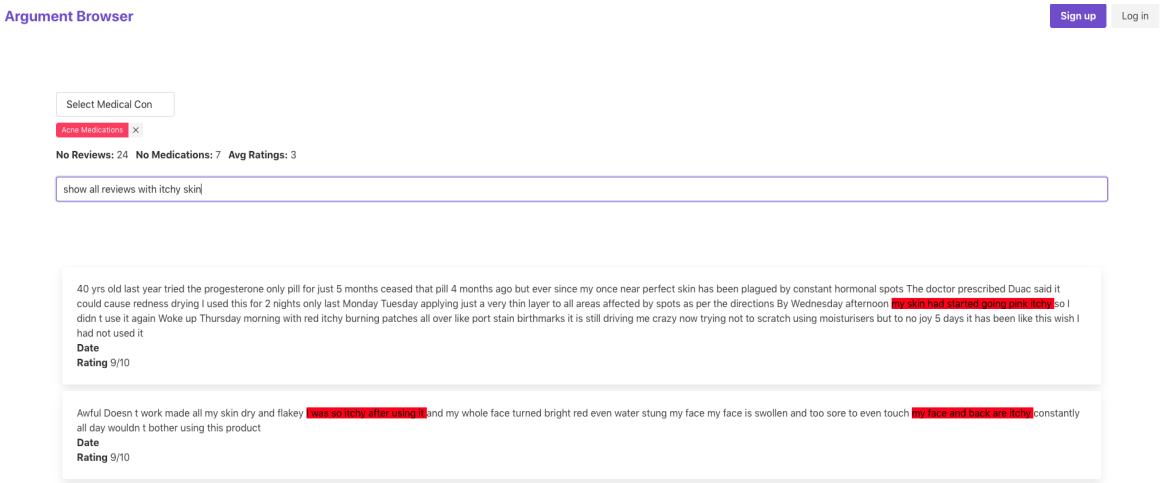


Figure 7.6: Searching via text2Sql would allow users to retrieve other reviews with similar arguments

Given our limited success with text2Sql models we also propose the addition of a search bar at the top of the page in which the user can query the argument database. The user can type in their query and an appropriate response will be displayed in the interface. We identified that there are essentially two categories of response. There are those in which the SQL query is simply a select statement, i.e 'select * from counterarguments etc' in which case the user is requesting to see a list of reviews. The second category is those in which the user is requesting the user of some type of SQL aggregate function i.e ' select count(*) from' etc. In this case the user is expecting a single numerical answer.

In the case where the user has requested a select statement then we simply return a view of all of the reviews returned by executing the SQL statement. In the case where the user has requested the use of some sort of aggregation operator we return an additional window at the top of the interface in which we present the numerical response, we also additionally return all of the reviews upon which the

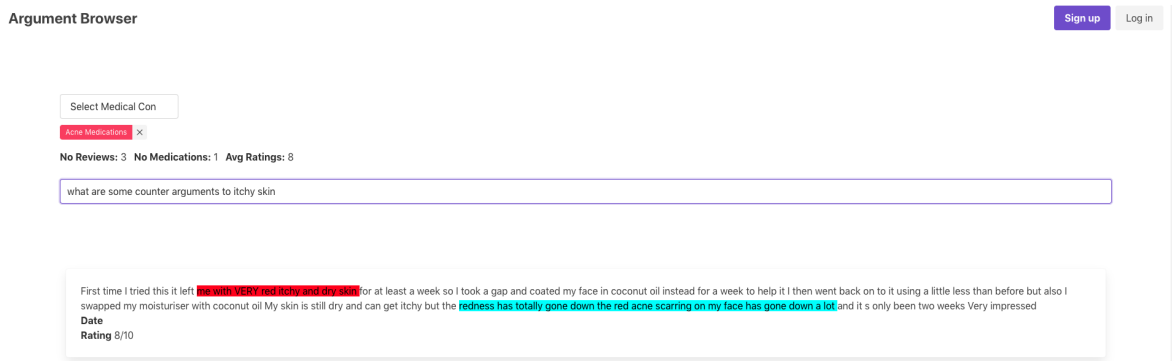


Figure 7.7: Searching via text2Sql would allow users to retrieve other reviews with counter arguments

aggregate query was run.

