Argument Mining for Twitter

Final Project Report

Author: Shivam Agarwal

Supervisor: Dr. Josh Murphy

Student ID: 1839751

April 8, 2021

Abstract

Recent advances within research in Argumentation Theory, Artificial Intelligence, and Natural Language Processing has laid the foundation for a rising research area called Argument Mining. Sitting at the intersection of the three fields, Argument Mining aims to automatically identify and extract argumentative structures from natural language. This scale of analysis of arguments present in natural language has the potential not only to answer what opinions are being held in public discourse but also provide deep insight into the reasoning behind those opinions. Naturally, argument mining finds applications in domains ranging from Web-based content to legal texts. Within social media, argument mining aims to tap the potential that comes with the dramatically large, unfiltered volumes of data generated regularly. The impact it can have on policy-making, financial markets, and driving public consensus, simply because of its ability to understand public perception, is truly extraordinary. Here we develop an argument mining system for one of the most popular social media platforms, Twitter. We devise a pipeline that aims to extract argumentative tweets and predict the structure of argumentation. The pipeline presented here is built using standard classifiers and artificial neural networks. Furthermore, we also present a novel, annotated data set containing tweets from different topics of discussion. The pipeline performs satisfactorily given the small data set used for training. The annotated data set created here marks a small but progressive contribution to the domain of social media, and can be used for further developments in the future.

Originality Avowal

I verify that I am the sole author of this report, except where explicitly stated to the contrary. I grant the right to King's College London to make paper and electronic copies of the submitted work for purposes of marking, plagiarism detection and archival, and to upload a copy of the work to Turnitin or another trusted plagiarism detection service. I confirm this report does not exceed 25,000 words.

Shivam Agarwal

April 8, 2021

Acknowledgements

I wish to express my deepest gratitude to my supervisor, Dr Josh Murphy, for introducing this challenging yet engaging research idea. His constant support and guidance have been invaluable to the completion of the project.

Contents

Chapter 1

Introduction

Argumentation or Argumentation Theory has been studied since Aristotle, and his ilk first came up with proposals and theories on public discourse and rhetoric. It is truly a multidisciplinary field that has been studied in Psychology, Linguistics, Philosophy and more recently, Computer Science. Formally, Argumentation can be seen as a *verbal, social, and rational activity aimed at convincing a reasonable critic of the acceptability of a standpoint by putting forward a constellation of one or more propositions to justify this standpoint (van Eemeren, Grootendorst, and Snoeck Henkemans, 2002).*

In natural language, Argumentation drives the ability to reason logically. Extracting these arguments from natural language and public discourse using models based on argumentation theory can lead to breakthrough advances in social sciences, policy-making and information technology. Manual analysis of natural texts for arguments is a skilful and time-consuming process. Although attempts have been made to increase the speed of manual argument analysis, it is infeasible to keep up with the rate of data being generated today, and this process requires redefining. Advances in Computational Linguistics, which deals with computational models of natural language, and Natural Language Processing, led to the emergence of a sub-field known as argument mining. Argument mining aims to automatically process and extract arguments and argumentative structures from unstructured textual data and further use the results for analysis.

By providing a digital forum that allows unrestricted access to all its users, Social Media proves to be a goldmine for such data. With such an enormous amount of raw data available, this technology's applications can have unprecedented impacts. Argument mining on social media would enable comprehensive analysis of textual content on social networks and digital

public forums. Consequently, this will boost research in the social and economic sciences. It also has the potential to redefine the industry in terms of marketing and businesses by providing real-time insight into how people think and how they reason. The same could apply to political science, where such an analysis of the public discourse could have significant impacts on policy-making.

However appealing the potential, this is a rather challenging problem. Firstly, this area has only recently received attention and is therefore developing constantly. From a technical perspective, building such systems requires large amounts of annotated data. Despite certain efforts made in this direction, there remains a lack of appropriately annotated data from social media. This raises the issue of scalability of such systems, which are only trained on data sets that are dramatically smaller than the volume of data available on social media. Therefore, finding suitable techniques and designing accurate systems remains a constant challenge.

1.1 Aims and Objectives

Here we aim to develop an argument mining system for a popular social media platform, Twitter. This system will identify argumentative tweets and then predict the internal structure of them. A standard argumentation model, the Toulmin Model, is used to structure arguments. This is achieved by using classification techniques from Machine Learning. There is a strong focus on using Artificial Neural Networks as part of the pipeline. This is motivated by the recent success the technique has seen and also a lack of its application in the argument mining field. We present the different techniques that are tried and compare the performance of these techniques. One issue that continues to be a considerable gap in the field is the availability of suitably annotated data sets, especially in the social media domain. Towards this, we present a novel annotated data set consisting of tweets belonging to different discussion topics - for instance, certain hashtags such as *#Lockdown* and *#Vaccine* - and each tweet labelled as either argumentative or non-argumentative. Through this work, we hope to make a modest attempt at narrowing down some of the current gaps in argument mining.

Chapter 2

Background

2.1 Technical Background

Machine Learning

As described earlier, Argumentation can be seen as a *verbal, social, and rational activity aimed at convincing a reasonable critic of the acceptability of a standpoint by putting forward a constellation of one or more propositions to justify this standpoint* (van Eemeren, Grootendorst, and Snoeck Henkemans 2002). From a Computer Science perspective, Argumentation has shown promise in Artificial Intelligence, where it has been used relatively recently.

Machine Learning techniques are absolutely vital to Argument Mining, simply owing to the fundamental task it entails - automatic extraction of arguments from text data. In his *tour de force*, Mitchell (1997) introduces machine learning as:

A computer program is said to learn from experience E with respect to some class of tasks T, and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

The types of learning can differ with respect to the nature of the task we aim to learn or the kind of experience E the machine is given. Broadly speaking, there are two types of learning based on the experience given to the machine: supervised and unsupervised.

Supervised Machine Learning involves a machine with a task T which essentially is to map

certain input features to some class labels. Some kind of experience E is given to the machine, in the form of training data which holds correct mappings of the task T. Unsupervised Machine Learning involves a machine being given some input data, and the task of the machine is to recognise patterns in the given data and then predict the output labels. Whereas on the basis of the nature of the task T, supervised learning can be segmented into Classification problems which involve mapping a given input to a class label, and Regression problems which aim to map an input to a real-valued quantity. Many of the classification problems can be solved by linear classification. This involves classifying inputs based on the value of a linear expression of the features of the input.

The evaluation of the performance measure P of a program is done using different metrics such as *accuracy, precision, recall*. For a binary classification task with two labels *positive* and *negative*, t_p denotes correctly predicted *positive* labels, and f_p denotes incorrect predicted *positive* labels, and similarly for *negative* labels:

The *accuracy* of a model is defined as proportion of the correctly predicted labels to all predicted labels.

$$Accuracy = \frac{t_p + t_n}{t_p + f_p + t_n + f_n}$$

The *precision* of a model is defined as the proportion of correctly predicted *positive* labels to the sum of all predicted *positive* labels.

$$Precision = \frac{t_p}{t_p + f_p}$$

The *recall* of a model is defined as the proportion of correctly predicted *positive* labels to the sum of all correctly predicted *positive* labels and incorrectly predicted *negative* labels.

$$Recall = \frac{t_p}{t_p + f_n}$$

A widely used metric that combines *precision* and *recall* is the F_1 *Score*. It is defined to be the harmonic mean of *precision* and *recall*.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

There is a wide range of algorithms to deal with both Classification and Regressions problems. Some of the popular algorithms that have been used over the years include Support Vector Machines (SVM), Naive Bayes Classifiers, and Decision Trees. In particular, **Support Vector Machine (SVM)**, which was first introduced in Boser et al. (1992), is a supervised, non-probabilistic model which has been widely used for classification problems. The process begins with mapping n-dimensional training samples into n-dimensional space and then constructing a separator in the form of a hyperplane (In n-dimensional space, a hyperplane is a subspace of dimension (n-1) that maximises the margin or gap between the given classes. Then test samples are mapped onto the same space, and the class is determined based on their position on the map with respect to the hyperplane. Essentially a linear classifier, it can also perform non-linear classification by transforming the inputs to a higher dimension space in order to learn a linear decision boundary. However, this explicit transformation is avoided by using the kernel trick. The kernel trick allows the computation of the dot product of the features in a higher dimension, needed for the construction of the hyperplane, without actually transforming the data. Despite being a binary classifier, it can successfully handle multi-class classification. This is achieved by reducing the multi-class problem into a binary classification problem.