7.6 Conclusion

In this chapter I have discussed the requirements to build a full-stack application through which a patient can interact with the reviews and the 'structured' argument data extracted for a set of reviews. I have considered the computational requirements needed store and produce probability distributions over the reviews. In order to reduce the amount of memory required to store the distributions for each review we propose the use of argument graph space templates and probabilistic graph templates. In the case of argument graph space templates these ensure that, where two or more reviews have the same combination of positive and negative arguments we do not need to store in memory separate graph spaces for the reviews. In the case of probability distribution templates where two or more review have the same set of arguments and the same rating, and no labelled data has been considered (unsupervised setting) we do not need to store in memory separate probability distributions for these reviews since they will be identical.

Following this I consider how the user will interact with the system. I note that since this data will be stored in a relations database an appropriate method is needed to ensure non-technical users (those unfamiliar with SQL syntax) are able to query the data. I propose a text2SQL model for tackling this problem which is a machine learning model capable of taking a free-text human readable query and converting it into a corresponding SQL query that can be executed against the

database containing the structured argument data. The text-2-SQL model means that user's can interact with the data on the system with queries such as 'what are counter arguments to this argument' and expect a visual response.

Finally I provide some suggestions of how the interface will look. I describe functionality that I believe is key to the interface such as viewing arguments in the review and the ability to see counterargument in the review as well as potential counter arguments in other reviews (by looking at what counter arguments have been made against the all examples of that argument label, e.g. if user is reading an argument with label 'cheap medication' the system will look at all arguments that have attacked that in the database).

Whilst the proposals in this chapter have not been evaluated this is certainly something I wish to pursue in the future. For one I would like to understand how poorly extracted data influences the reader's experience. For example if the model has incorrectly identified an argument and a counter-argument how does this effect the conclusion the reader draws.

Chapter 8

Conclusions

This thesis attempts to address the problem patients face when reading medical information on the web. The patient is faced with the task of synthesising and processing the arguments and counter arguments to the drugs found in this information. In order to limit the scope of my investigation I chose to focus on patient provided drugs reviews on the internet as a source of arguments and counterarguments for the drugs due to their increasing commonplace. I stated in Chapter 1 that a number of issues may arise for the user when reading over the reviews. Firstly there is issue of 'volume' that poses a challenge to the reader since they will be required to interpret the opinions in each review, keep track of these opinions and ultimately form a balanced opinion. Secondly there is the issue that reviews are not simply collections of side effects that the drug has caused but also have more complex information that can be modelled with computational models of argumentation.

These problems lead me to the research question I was addressing in the thesis which was **'Is it possible to assist patients in synthesising the information presented in online medical drug reviews?'**. The question was further broken down into two questions namely: **'How can we extract information from the reviews to present to the reader?'**. This question deals with the extraction of arguments and counter arguments in a review since this is what patients are trying to ascertain from the review. The second question is **'How can we structure this information to allow the information to be easily synthesised by the reader?'**. This questions deals with how one can take the arguments and counter arguments and identify

more meaningful structured information between the arguments/counter arguments that will help the user navigate their way through the reviews.

To address this problem I have proposed a pipeline of tasks needed to help patients better synthesise the positive and negative aspects in the drug reviews; in other words arguments and counter arguments in patient provided drug reviews for medications they are interested in. To this end I proposed a pipeline through which arguments are extracted, attacks/argument graphs identified and finally visualised for the user. The first pipe in the pipeline dealt with the task of argument extraction.

In Chapters 3 and 4 I address the issue of 'how to extract important information from drug reviews'. In Chapter 3 I proposed the use of a rule-based approach to extract seven types of arguments. I demonstrated that using a combination of keywords and pattern matching that it is possible to target the extraction of particular arguments relevant to drug reviews. A rule based approach is however limited in that it does not scale as the dataset grows and the types of arguments a user may encounter vary.

To address these shortcomings I proposed a machine learning approach in Chapter 4. I produced an annotated dataset of a total of 680 reviews and identified 41 argument types. After the annotation process I discovered that the annotated arguments were always contained within a single sentence. In terms of choosing a natural language processing model it meant I chose to frame the problem as a sentence classification problem. I chose to use a pretrained language model and used it as a sentence multi class classification model and reported reasonably high recalls and precisions.

In Chapter 5 and 6 I address the issue of 'how to structure the information extracted' in the previous two chapters. In these chapters I dealt with the task of identifying the argument graph for a review. Our proposal makes use of the constellations approach to probabilistic argumentation. I consider the possible set of argument graphs that the review can be assigned and identify as probability distribution over that set. Our probability distribution is given by a function that utilises the rating of the review in order to identify the distribution. The function is designed

such that if the rating is high, graphs with more positive features (defined in Chapter 5) are given more probability mass and vice versa.

In Chapter 6 I extend my approach to be able to incorporate annotator submitted argument graphs for a review. I propose the use of Bayesian inference to be able to update the probability distribution if an annotator submits what they feel is the 'true' argument graph for the review. In order to test my proposal I demonstrate, through the use if synthetic data, that the resulting distribution once annotator provided argument graphs have been incorporated is as expected.

In Chapter 7 I tackle the question of what an end-to-end system would look like. In the Chapter I outline the requirements for implementing the system as an end-to-end system. This includes considering computational costs associated with storing and updating the probability distributions for each review. In addition I make some simple proposals for what I expect the interface to look like. In particular I focus on the user's ability to interact with the argument graph data generated. I suggest the use of text-to-SQL queries that would enable the user to query the argument graph data that has been captured by the system using free-text queries. I also make some basic proposals for how the front-end would look like.

8.1 Contributions

The contribution this thesis makes are:

1. A workflow for training a machine learning model to detect arguments/counter arguments in reviews. This was proposed in Chapter 4. This workflow is flexible and ensures that the users are not limited to a finite set of labels when annotating/training a model. This is valuable since during the annotation process the annotator may come across an argument that does not resemble any argument that they have previously encountered. Once the annotation process is completed the labels are grouped together if they are semantically similar so as to simplify/reduce the set of labels we are training the model to detect.

2. Is the ability to mine argument graphs from reviews through an algorithm

that works both without supervision (Chapter 5 and with supervision Chapter 6 when human provided labelled data is available. Whilst there have been some efforts to mine arguments from reviews [77] none to our knowledge has produced one that can continue to learn from labelled data.

3. Our proposals for visualising the mined argument graphs within the reviews. This encompasses our use of text2SQL models for allowing users to query the arguments using free text queries and our proposals for visualising arguments and counter arguments in a review.

8.2 Limitations

There limitations with the current work are as follows:

1. A full evaluation of the user interface was not conducted. Such as study would be helpful in understanding if the full 'end-to-end' system is ultimately useful to the user. Whilst, efforts were made to test individual pipes an evaluation of the interface with real users would encapsulate all of these. This is something that I would like to revisit in the future.

2. A lack of labelled graph data for the supervised learning chapter meant that we was limited to using synthetic data and simulations. Again a more thorough validation would be something I would wish to undertake going forward.

8.3 Going Forward

Going forward I would like to better focus on the following:

1. The ability of the argument extraction model to extract individual symptoms: Much work has been done of extracting medical terminology from patient provided feedback/text [78] [79]. These models could be used to find more specific arguments in the text related to symptoms. For example we could consider 'I had a back ache' and individual argument and not have to abstract it into a more generic argument type, e.g. 'negative side effect'.

2. Furthermore many more pre-trained language models have emerged in recent years [80] [81]. I would like to explore the potential performance improvements in argument extraction if I was to use these models. In particular I would be keen on bench marking all of the state-of-the-art models against our dataset.

3. Investigate other extensions (preferred semantics) as part of our unsupervised method for identifying a probability distribution for a review. I believe the current proposal could be extended to encompass a wider range of semantics and thus introduce greater flexibility in how the system expresses confidence in an argument graph.

4. Question-answering system: Since the pipeline essentially extract an argument graph for each review it would be nice to explore the possibility of having a chat style interface in which the system can interact with the user using natural language. Much progress has been made in this front with the emergence of large language models [82] [83] [63] that can be fine-tuned to perform conversational tasks. This line of work would mean that patients could interact with the mined arguments through a chatbot in addition to/as opposed to reading through the reviews. This would mean they could potentially parse the arguments related to a drug much faster.