Ensemble learning methods employ two or more learning algorithms to improve the performance of the model. It involves several individual algorithms combined together into a unified model that offers more flexibility in its functions than any of the components. An ensemble classifier is itself trained in a supervised manner and then asked to predict output values. Standard learning algorithms used in ensemble classifiers include Random Forests, Naive Bayes Classifiers.

Another method that has gained much popularity due to its success in a wide variety of machine learning domains is **Artificial Neural Networks (ANN)**. Artificial Neural Networks are computing systems largely modelled on the human brain. An Artificial Neural Network is

a collection of individual nodes or neutrons. Each node processes the input it takes and then passes it on to other nodes as its output. Every node in a network has its own associated weight, which it uses to process the input. Every neural network is parameterised by the weights mentioned above and a bias vector. Generally, an ANN is constructed to have different layers - an input layer, one or more hidden layers and a final output layer. Mathematically, a simple neural network with one layer each, processing input x and making predictions y, weight vectors *U and W*, and bias vector b, would be represented as the following system of equations:

$$h = \sigma(Wx + b)$$

$$z = Uh$$

$$y = softmax(z)$$

Note: σ , known as the activation function, is a non-linear function applied to the linear combination of x. softmax is a function used to normalise the output values to a probability distribution of the classes

While training a neural network, the goal is to learn those parameters (weights and bias) for each layer that enable the network to make predictions as close as possible to the correct labels. The way to quantify this notion of the closeness of predictions and actual labels is using a loss function. Training is done based on the fact that the loss function is differentiable and the knowledge that a function's minimum is when the derivative of the function is 0. Using this, the parameters are learnt by finding the minimal point of the derivative of the function. This algorithm is commonly known as the *gradient descent* algorithm.

Despite its scalability and performance in varied domains, ANNs fail to process sequential information such as time series or textual data. A version of ANNs, **Recurrent Neural Networks (RNN)** store information from previous inputs and then use it in the processing

of the current input. This introduces the notion of the network having a memory and thus enabling them to process sequential information. This is fundamentally achieved by having a recurrent connection in the hidden layer. Another difference in RNNs is that all nodes within a layer have the same associated weights. Theoretically, RNNs are fully capable of using stored previous information to process and analyse the present input. This is practically achieved when the gap between the past information and the current task at hand is small. However, it is seen that when a larger context is required and the gap between the relevant information needed and the current input task increases dramatically, RNNs fail to perform as expected.

Long Short Term Memory (LSTM) architecture, first introduced by Hochreiter (1997), is a type of Recurrent Neutral Network that is capable of analysing and storing long-term dependencies in the input. The motivation behind the development of LSTMs was the investigation into the shortcoming of RNNs as discussed above, and the results thereafter (Bengio et al., 1994). This work presented the inefficiency of the gradient descent algorithm in the context of learning long-term dependencies. Given its innate ability to capture and store long-term dependencies, LSTMs have been used widely and perform tremendously well.

From a Natural Language Processing (NLP) perspective, textual data extracted from the Web is unfiltered and therefore, a lot of the content may be insignificant to the extraction of knowledge from the data. These insignificant elements of the textual data that may hinder data processing and cause limitations are commonly referred to as noise in the data. This could range from characters in the text not belonging to the dominant language - such as emoticons - or even slang words that do not have a dictionary definition. For almost all tasks within NLP, this act of data refining or cleansing is essential to ensure the success of the model thereon built. As discussed earlier, machine learning models are built and trained, fundamentally due to their mathematical foundations, on numeric values. Therefore, employing such learning models on textual data requires an appropriate conversion of the data into meaningful numeric representations. This is generally achieved by creating some sort of vector values out of the textual data. Several algorithms can be used for this depending on the nature of the data, choice of learning model and the eventual goal of the work.

The use of neural networks in Natural Language Processing requires the use of word embeddings. A word embedding is a representation of words, in-text analysis, as a real-valued vector that encodes the word meaning and word context and the similarity of words along certain semantic axes. It is used to transform words into input vectors for neural networks.

Argumentation Theory

The field of argumentation theory studies the act of human reasoning and how certain conclusions are reached based on premises and following a logical form. Parallel progress in Argumentation Theory and Artificial Intelligence saw an increase in the usage of models and theories of argumentation in areas of knowledge extraction and representation and multi-agent systems research. Bentahar et al. (2010) proposed a taxonomy of argumentation models based on three categories: monological, dialogical and rhetorical. Monological models focus on the internal structure of an argument. Dialogical models focus on the ways arguments are structured in dialogues. Rhetorical models deal with the persuasive intent of the speaker and the audience's beliefs. Another axis along which argumentation models can be viewed is the central notion of an argument. Abstract argumentation, much of which was part of Dung (1995), considers each argument as atomic. It sees no internal structure to an argument. Thus, this can be thought of as a dialogical model allowing the definition of attack relations between arguments without looking at internal structure. In contrast, structured argumentation considers arguments a formally structured entity. Thus, most of the argumentation models employed in argument mining are structured, monological models. There have been different models proposed, and there is no single model that can be applied to all domains. One of the more fine-grained of these, is the **Toulmin Model** (S. Toulmin, 1958).

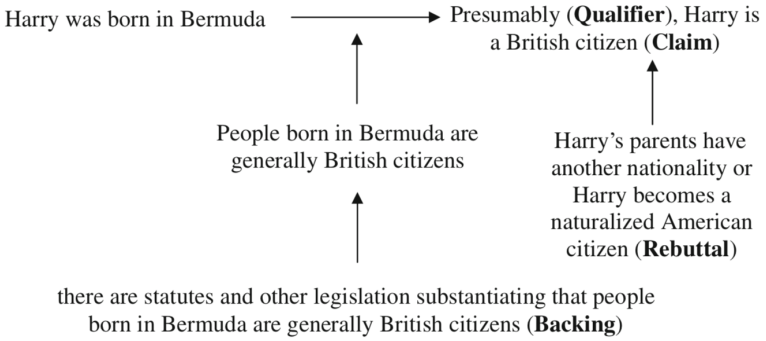

Figure 2.1: An illustration of the Toulmin's Model, (Bentahar et al., 2010)

The **Toulmin Model** proposes that the logical structure of human argumentation and reasoning consists of six categories: an (incontrovertible) *datum*, which forms the basis for

making a (subjective, possibly controversial) *claim*, the rule of inference or *warrant* that links them, and other elements that serve to show how certain we are of the claim, *qualifiers*, or to set conditions for the claim to hold, *rebuttal*, or even to give a justification to the warrant, or *backing*. This model has been largely influential, being applied to various areas of argumentation. Often, the philosophical and empirical foundations of the model are attributed as one of the key factors of its success (Bentahar et al., 2010). There are numerous studies (Habernal, Eckle-Kohler, et al., 2014) that suggest the Toulmin model being more expressive in terms of knowledge representation and also more suitable for short-styled text like microblogs (and hence tweets). (Habernal, Eckle-Kohler, et al., 2014) also propose an extended version of the model. They propose: (1) omitting the qualifier as it is not practiced anymore to denote the level of clarity, (2) omitting the warrant as reasoning for justifying the move from grounds to claims is usually implicit, (3) extending the role of backing, so it provides an additional set of information to support the argument as a whole and not being directly linked to the claim and (4) adding refutation which attacks the rebuttal.

While argumentation models like the Toulmin model help us structure an argument, argumentation schemes provide a different kind of insight into arguments. Argumentation schemes, essentially, are templates upon which real-life reasoning and arguments can be based. They provide means to classify arguments by defining a link between the claim and premise of the argument. This definition of a linkage is done in the form of a logical transformation and is known as a form of inference. There have been many argumentation schemes defined. For instance, *argument from expert opinion* talks about an argument based on the knowledge of some expert source in some domain within which a proposition is made. D. N. Walton (1996) and D. Walton et al. (2008) made major contributions towards furthering research into argumentation schemes. The schemes proposed by these works are also used in argument identification and related tasks.