Bibliography

[1] Angela Coulter, Alf Collins, et al. Making shared decision-making a reality. 2011.

[2] Natalie Joseph-Williams, Amy Lloyd, Adrian Edwards, Lynne Stobbart, David Tomson, Sheila Macphail, Carole Dodd, Kate Brain, Glyn Elwyn, and Richard Thomson. Implementing shared decision making in the nhs: lessons from the magic programme. *Bmj*, 357, 2017.

[3] Catherine Foot, Helen Gilburt, Phoebe Dunn, Joni Jabbal, Becky Seale, Joanna Goodrich, David Buck, and Jeremy Taylor. People in control of their own health and care. 2014.

[4] Romy Lamers, Maarten Cuypers, Marieke de Vries, Lonneke vd Poll-Franse, Ruud Bosch, and Paul Kil. Pd47-07 the effect of a preference sensitive online decision aid on localized prostate cancer treatment: First results of a randomized cluster controlled trial. *The Journal of Urology*, 197(4):e898, 2017.

[5] Margaret Holmes-Rovner, Akshay Srikanth, Stephen G Henry, Aisha Langford, David R Rovner, and Angela Fagerlin. Decision aid use during post-biopsy consultations for localized prostate cancer. *Health Expectations*, 21(1):279–287, 2018.

[6] Muhammed Ordu, Eren Demir, Chris Tofallis, and Murat M Gunal. A novel healthcare resource allocation decision support tool: A forecasting-simulation-optimization approach. *Journal of the operational research society*, 72(3):485–500, 2021.

[7] Erin K Tagai, Suzanne M Miller, Alexander Kutikov, Michael A Diefenbach, Ronak A Gor, Tahseen Al-Saleem, David YT Chen, Sara Fleszar, and Gem Roy. Prostate cancer patients' understanding of the gleason scoring system: Implications for shared decision-making. *Journal of Cancer Education*, pages 1–5, 2018.

[8] Siti Noorsuriani Maon, Naffisah Mohd Hassan, and Sharidatul Akma Abu Seman. Online health information seeking behavior pattern. *Advanced Science Letters*, 23(11):10582–10585, 2017.

[9] Young Ji Lee, Bernadette Boden-Albala, Elaine Larson, Adam Wilcox, and Suzanne Bakken. Online health information seeking behaviors of hispanics in new york city: a community-based cross-sectional study. *Journal of medical Internet research*, 16(7):e3499, 2014.

[10] Linqi Lu, Jiawei Liu, and Y Connie Yuan. Health information seeking behaviors and source preferences between chinese and us populations. *Journal of Health Communication*, 25(6):490–500, 2020.

[11] Vladan Starcevic. Cyberchondria: challenges of problematic online searches for health-related information. *Psychotherapy and psychosomatics*, 86(3):129–133, 2017.

[12] Sharon Swee-Lin Tan and Nadee Goonawardene. Internet health information seeking and the patient-physician relationship: a systematic review. *Journal of medical Internet research*, 19(1):e9, 2017.

[13] Sanjiv Ahluwalia, Elizabeth Murray, Fiona Stevenson, Cicely Kerr, and Jo Burns. 'a heartbeat moment': qualitative study of gp views of patients bringing health information from the internet to a consultation. *British Journal of General Practice*, 60(571):88–94, 2010.

[14] Surfing, self-medicating and safety: buying non-prescription and complementary medicines via the internet. *BMJ Quality & Safety*, 12(2):88–92, 2003.

[15] Jennifer Cole, Chris Watkins, and Dorothea Kleine. Health advice from internet discussion forums: how bad is dangerous? *Journal of medical Internet research*, 18(1), 2016.

[16] Sasha Shepperd, Deborah Charnock, and Bob Gann. Helping patients access high quality health information. *Bmj*, 319(7212):764–766, 1999.

[17] Luca Soldaini, Andrew Yates, Elad Yom-Tov, Ophir Frieder, and Nazli Goharian. Enhancing web search in the medical domain via query clarification. *Information Retrieval Journal*, 19(1-2):149–173, 2016.

[18] Luca Soldaini and Nazli Goharian. Learning to rank for consumer health search: a semantic approach. In *European conference on information retrieval*, pages 640–646. Springer, 2017.

[19] Jimmy Jimmy, Guido Zuccon, Bevan Koopman, and Gianluca Demartini. Health cards for consumer health search. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 35–44, 2019.

[20] Christopher V Almario, Michelle S Keller, Michelle Chen, Karen Lasch, Lyann Ursos, Julia Shklovskaya, Gil Y Melmed, and Brennan MR Spiegel. Optimizing selection of biologics in inflammatory bowel disease: development of an online patient decision aid using conjoint analysis. *Official journal of the American College of Gastroenterology— ACG*, 113(1):58–71, 2018.

[21] Bibiana Martinez, Francis Dailey, Christopher V Almario, Michelle S Keller, Mansee Desai, Taylor Dupuy, Sasan Mosadeghi, Cynthia Whitman, Karen Lasch, Lyann Ursos, et al. Patient understanding of the risks and benefits of biologic therapies in inflammatory bowel disease: insights from a large-scale analysis of social media platforms. *Inflammatory bowel diseases*, 23(7):1057–1064, 2017.

[22] Christopher C Yang, Haodong Yang, and Ling Jiang. Postmarketing drug safety surveillance using publicly available health-consumer-contributed con-

tent in social media. *ACM Transactions on Management Information Systems (TMIS)*, 5(1):1–21, 2014.

[23] Felix Gräßer, Surya Kallumadi, Hagen Malberg, and Sebastian Zaunseder. Aspect-based sentiment analysis of drug reviews applying cross-domain and cross-data learning. In *Proceedings of the 2018 International Conference on Digital Health*, pages 121–125, 2018.

[24] Lianzhuang Qu and Patrick YK Chau. Nudge with interface designs of online product review systems–effects of online product review system designs on purchase behavior. *Information Technology & People*, (ahead-of-print), 2022.

[25] Swarnaseetha Adusumalli, HueyTyng Lee, Qiangze Hoi, Si-Lin Koo, Iain Beehuat Tan, Pauline Crystal Ng, et al. Assessment of web-based consumer reviews as a resource for drug performance. *Journal of medical Internet research*, 17(8):e4396, 2015.

[26] Vinaya Manchaiah, Rebecca J Bennett, Pierre Ratinaud, and De Wet Swanepoel. Experiences with hearing health care services: What can we learn from online consumer reviews? *American Journal of Audiology*, 30(3):745–754, 2021.

[27] Kawsar Noor, Anthony Hunter, and Astrid Mayer. Analysis of medical arguments from patient experiences expressed on the social web. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 285–294. Springer, 2017.

[28] Kawsar Noor and Anthony Hunter. Analysing product reviews using probabilistic argumentation. In Henry Prakken, Stefano Bistarelli, Francesco Santini, and Carlo Taticchi, editors, *Computational Models of Argument - Proceedings of COMMA 2020, Perugia, Italy, September 4-11, 2020*, volume 326 of *Frontiers in Artificial Intelligence and Applications*, pages 295–306. IOS Press, 2020.

[29] Kawsar Noor and Anthony Hunter. A bayesian probabilistic argumentation framework for learning from online reviews. In *32nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2020, Baltimore, MD, USA, November 9-11, 2020*, pages 742–747. IEEE, 2020.

[30] Philippe Besnard, Claudette Cayrol, and Marie-Christine Lagasquie-Schiex. Logical theories and abstract argumentation: A survey of existing works. *Argument & Computation*, 11(1-2):41–102, 2020.

[31] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence*, 77(2):321–357, 1995.

[32] Pietro Baroni, Martin Caminada, and Massimiliano Giacomin. An introduction to argumentation semantics. *The Knowledge Engineering Review*, 26(4):365–410, 2011.

[33] Gerhard Brewka, Sylwia Polberg, and Stefan Woltran. Generalizations of dung frameworks and their role in formal argumentation. *IEEE Intelligent Systems*, 29(1):30–38, 2013.

[34] Leila Amgoud and Claudette Cayrol. A reasoning model based on the production of acceptable arguments. *Annals of Mathematics and Artificial Intelligence*, 34(1):197–215, 2002.