Extracting data from Social Media (Twitter, in this case) requires the use of API (Application Programming Interface). An API provides a way to interact with a service (Twitter) directly through a software you implement. The Twitter API enables its clients to extract tweets up to a limited number and provides a comprehensive list of tools to do so using different parameters as required.

2.2 Literature Review

Although argument mining is a young field of study, there have been significant contributions made towards it. Research in this area began to appear with one of the first studies dealing with extracting arguments from the legal domain and online public discourse forums (Mochales and Moens, 2011; Cabrio and Villata, 2012). Here we present related work in the field, focusing on user-generated domains and survey papers along with a major real-world project that applies argumentation technologies, and discuss their approaches and successes.

- *Argumentation Mining in User-Generated Web Discourse,* Habernal and Gurevych, 2017

  In this paper, the authors propose to go beyond the state-of-the-art by creating a gold standard annotated corpus centred around different domains ranging from blog posts to comments on news articles, forum posts and similar user-generated content. They also aim to empirically study the traditional Toulmin's model (S. Toulmin, 1958) and propose a revised version of it that is better adapted to the nature of the domain.

  The approach taken here focuses on information-seeking from a user's point of view. This is something that helps expand the field, as it is a fresh perspective towards the problem. It hopes to provide better insights to users who seek information on any controversial topic by presenting both sides of the arguments based on their revised Toulmin's model. Therefore, this is something that has direct applications to the users on the social web. Most related works have only attempted to pick domains that offer more formal textual data, e.g., hotel reviews, student essays or other academic texts. This paper aims to deal with user-generated content on the social web. The nature of data on the social web, as discussed earlier, is noisy and unstructured. Hence, it poses a challenge to process the data automatically. This is something that they aim to overcome.

  A major challenge in the field is the lack of suitable annotated data corpus. Owing to this being a relatively new and emerging field, there are not enough supervised data corpus that fit for all domains, and hence this paper hopes to make a huge contribution of a gold standard data corpus, of 90k tokens in 340 documents, derived from the various domains mentioned above.

  The extraction of argument components and then further analysis on it requires the choice of an argumentation model. The authors claim that this choice is a function of the work scope, and hence different models will act suitable for different domains. A key

research question has been how to effectively model the data extracted from the social web and analyse the arguments. They address this by first extensively studying the empirical evidence of two models - Toulmin's Model and Walton Model. The evidence suggests that there have been applications of Toulmin's model, and it is fairly successful in modelling and representing real-life argumentation. An analysis on Toulmin's model's suitability for the social web was conducted, and the results suggested the need for a modification of the model. The extended Toulmin's model proposed here is backed by empirical evidence and analysis and is hence suitable for user-generated content. The approach taken towards the identification of argument components involves supervised and semi-supervised techniques. It is dealt as a sequence labelling problem using a class encoding that represents all the possible components. This approach was significantly successful, and so do the empirical evidence suggest.

This paper has made significant contributions towards argument mining on user-generated content by creating a proper gold standard corpus. This data corpus marks significant progress towards creating large data sets in the field, especially in the social media domain. The insights and results with respect to argumentation models is something that will further research into similar domains. We take inspiration and use the revised version of the Toulmin's model in our work due to it's suitability to user-generated content as pointed out earlier. The data corpus presented here will be our primary training data set for all segments of the argument mining pipeline. Although this was not the objective of this paper since the paper claimed to have an information-seeking perspective, the choice of domains could have been wider. It could have included more social media platforms such as Twitter, etc., which pose a real challenge for researchers and annotators alike due to the nature of the text. Addressing these challenges can have tremendous academic and social impacts.

- *Argumentation Mining: State of the Art and Emerging Trends*, Lippi and Torroni, 2016

  A leading survey paper on the field of Argument Mining, this proves to be a fundamental introduction to the problem while giving a broad overview of the different approaches attempted and their successes. It also presents the open challenges that remain in a concise way while also directing and furthering future work into argument mining.

  To begin with, it focuses on a problem definition that is solid and provides a base for the rest of the paper. A strong motivation for the problem is presented, which justifies the

increasing interest and attention in the field. It presents information in a very structured and concise manner without losing the essence of the problem, making it suitable for a reader new to argument mining.

It discusses the argumentation models currently being used and the applicability of these models in various domains. It further touches upon the dichotomy of abstract and structured argumentation, and how, at it's core, argument mining deals with structured argumentation which is something that's rarely mentioned in related works. A fine-grained taxonomy of argument mining problems is presented which helps classify problems along different dimensions. This is followed by a core definition of each subtask, the attempted techniques for each subtask, and the associated accuracy and scalability of these attempts. This provides a broad overview of different techniques used, with respect to each subtask. The authors also present a correspondence between these mentioned argument mining sub-tasks and the associated Natural Language Processing tasks, which is particularly helpful in designing argument miners and aids in analysing and comparing the different techniques available.

As with any other task involving machine learning, the availability of well-formed data sets is vital. The authors then present the different data sets currently available covering wide-ranging domains. They further discuss the characteristics of these corpus and also refer to and analyse the associated argument mining systems. This proves to be a comprehensive source for finding data corpus, thus enabling further research into these domains. In the context of social media, too, an array of data sets are presented and discussed, with the focus being on the impact and potential argument mining on social media carries.

Furthermore, it identifies the current gaps in argument mining and consequently discusses the possibility of future work towards closing those gaps. Open problems like dealing with big data and employing unsupervised learning are highlighted. This is something that provides a brief but strong insight into what the future holds for argument mining and also gives a starting point for future work.

To summarise, this paper is a good starting point to understand what the problem of argument mining entails and to gather information about different approaches or solutions already in place. The extensive survey conducted in the paper is quite evident as it attempts to cover the breadth of the field while also backing the discussion with studies and empirical evidence. We grasp most of our understanding of the Argument Mining

problem from this paper. The enormous range of the survey helped us to explore different directions of research and analyse them on the basis of our aims and objectives.

- *Five Years of Argument Mining: a Data-driven Analysis*, Cabrio and Villata, 2018

  A survey-analysis paper on the field of Argument Mining, this work extends the predecessor survey papers and gives a broad overview of the different approaches made and their successes from a data-driven perspective. Within the broad context of Artificial Intelligence, it talks about how the field of argument mining arose as a result of parallel advances in Argumentation Theory and techniques in Natural Language Processing (NLP). It places Argument Mining in the wide spectrum of Artificial Intelligence and maps the relations it holds to other sub fields such as Knowledge Representation and Reasoning (KRR) and Human-Computer Interaction (HCI). It also presents, in a concise way, the open challenges that remain while also exploring future possibilities in the field. In addition, the paper mentions possible areas where Argument Mining can provide meaningful contributions. It explicitly suggests actual real-life scenarios where argument mining can directly be applied (For instance, within the field of politics, argument mining can provide ways to identify fallacies and propaganda).

  Having a data-driven perspective, it focuses on the developments made in Natural Language Processing techniques within Argument Mining, and focuses more on the techniques used within all systems so far, compares them and highlights the shortcoming of each one. It begins with a rather brief definition of the Argument Mining problem by mentioning the two crucial stages - Argument Detection and Relations' Prediction. This may not be suitable as an introductory paper meant for newcomers to the field. The lack of an in-depth definition may possibly be because the author aims to address a more technically mature audience than earlier survey papers did. Hence, it chooses not to delve into the fundamentals of the problem but rather highlight the state of the art and expand more on the techniques used and discuss their shortcomings. All approaches and attempts surveyed in this paper, are classified into different domains - Education, Legal Text, Web content, etc. This classification helps in comparing different attempts within a single domain. Specifically, in terms of Web Content and Social Media, it highlights the major contributors so far and also presents current work in progress. Lastly, this survey paper establishes argument mining as one of the most promising fields, and successfully expresses the increasing interest and efforts towards the field. It provides some actual

ground in terms of the future scope of applications of argument mining and has helped in directing us to recent developments made.

- *Re-using an Argument Corpus to Aid in the Curation of Social Media Collections*, Llewellyn et al., 2014

  This is an investigative paper that aims to explore the feasibility of automated argument extractions from social media. In the process, it creates an annotated data set to train different Machine learning algorithms and compare them, and further identify the challenges faced in dealing with data from social media. Furthering existing work, it uses an existing annotated data set to train machine learning models on. In order to test the applicability of the model, other data sets obtained from open source contributions are also tested. The algorithms that are tested are Naive Bayes, Support Vector Machines and Decision Trees. It is important to note here that this paper does not choose a particular argumentation model but rather defines different argument classes themselves and uses that as a baseline to classify the text from social media. This is something that seems unconventional because argument mining normally involves the modelling of arguments using a predefined model. Having a formal argumentation model provides for deeper analysis, and ensures clarity in the representation of arguments, and allows for comparisons to other solutions. One reason for not conferring to a particular argumentation model might be the nature of the work - which is investigative in nature, which aims to compare different techniques and the suitability of the domain, and hence the lack of focus on a model.

  A key contribution made by this work is the analysis results. It is seen that Decision Trees prove to be the most accurate among the techniques used. They also further assert that it is difficult to pin down one algorithm on the basis of empirical evidence because the suitability and success of algorithms depend on the domain of work and chosen data set.

  They also go on to highlight some issues with handling Twitter data which will be highly beneficial for future work on Twitter. They express the issue with handling retweets, which is an important feature of Twitter. They discuss how the presence of retweets in the training data can lead to redundancies and overfitting of the model. All the findings in this paper help contribute to future work on Twitter or similar domains. It will help us make informed design choices and also better grasp the features of Twitter data, and

predict the challenges we might face.

- *An autonomous debating system*, Slonim et al., 2021

  The field of computational argumentation is very young and is constantly evolving. This paper here proposes to build a real-world autonomous debating system named Project Debater. This project's importance and relevance here is emphasised because there have not been many argumentation technologies built so far. Therefore, this project sets a milestone for research within computational argumentation and shows its applicability to the real world. The primary goal of the project is to build an autonomous agent that can reason and argue with a human agent on a proposed topic of debate. They highlight the key challenges that come with the project from a language processing perspective. Given these challenges, they propose an approach that breaks down the problem and consider it as a set of smaller tasks in parallel. The research required extensive work in some of the sub-tasks, and as a consequence, it fostered research into those subfields. The system proposed here includes argument mining, an argument knowledge base, argument rebuttal and debate construction. The operation of the system begins with a given motion of debate. The argument mining component of the system deals with two stages. The first stage uses a large database to extract sentences which are then used as a source to identify arguments. Claims and Evidences related to the motion of debate are extracted from the sentences. The second stage uses a set of neural models and knowledge-based models to classify the stance of each argument extracted. The next part of the system is the knowledge base which acts as a platform to store general information common to all formal debates. This provides an added dimension that helps the system to highlight topics that may be related to the motion and thus further their stance. The rebuttal component of the system keeps track of the claims that could be mentioned by the opponent and creates possible counterarguments. Finally, the debate constructor sets up a coherent debate speech using all other modules. The consequent analysis of the system shows that the system manages to outperform any other real-world debating systems that exist. As noted by the authors there is a lack of a formal evaluation metric for debates which restricts a formal evaluation of the system. A strong focus is on the comparison between the grand challenges of AI and this project. They mention how their work aims to take the field to a new domain that does not have solid frameworks or metrics to evaluate the model's performance. Therefore, Project Debater stands out as a leading application of research within computational argumentation and present a new paradigm of problems

within AI that will help advance the field further.

Chapter 3

Annotated Data Set for Twitter

Data Annotation is the process of labelling naturally available data such as text or images. Within the realm of supervised machine learning this represents an important stage in building a reliable and effective model. With the large amount of data being generated on a constant basis, and the increasing need of machine learning models, this creates an increasing requirement for labelled data sets. Likewise, the field of argument mining faces a constant need for publicly available annotated data sets. Furthermore, in the domain of social media or user-generated content there is a lack of appropriate data sets. During the initial background study and review of related work, this shortage of data sets was noticed. Therefore, recognizing this gap, here we aim to create an annotated data set consisting of roughly 900 tweets. The tweets are labelled to either be argumentative or non-argumentative. This will prove beneficial for tasks such as argument identification which often turns out to be the first step in argument mining.

3.1 Related Work

Current attempts at creating annotated data sets for the social media domain are mostly limited to Twitter. Llewellyn et al. (2014) in their work on argument mining on Twitter use the dataset provided by Procter et al. (2013). It consists of tweets initially annotated during the London riots in 2011. DART, an annotated corpus for Twitter was introduced by Bosc et al. (2016). It consists of roughly 2200 tweets annotated as argumentative. This data set is quite relevant to the project, and would be an appropriate choice to train our model upon. However, it was found that this data set was no longer available publicly.

3.2 Data Annotation

Here we aim to label data from Twitter (in the form of tweets) as either *argumentative* or *non-argumentative*.

The first step is the extraction of data from Twitter (more of which is discussed in the chapters that follow). To ensure that the data extracted is rich in argumentative features, it is reasonable to query tweets based on current real-world happenings in public discourse. We extract tweets associated with the hashtags - #Vaccine and #Lockdown. Once this is achieved, the next step is the labelling of the extracted data. Towards achieving this, we follow the guidelines suggested in Bosc et al. (2016). The guidelines talk about how the labelling should be carried out by aiding in the decision-making process. Here we present the guidelines along with a use-case example.

Tweets that present an opinion held by the author are labelled as *argumentative*. For instance, in the following tweet the author makes his opinion clear in the last line - *When it's your time, take your shot.*

Tweet: *I took the #COVID vaccine, because I believe in the first principle of my Christian tradition: Love your neighbor as yourself. There's a spot for everyone. When it's your time, take your shot*

Tweets that contain claims presented as rhetorical questions are also considered as *argumentative*. For instance, there is a rhetorical question being asked here.

Tweet: *Govt refuses to do a paid lockdown. We remain open while racing to vaccinate, creating variants that show signs of vaccine resistance. At the same time, we're protecting drug company IP, hamstringing vaccination efforts abroad. And this is the response? https://t.co/q7NKA70S6V*

Tweets written in a sarcastic or ironic tone are also labelled *argumentative*, due to the nature of the content. For example, the following tweet uses sarcasm to express it's opinion against lockdowns and curfews.

Tweet: *Finland. One year of pandemic! Now we are going in lockdown and putting curfews in some areas. How cool is that! Yeyyy we wanna play pandemic too, because everyone else had their pandemic, why can't we have too. https://t.co/LUyj83u72n*

Tweets that contain links to factual information are also labelled *argumentative* as they may be the source of a premise or backing. Here, the link provided at the end acts as a source of credible information.

Tweet: *IMP INVESTIGATION India's lockdown was called world's largest, strictest etc. @ BBC, we wanted to know how the govt planned it. So, what're we looking at? Govt docs that reveal every department/agency who was consulted by PM Modi before imposing #lockdown. #Thread https://t.co/Ph6COQQF8O*

Throughout the annotation process, it was important to ensure that noisy data, tweets not related to the topics of discussion are ignored and correctly identified as *non-argumentative*. This mainly consisted of advertisements tweets, automated tweets generated by bots, or tweets whose content had no connection to the topics of discussion. For instance, the following tweet has no relevant information

Tweet: *#VaccineNation2021 #vaccination #vaccine #MovieNight #movie #Filmmaker #films #Filmmaking A #LongIsland couple is forced to deal with their relationship #AmazonPrimeVideo #Cinema #amazonprimevideowithvodacom #lirr The F3ar Movie on #Amazon (https://t.co/DpmXpdIElS) https://t.co/99W5u0cpAH*

3.2.1 Annotation Tool

To facilitate this process of annotation, we use a data labelling tool, *Label Studio*. It is an open-source tool that allows for data labelling. It provides an easy-to-use interface that assists in labelling data corpus, one element at a time.