[35] Didier Dubois and Henri Prade. An introduction to bipolar representations of information and preference. *International Journal of Intelligent Systems*, 23(8):866–877, 2008.

[36] Leila Amgoud and Srdjan Vesic. A new approach for preference-based argumentation frameworks. *Annals of Mathematics and Artificial Intelligence*, 63(2):149–183, 2011.

[37] Leila Amgoud and Jonathan Ben-Naim. Ranking-based semantics for argumentation frameworks. In *International Conference on Scalable Uncertainty Management*, pages 134–147. Springer, 2013.

[38] Stefano Bistarelli, Paolo Giuliodori, Francesco Santini, and Carlo Taticchi. A cooperative-game approach to share acceptability and rank arguments. In *AI*^{3}*@ AI* IA*, pages 86–90, 2018.

[39] Paul-Amaury Matt and Francesca Toni. A game-theoretic measure of argument strength for abstract argumentation. In *European Workshop on Logics in Artificial Intelligence*, pages 285–297. Springer, 2008.

[40] Elise Bonzon, Jérôme Delobelle, Sébastien Konieczny, and Nicolas Maudet. A comparative study of ranking-based semantics for abstract argumentation. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[41] Phan Minh Dung, Robert A Kowalski, and Francesca Toni. Assumption-based argumentation. *Argumentation in artificial intelligence*, pages 199–218, 2009.

[42] Sanjay Modgil and Henry Prakken. The aspic+ framework for structured argumentation: a tutorial. *Argument & Computation*, 5(1):31–62, 2014.

[43] Alejandro J García and Guillermo R Simari. Defeasible logic programming: An argumentative approach. *Theory and practice of logic programming*, 4(1-2):95–138, 2004.

[44] Hengfei Li, Nir Oren, and Timothy J Norman. Probabilistic argumentation frameworks. *TAFA*, 7132:1–16, 2011.

[45] Anthony Hunter. Some foundations for probabilistic abstract argumentation. *Computational Models of Argument: Proceedings of COMMA 2012*, 245:117, 2012.

[46] Régis Riveret and Guido Governatori. On learning attacks in probabilistic abstract argumentation. In *Proceedings of the 2016 International Conference*

on Autonomous Agents & Multiagent Systems, pages 653–661. International Foundation for Autonomous Agents and Multiagent Systems, 2016.

[47] Hiroyuki Kido and Keishi Okamoto. A bayesian approach to argument-based reasoning for attack estimation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 249–255, 2017.

[48] Learning and updating user models for subpopulations in persuasive argumentation using beta distribution. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems*, volume 17, pages 1141–1149. Association for Computing Machinery (ACM), 2018.

[49] Nico Potyka. Interpreting neural networks as quantitative argumentation frameworks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6463–6470, 2021.

[50] Artur d'Avila Garcez, Dov Gabbay, and Luís C Lamb. Argumentation neural networks. In *International Conference on Neural Information Processing*, pages 606–612. Springer, 2004.

[51] Régis Riveret, Dimitrios Korkinof, Moez Draief, and Jeremy Pitt. Probabilistic abstract argumentation: an investigation with boltzmann machines. *Argument & Computation*, 6(2):178–218, 2015.

[52] Pierpaolo Dondio. Towards argumentative decision graphs: Learning argumentation graphs from data. In AI^3@ *AI* IA*, 2021.

[53] Marco Lippi and Paolo Torroni. Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology (TOIT)*, 16(2):10, 2016.

[54] Simone Teufel, Jean Carletta, and Marc Moens. An annotation scheme for discourse-level argumentation in research articles. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, pages 110–117. Association for Computational Linguistics, 1999.

[55] Zeljko Kraljevic, Thomas Searle, Anthony Shek, Lukasz Roguski, Kawsar Noor, Daniel Bean, Aurelie Mascio, Leilei Zhu, Amos A Folarin, Angus Roberts, et al. Multi-domain clinical natural language processing with medcat: the medical concept annotation toolkit. *Artificial intelligence in medicine*, 117:102083, 2021.