Finally,

No.	Argumentative	Non-Argumentative	Hashtag
1	106	328	Vaccine
2	183	236	Lockdown
Total	289	564	853

Table 3.1: Breakdown of Labelling

3.2.2 Remarks on Twitter Data

During the labelling process, certain observations were made about the nature of data on Twitter.

A vast majority of content posted by users contain emotional content. From a rhetorics perspective, this is the call to *pathos* - appealing to the readers' emotions and using it to convince them. Certain tweets were highly emotive and largely lacked logical content. The Toulmin model used in the project, therefore, is not ideal in modelling emotive components.

An issue with the structure of the texts, is that most tweets are not grammatically sound. They occur in small phrases, some of which make little grammatical sense.

Chapter 4

Design & Specification

4.1 Design

The approach taken here breaks down the problem into two individual subproblems, namely *Argument Identification* and *Argument Component Detection*. Argument Identification is the classification of tweets into *argumentative* and *non-argumentative*. Argument Structure Detection breaks down the argumentative tweets into their individual components based on the extended Toulmin model presented earlier.

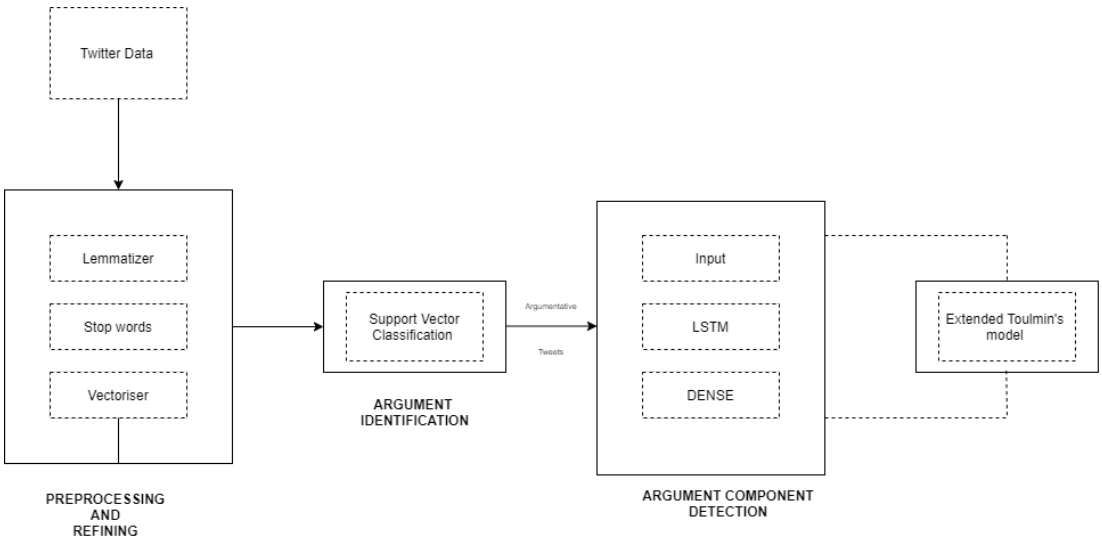

Figure 4.1: **Pipeline blueprint**

Although the pipeline's basic mould is identical to most other systems or pipeline used for argument mining, certain parts are slightly different. For instance, most studies employ a claim/premise model. This model proposes that arguments have two basic components namely, *claim* and *premise*. This project deviates from this trend and explores a revised argumentation model, the extended Toulmin model (Habernal and Gurevych, 2017). As pointed out in Lippi and Torroni (2016) A major chunk of existing argument mining systems begin the pipeline with the identification of argumentative tweets and detecting the component boundaries. Here, we skip the detection of boundaries and assume that an entire tweet can either be argumentative or non-argumentative. As discussed above, the noisy nature of user-generated content makes does not allow for effective boundary detection. The informal textual data present on Twitter may not entirely be suitable for such a task and may lead to inaccuracies.

4.1.1 Twitter Data

In this project, we use Twitter as our social media platform to get raw textual data. Users enter text on the platform in the form of short blocks of characters, called Tweets. Each tweet is limited to 280 characters. Hashtags, a way of indicating keywords and topics, are pretty popular on Twitter, and the extraction of data in the project is parameterized on different hashtags - #Vaccine, #Lockdown, etc.

Extracting data from Social Media (Twitter, in this case) requires the use of API (Application Programming Interface). An API provides a way to interact with a service (Twitter) directly through a software you implement. The Twitter API enables its clients to analyze tweets' performance, stream tweets in real-time, and extract past conversations. It allows us to extract tweets up to a limited number and provides a comprehensive list of tools to use different parameters as required. The Twitter API requires creating a Developer account that provides you tokens to access the various endpoints of the API.

4.1.2 Preprocessing & Refining

The nature and genre of the tweets posted by users can be informal and wide-ranging. This being a social media platform, neither is the text necessarily formal nor is it grammatically structured. Along with noisy elements, such as emoticons, slang words, and other tags, this

poses a challenge from a Natural Language Processing perspective. This is where ideas like *lemmatization* and *stop words* from Natural Language Processing come in. Lemmatization is the process of detecting the scenario where two seemingly different words originate from the same root. This is followed by replacing all such words with their common root, or *lemma*. For example, the words: *play, player* and *playing* all have the same common root - *play*. Here, all the three words would be replaced by their lemma, *play*. Similarly, *is* and *are* are replaced by their lemma *be*. Lemmatization allows for replacing all words with their base dictionary version, ignoring the plurality, tense or other grammatically morphed versions. Thus, eliminating superficial differences in the data set and creating a data set rooted in the foundations.

Stop words are extremely common words that hold negligible value in representing our data and neither hold information about the model. In the English language, words like *the, and, we* would be considered stop words as they provide negligible information. Removing such stop words is important because it allows the focus to remain on more important words that help characterize the input data in a more comprehensive manner. Stop words can be defined for any language; and also for specific domains. For social media too, there is a possibility of creating an exclusive set of stop words. However, we choose to use the standard set of stop words in the English language, as they seem to be sufficient for the task at hand. All of this leads to refining data, thus making it more suitable for passing onto the classification stages in the pipeline.

When dealing with textual data, it is essential to have an appropriate transformation of these words to a vector containing numeric representations. This is known as the preprocessing of data. Here, we use a popular method known as TF-IDF Vectorizer. It transforms the given textual data into a meaningful feature representation by analyzing how important a word is to its document while also keeping in mind the general commonality of the word in the language. Another approach to preprocess data would be to use CountVectorizer. This creates a vector representation of the count for each word in a given document. This results in an inclination towards more frequent words, and certain rare words that hold vital information may not be given attention. On the other hand, as noted above, the TF-IDF Vectorizer uses the overall weight of a word in a document. This can be considered an upgrade from the CountVectorizer. Hence, we choose to work with TF-IDF Vectorizer.

4.1.3 Training Data

Like all machine learning problems, argument identification and subsequent component detection requires annotated training data to run these models. The training data used here was presented in Habernal and Gurevych (2017). The data set created by them was obtained from several web content sources such as discussion forum posts, blog posts, and newswire articles. The first part consists of texts labelled as either persuasive (or argumentative) or not. This will be used to train the Argument Identification model of the pipeline. The second part deals with labelling the different components of persuasive texts based on the extended Toulmin Model also presented in the paper. This will act as the training data for the component detection model. Both Twitter and the sources mentioned here share the same user-generated content paradigm. Furthermore, the genre and size of the texts presented in both these sources are similar. Hence, considering the general shortage of data sets and the potential suitability of this data set to the social media domain, this seems like a reasonable choice.

4.1.4 Argument Identification

The first stage of the pipeline after data is preprocessed, is the identification of arguments. This entails the classification of extracted tweets into argumentative and non-argumentative. Since tweets can be considered to be the basic unit of dialogue and discourse on Twitter, it is reasonable to assume that they will contain at most one argument. This helps to define the granularity of the task. From a Natural Language Processing perspective, argument identification is basically a text/sequence classification problem.

The selection of features when dealing with natural language is key to the success of the model's performance. Here, the *unigram and bigram model* is used to represent the features of the model. Working along with the TF-IDF Vectorizer mentioned above, the bigram model uses sequences of 2 words to extract features while unigrams pick single words. Considering this, standard learning models can be employed towards solving this. Here, we propose to use a Support Vector Machine that will perform binary classification on the training label. Several studies have used Support Vector Machines in some kind of argument mining pipeline and have found reasonable success (Mochales and Moens, 2011; Stab and Gurevych, 2014; Eckle-Kohler et al., 2015). To give a comparison, an ensemble classifier consisting of a Random Forest classifier, a Naive Bayes classifier and the Support Vector Classifier mentioned above is also built. Both

these models will be trained using the training data mentioned above - it will consist of samples from various web sources labelled as either argumentative or not. During the evaluation, we aim to present and discuss the performance of both methods.

4.1.5 Argument Component Detection

The final and arguably the most complex part of the pipeline is the argument component detection. Here, the argumentative tweets, identified and extracted in the Argument Identification model, are processed and broken down into individual argument components. This part of the pipeline is heavily dependent on the argumentation model used, and the granularity of the desired target structure. For instance, a different pipeline could employ a claim/premise model, and then the task would be to predict the link between them. When employing more sophisticated models, the task at hand changes to correctly predicting all the components to analyze the final structure. Here, we use the extended Toulmin Model proposed by Habernal and Gurevych (2017). This is the part of the pipeline that drives the understanding of the reasoning behind the opinions or standpoints. By employing an argumentation model, we can dissect the argument into the individual parts; thus leading to a more theoretical and profound understanding of the standpoint and its associated reasoning.

Current argument mining systems that deal with this task of component detection have put to use several traditional machine learning models such as Support Vector Machines. Here, in this project, we choose to explore the suitability of artificial neural networks (LSTM, in particular) for this task. Using neural networks for argument mining has not been considered as extensively as traditional classifiers have been. Cocarascu and Toni (2017) employ deep learning models for relation-based dialogical argument mining to identify relations of attack and support between arguments. Here a Long Short Term Memory model is built that works as a multiclass classifier to predict the individual argument components. The training data used is as mentioned above. As is the case with all other neural networks, there is a need to transform words from the argumentative tweets into vectors. Here, we use a popular word embedding tool - GloVe: Global Vector for Word Representation (Pennington et al., 2014). GloVe works as an unsupervised learning algorithm by mapping words into a meaningful space, factoring in the semantic similarity of words. Then, the vector representations are obtained from the vector space by aggregating the word-word co-occurrences. GloVe provides pre-trained word embeddings built from various sources such as Twitter, Wikipedia. The availability of a pre-trained embedding

for Twitter was the prime reason for choosing GloVe over other state-of-the-art embeddings. On the other hand, it also provides functionality to train and learn the embeddings based on a new data corpus. Here, pre-trained word embeddings are used because of the fact that the target domain on which these embeddings are trained is in fact, Twitter. Finally, we aim to share and discuss the results obtained by this neural network.

4.1.6 Extended Toulmin Model

Every argument mining system requires the selection and usage of a theoretical argumentation model. The argumentation model provides a theoretical and empirical foundation for the analysis of arguments. Many studies, such as Habernal and Gurevych (2017), claim that the argumentation model is a function of the domain of the task, and hence the performance of a model may vary in different domains. Any such essential design choice should ideally be made on the basis of empirical evidence and past success. Therefore, we follow on from the extensive study of the applicability of the Toulmin Model done in Habernal and Gurevych (2017). Even though there have been critics in the past, who challenge the usability of the Toulmin Model in real-life argumentation (closely linked to user-generated content), several studies have advocated for its use in modelling real-life arguments. Simosi (2003) uses the Toulmin Model and examines the arguments made by employees of a company in order to resolve conflicts in their workplace. Weinberger et al. (2007) explore the idea of analyzing arguments made by users on computer-supported collaborative learning platforms. Therefore, based on these studies, the Toulmin model can be considered to be suitable and give satisfactory results for argument mining on short spans of text, such as blog posts and forum discussions. Habernal and Gurevych (2017), test the applicability of the original Toulmin model on a small data set and observe certain redundancies in the model. Therefore they propose a slightly modified, extended Toulmin Model.

Originally, Toulmin proposed the use of a *qualifier* - used to denote the clarity or probability of the *claim*. However, the study found that that there is no explicit usage of the *qualifier* and hence the extended model does not use it. Similarly, in his original model, Toulmin included a *warrant* that gives a logical explanation of why the claim should be accepted based on the premise and backing. Nevertheless, S. E. Toulmin (2003) also went on to express that warrants are made implicitly. Therefore, the extended model chooses to omit the use of a *warrant*. In the original model, there exists a *rebuttal* which is a statement made to attack the validity of

the claim. As seen in general public discourse, it is often the case that *rebuttals* are followed up by a counter-rebuttal made by the original claim's author, known as a *refutation*. It is often the case that the claim made by a speaker in an argument is not explicitly stated. Therefore, the model extends the Toulmin model by allowing implicit claims and manually writing the claim when annotating the data. To sum it up, the modified Toulmin Model consists of the following argument components: *claim, premise, backing, rebuttal and refutation.*

Another possible approach would be the use of argumentation schemes such as Walton's schemes (D. Walton et al., 2008). Argumentation schemes provide a template to represent various kinds of reasoning and argumentation. Even though these argumentation schemes appear to be domain-independent, questions have been raised on their application to real-world argumentation in natural language textual data. For instance, Walton himself observed the unsuitability of certain argumentation schemes in political argumentation; this required the creation of new schemes to give satisfactory results. As presented in Habernal and Gurevych (2017) several studies found that the schemes fail to adapt to natural language content. Bentahar et al. (2010) note that the Walton schemes are based on informal logic and therefore are unable to define the formal relation between different argument components.

Given the empirical evidence discussed earlier, this project employs the modified Toulmin Model for the prediction of internal argument structure. To facilitate a swift process, and also keeping in mind the nature of content on Twitter, we assume that each tweet can contain at most one argument.

4.2 Specification

4.2.1 Requirements

Must

- Setup connection with the Twitter API in order to get data.
- Extract tweets (under the limits provided by the Twitter API) when needed.
- Query and Extract tweets using hashtags or other search parameters.
- Store the extracted tweets in an appropriate file format.

- Identify and present a structured pipeline of the solution.
- Identify and use a suitable argumentation model.
- Parse through the tweets file and prepare and preprocess the data for the next step in the pipeline.
- Consider the ethical and professional issues and practice good code.

Should

- Extract real-time tweets on search parameters (hashtag or search queries)
- Present an in-depth explanation of the approach taken.
- Train and use a classification model for argument identification.
- Train and use a classification model for component detection.
- Create an annotated data set for Twitter.

Could

- Compare the performance of various supervised and unsupervised classification techniques on the problem
- Compare the performance of different features on the training data.

Chapter 5

Implementation

Towards implementing this model's design as proposed earlier, we start by setting up a Python environment for all our source code development. The choice of Python is primarily motivated by the fact that almost all of the machine learning models have been built using Python. There are several factors behind this. Python was created to be a readable, high-level language that allows creating simple yet effective programs. There is a vast active community behind the language that constantly reviews the language and updates it. Most importantly, many libraries and packages are supporting the language that helps create mathematically consistent machine learning models. Therefore, it would be reasonable to state that Python is considered the current state of the art in Machine Learning, and more broadly, Artificial Intelligence.

5.1 Data Extraction from Twitter

As mentioned earlier, the first step in the argument mining pipeline is the collection of data from Twitter. Like many other web services, this is done using the API (Application Programming Interface) provided by Twitter. The Twitter API is extensive and has a comprehensive list of endpoints that can be used—ranging from extracting tweets based on a set of parameters to posting tweets from a user account. To use these endpoints in the API, we need to set up a Developers Account with Twitter. This requires the submission of a request that can either be for academic purposes or commercial use. Given the nature of the project, we chose to apply for the academic track. The request mainly contains the intended use-case of the project to better understand the requirements and limitations of the project. Once this is set up, we use Tweepy for our extraction of data.