[56] Christos Sardianos, Ioannis Manousos Katakis, Georgios Petasis, and Vangelis Karkaletsis. Argument extraction from news. In *ArgMining@ HLT-NAACL*, pages 56–66, 2015.

[57] John Lawrence and Chris Reed. Mining argumentative structure from natural language text using automatically generated premise-conclusion topic models. In *Proceedings of the 4th Workshop on Argument Mining*, pages 39–48, 2017.

[58] Adam Z Wyner, Jodi Schneider, Katie Atkinson, and Trevor JM Bench-Capon. Semi-automated argumentative analysis of online product reviews.

[59] Simone Gabbriellini and Francesco Santini. A micro study on the evolution of arguments in amazon. com's reviews. In *International Conference on Principles and Practice of Multi-Agent Systems*, pages 284–300. Springer, 2015.

[60] Leila Amgoud and Srdjan Vesic. On the role of preferences in argumentation frameworks. In *Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on*, volume 1, pages 219–222. IEEE, 2010.

[61] Joao Leite and Joao Martins. Social abstract argumentation. In *IJCAI*, volume 11, pages 2287–2292, 2011.

[62] J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.

[63] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[64] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[65] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[66] Jakub Piskorski, Jacek Haneczok, and Guillaume Jacquet. New benchmark corpus and models for fine-grained event classification: To bert or not to bert? In *Proceedings of the 28th international conference on computational linguistics*, pages 6663–6678, 2020.

[67] Ken Barker, Parul Awasthy, Jian Ni, and Radu Florian. Ibm mnlp ie at case 2021 task 2: Nli reranking for zero-shot text classification. In *Proceedings of the 4th Workshop on Challenges and Applications of Automated Extraction of Socio-political Events from Text (CASE 2021)*, pages 193–202, 2021.

[68] John Lawrence and Chris Reed. Argument mining: A survey. *Computational Linguistics*, 45(4):765–818, December 2019.

[69] Aspect based sentiment analysis.

[70] Hai Ha Do, Angelika Maag, and Abeer Alsadoon. Deep learning for aspect-based sentiment analysis: a comparative review. *Expert systems with applications*, 118:272–299, 2019.

[71] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177, 2004.

[72] Andrea Gasparetto, Matteo Marcuzzo, Alessandro Zangari, and Andrea Albarelli. A survey on text classification algorithms: From text to predictions. *Information*, 13(2):83, 2022.

[73] Phan Minh Dung and Phan Minh Thang. Towards (probabilistic) argumentation for jury-based dispute resolution. *COMMA*, 216:171–182, 2010.

[74] Anthony Hunter and Kawsar Noor. Aggregation of perspectives using the constellations approach to probabilistic argumentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2846–2853, 2020.

[75] Oana Cocarascu, Antonio Rago, and Francesca Toni. Extracting dialogical explanations for review aggregations with argumentative dialogical agents. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1261–1269. Association for Computing Machinery, 2019.

[76] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[77] Teeradaj Racharak. Abstract argumentation for summarizing product reviews: A case study in shopee thailand. In *2019 11th International Conference on Knowledge and Systems Engineering (KSE)*, pages 1–6. IEEE, 2019.

[78] Mike Conway, Mengke Hu, and Wendy W Chapman. Recent advances in using natural language processing to address public health research questions using social media and consumergenerated data. *Yearbook of medical informatics*, 28(01):208–217, 2019.

[79] Oladapo Oyebode, Chinenye Ndulue, Ashfaq Adib, Dinesh Mulchandani, Banuchitra Suruliraj, Fidelia Anulika Orji, Christine T Chambers, Sandra

Meier, Rita Orji, et al. Health, psychosocial, and social issues emanating from the covid-19 pandemic based on social media comments: text mining and thematic analysis approach. *JMIR medical informatics*, 9(4):e22734, 2021.

[80] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[81] Tiffany H Kung, Morgan Cheatham, Arielle Medenilla, Czarina Sillos, Lorie De Leon, Camille Elepaño, Maria Madriaga, Rimel Aggabao, Giezel Diaz-Candido, James Maningo, et al. Performance of chatgpt on usmle: Potential for ai-assisted medical education using large language models. *PLOS Digital Health*, 2(2):e0000198, 2023.

[82] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[83] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.