Tweepy, a Python library, allows users to access and use the Twitter API in a much easier and straightforward manner. It is installed using the Python Package Index (PyPI), an online repository of software packages for the Python programming language. Since the project's needs were only limited to a basic tweet extraction task, instead of directly using the Twitter API, we use Tweepy. Using the Twitter API would have been the suitable option only if we are engaging in more complex tasks with Twitter - such as analysing the performance and streaming real-time tweets. The use of Tweepy provides a simple and basic environment to extract data from Twitter. We set up a Python environment where we import and set up the required Tweepy library. The standard endpoint to search for tweets is `api.search`. The Twitter API also provides a way to combine large results. This is done using a technique called *Cursoring*. This creates different pages of results and also allows both forward and backward movements in these pages. Without *cursoring*, each search query results in a page of only 500 tweets at the upper limit. We use this technique to ensure we can extract a large number of tweets and all extracted tweets are stored together.

The API allows for many parameters when searching for tweets. Tweets are extracted using the following parameters:

- **q** - the search query string that can include hashtags or particular hot topics. Here we use two hashtags - #vaccine and #lockdown
- **lang** - restricting search results to a particular language. Here, we restrict ourselves to only English tweets.
- **count** - this determines the number of results retrieved per page. The default value, 100, is chosen here.
- **tweet_mode** - This allows tweet results to either be extended or compat. The extended mode allows for every tweet with its entire text attribute. Given the nature of the task at hand, having the entire text in a tweet is vital; therefore, we use the extended mode here.

To do this task the project uses the following packages and libraries.

- **demoji** - It is a popular Python library that removes emojis from text strings.

An important consideration here is how to deal with *retweets*. Retweets are removed from the final data set because it is safe to assume that retweets are not argumentative. Simply retweeting something does not clearly reflect a user's opinion or standpoint.

Before creating the final data set, we also remove all forms of emojis and emoticons as they mainly increase the noise in data and in no way add any value to the text. Once this refining is done, we export the data extracted into a `csv` (comma-separated values) file. The file is structured in a way that has three columns, namely *Index*, *Text/Tweet* and *Label*. Having data stored in such a format is ideal for labelling data and then, furthermore, using that data to train models.

5.2 Training Data

The primary source of training data used here, as discussed above, is the gold standard data corpus provided in Habernal and Gurevych (2017). The data corpus provided by them consists of two subsets. One consists of around 990 data points labelled as either argumentative or not. The second has around 990 elements labelled according to the different parts of the extended Toulmin model presented in the same work. The data corpus has been annotated using the UIMA XMI Standard. In order to extract these annotated data points into whatever desired format, the authors have created a tutorial that includes Java source code that reads these data points and converts them into a readable format. Using this resource, we convert the data sets into a `csv` file format. This is followed by some amount of manual reorganisation and refining of data. Keeping in mind the end purpose of the data set, we remove all unnecessary parts.

5.3 Argument Identification

Argument Identification is the second stage of the pipeline that involves the classification of tweets as either argumentative or non-argumentative. Towards building this, we set up a Python environment that uses the following packages and libraries.

- **scikit-learn** - This is an open-source Machine Learning library that supports various features such as classification, regression and clustering. The algorithms include Decision Trees, Support Vector Machines, etc.
- **pandas** - This is an open-source data manipulation and analysis tool.
- **nltk** - This is a range of tools and libraries aimed for natural language processing in English.

The first step is importing the training data created in the previous section. This is done using the *pandas* library. A Dataframe in *pandas* is a two-dimensional data structure that is primarily used to store data sets. Here, too, we create a Dataframe consisting of two columns, namely Text and Label, to store the data extracted from the training data `csv` file.
The next step is to clean and preprocess the data. To do this, we use the stopwords and WordNetLemmatizer modules from *nltk*. WordNet is an online lexical database of English words and their semantic relations. The WordNetLemmatizer uses WordNet to lemmatise the input words. The stopwords module provides a list of common words in English that hold negligible value in information extraction. The refined data is then transformed into a feature vector using *scikit-learn*'s TFidfVectorizer. This uses the TF-IDF model mentioned earlier to create a vector of the chosen features. The features chosen here include unigrams and bigrams. This was done after experimenting with other possible features. We also present the results and further discuss this in Chapter 7.
In the next step, the training data set is split into two subsets - one for training the model and the other for testing the model. This is achieved using *scikit-learn*'s train_test_split module. A common ratio for splitting data sets is 80% training data and 20% testing data. Given the small size of our used data set, we also chose to go with this ratio. Once the segregated training and test data have been created, the machine learning model is defined. Here, we build a Support Vector Classifier (SVC) provided by *scikit-learn*. All default parameters are used for this. For instance, the default kernel function 'radial basis function' is used. Next, the ensemble classifier is built using the *Voting Classifier* module provided by *scikit-learn*. This consists of three individual classifiers: Support Vector Classifier, Gaussian Naive Bayes Classifier, and Random Forests. The final classification results are based on the majority result given by three individual classifiers. Finally, to test and evaluate the performance of the models described above, we used various metrics such as Accuracy, Cross-Validation and F1 Scores.

5.4 Argument Component Detection

This final step of the pipeline aims to predict the internal structure of a given argument correctly. The approach proposed earlier is to use Recurrent Neural Networks (RNN). More specifically, we build a Long Short Term Memory architecture. Towards building this, we set up a Python environment that uses the following packages and libraries.

- **TensorFlow** - It is a popular, open-source machine learning platform.

- **Keras** - It is a deep-learning API intended to be used in Python. It is built on top of *TensorFlow*. The simple and consistent API provided ensures a smooth developer experience. The main reason for choosing this as our deep learning framework is the wide range of functions on offer and the simple and clean usage of each. It has been widely adopted as the go-to framework both for academic and commercial purposes.

- **Matplotlib** - This is a Python library that is used to create different kinds of mathematical visualisations.

- **scikit-learn** - This is an open-source Machine Learning library that supports various features such as classification, regression and clustering. The algorithms include Decision Trees, Support Vector Machines, etc.

- **pandas** - This is an open-source data manipulation and analysis tool.

- **nltk** - This is a range of tools and libraries aimed for natural language processing in English.

First, using the *pandas* library, a Dataframe consisting of the training data is built. Similar to the previous stage, it consists of two-column - Text and Label. The data thus obtained needs to be refined and preprocessed before being used in the LSTM model. To do this, we use the stopwords and WordNetLemmatizer modules from *nltk*. WordNet is an online lexical database of English words and their semantic relations. The WordNetLemmatizer uses WordNet to lemmatise the input words. The stopwords module provides a list of common words in English that hold negligible value in information extraction. As is essential with all neural networks, this data needs to be converted into real-valued vectors that represent the meaning and context of the word; this is done by using word embeddings. Here, we employ the use of pre-trained GloVe embeddings as discussed earlier.

GloVe (Global Vectors for Word Representation) embeddings contain pre-trained embeddings on domains such as Wikipedia and Twitter. These pre-trained embeddings are available with different feature dimensions - 25,50,100, and 200. The Twitter embeddings are of particular interest to the project. This consists of 2 billion tweets or 27 billion tokens. This vast-sized embedding is used here because it being on the same domain as this project. Here, the 100-dimension subset of the pre-trained embeddings is used. Using this embedding, first, a dictionary of all the words are created that holds their associated real-valued vectors. From this dictionary, we create a GloVe matrix of all sentences in the training data set. This uses each

word's values from the dictionary and then uses the mean value for each dimension from each word to generate the final vector for a sentence. This is then stored in the matrix.

For instance, to convert the sentence *Please pass me the bottle of water* using the GloVe embeddings, first, the values for each word (*please, pass, etc.*) is determined. Each of these values will be a vector of 100-dimensions. Then for each of these 100-dimensions, all the words' values are summed up, and the mean is calculated. These set of mean value are then taken to be the final vector for the entire sentence.

Similarly, another matrix is created for the different possible labels - which are the different parts of the Toulmin model.

With the GloVe embedding in place, in the next step, the training data set is split into two subsets - one for training the model and the other for testing the model. This is achieved using *scikit-learn*'s train_test_split module. A common ratio for splitting data sets is 80% training data and 20% testing data. Given the small size of our used data set, we also chose to go with this ratio.

The chosen Recurrent Neural Network for this stage of the pipeline is the Long Short Term Memory (LSTM) model. *Keras*, allows for the creation of an LSTM network. Any neural network is defined by its individual layers; the input layer, the hidden layer(s) and the output layer. Here, the network is defined using the `Sequential` module from *Keras*. This is used to define a model of neural networks. The first layer is the input layer that takes as input the embedded vectors and passes them onto the hidden layer. The hidden layer here is an LSTM layer of 64 units that uses *ReLu* or Rectifier as its activation function. Finally, the output layer is a Dense Layer of 5 units with *softmax* as its activation function. Finally, the model is trained on the *categorical_crossentropy* loss function with an *adam* optimizer. The model is trained for 14 epochs with a batch size of 64. Although most state-of-the-art approaches to deep learning use several layers stacked together, here, we choose to use a single hidden layer. This was mainly done keeping in mind the small size of the available training data. The task at hand was a simple multiclass classification. Therefore, using highly complex networks might lead to overfitting of the data. For classification tasks using neural networks, the number of nodes in the output layer is equal to the number of possible classes. This is because using the *softmax* activation function creates a probability distribution across these different possible classes, and the one with the highest probability is selected. Here as well, we have 5 nodes in the last layer to represent the 5 different possible values.

Chapter 6

Legal, Social, Ethical and Professional Issues

6.1 Ethical Issues

The project extracts and eventually analyses content posted by users on social media. This may seem to be riddled with ethical concerns because of social media data being a controversial issue today and something that is under the constant scrutiny of the regulators. However, there are no ethical or legal challenges that the project faces. This is mainly because it simply uses the textual content that users willingly post on social media. As such, the data collected here is publicly available and therefore has no privacy concerns. It applies theoretical models to understand the content better and does not use any other source of information of the user. In no way is it capable of affecting a user's behaviour on social media or manipulating users into doing something. The eventual use of the analysis should also not be an issue from a professional, legal perspective because there is a manipulation of user data. It simply utilises an automated understanding of the reasoning behind what is publicly said. Throughout this project, we ensure that in no way can any segment of the work be used for malicious purposes.

6.2 British Computing Society Code of Conduct & Code of Good Practice

The British Computing Society has laid out a Code of Conduct and Code of Good Practice to provide a guidance framework when working in Information Technology (IT). We strive to abide by them during all work done as part of this project.

Chapter 7

Evaluation

7.1 Requirements

Must

– Setup connection with the Twitter API in order to get data.

- **Achieved by setting up a Developer Account and accessing the associated authentication tokens.**

– Extract tweets (under the limits provided by the Twitter API) when needed.

- **Achieved using Tweepy and the `api.search` endpoint.**

– Query and Extract tweets using hashtags or other search parameters.

- **Achieved using Tweepy and the q parameters of `the api.search` endpoint.**

– Store the extracted tweets in an appropriate file format.

- **The extracted tweets are stored in a `csv` file using the csv library. For instance `new_vaccine.csv` file.**

– Identify and present a structured pipeline of the solution.

- **The structured pipeline is presented in Chapter 3, section 3.1**

– Identify and use a suitable argumentation model.

- **The chosen argumentation model is presented and discussed in Chapter 3, section 3.1**

- Parse through the tweets file, and prepare and preprocess the data for the next step in the pipeline.

- **Achieved by using the `pandas` library, and then preparing the data as mentioned in Chapter 3.**

- Consider the ethical and professional issues, and practice good code.

- **The ethical issues are discussed in Chapter 6**

Should

- Extract real-time tweets on search parameters (hashtag or search queries)

- **Achieved using Tweepy, the `Cursor` feature of the Twitter API, and `the api.search` endpoint.**

- Present an in-depth explanation of the approach taken

- **The design choices are discussed in Chapter 3.**

- Train and use a classification model for argument identification.

- **Achieved by building a Support Vector Classifier as presented in Chapter 4, section 4.3**

- Train and use a classification model for component detection.

- **Achieved by building a Recurrent Neural Network model as presented in Chapter 4, section 4.4**

- Create an annotated data set for Twitter.

- **Achieved by creating a data set of 1000 tweets, as presented in Chapter 5**

Could

- Compare the performance of various supervised and unsupervised classification techniques on the problem

- **This is achieved by proposing different models in Chapter 3, and then comparing the performances in this chapter.**

– Compare the performance of different features on the training data.

- **This is partly achieved by comparing different features in the first stage of the pipeline.**

7.2 Model Performance

Argument Identification

The first stage of the model deals with identifying arguments in tweets. Towards this we implemented a Support Vector Classifier. Along with this, as means to provide a comparison, we built an ensemble of three classifiers. The performance of both these approaches is discussed here. A different set of features have also been tested, and the performance of each of them are presented.

To gauge the performance of the model, a set of baseline scores are determined. Two common baseline scores are evaluated. The first one (Baseline-1) chooses to predict the most frequent class every time. This is a standard baseline that is used in both binary and multiclass classification models. The second one (Baseline-2) makes predictions proportional to the class distribution in the training data.

Classifier	Accuracy	F1-Score
Baseline-1	0.53	-
Baseline-2	0.54	-
Support Vector Classifier	0.717	0.710
Ensemble Classifier(Voting)	0.717	0.707

Table 7.1: Performance of Classifiers (unigrams & bigrams)

As observed here, the proposed Support Vector Classifier's performance is much greater than both the baseline models. The training data, as discussed in Chapter 3, consists of 988 data points, with a class distribution of 524 (argumentative) and 464 (non-argumentative). Given the small size of training data, it is fair to say that the model performs satisfactorily.

The ensemble classifier also performs to a similar level. Even though it may seem that an ensemble voting classifier should outperform a standalone SVM, the results suggest otherwise. It is also important to note that there is no empirical evidence or mathematical foundation to prove that ensembles consistently outperform individual classifiers. One possible explanation for this could be the limited training data used. Having a larger data set could have led to the ensemble outperforming the SVM, but this is beyond the scope of the project and is a potential axis for future work.

Classifier	Features	Accuracy	F1-Score
Support Vector Classifier	unigrams	0.707	0.705
Support Vector Classifier	bigrams	0.570	0.473
Support Vector Classifier	bigrams, trigrams	0.525	0.344
Ensemble Classifier(Voting)	unigrams	0.680	0.670
Ensemble Classifier(Voting)	bigrams	0.621	0.561
Ensemble Classifier(Voting)	bigrams,trigrams	0.616	0.553

Table 7.2: Performance of different features

As mentioned in Chapter 4, the chosen features are unigrams and bigrams. The other features experimented with are *only bigrams*, *only unigrams* and *bigrams and trigrams*. The model gave the best results when using unigrams and bigrams together.

This model is also then tested on the data set presented in Chapter 3. The baseline performance and the model's performance on Twitter data are presented here. We obtained a 10-Fold cross-validation score of 0.56.

Classifier	Accuracy	F1-Score
Baseline-1	0.34	-
Baseline-2	0.48	-
Support Vector Classifier	0.64	0.45

Table 7.3: Performance of Classifiers (unigrams & bigrams) on Twitter Data Set

Current studies that have also used Support Vector Classifiers to identify arguments have also reported similar performance results. Park and Cardie (2015) use Support Vector Machines and features such as unigrams, bigrams and other experimental features. They present a resulting F1 score of 0.68. Eckle-Kohler et al. (2015) uses unigrams to train an SVM and obtain an F1 score of 0.71. Stab and Gurevych (2014) use SVMs to detect argument components in

persuasive essays. They use lexical features such as n-grams, verbs, and adverbs to obtain an F1 score of 0.72. Therefore, given the current state of the art, the model performs to a reasonable level.

Some additions could be made to the model to improve its performance. One possibility could be the use of extensive lexical features. Feature such as sentiment type, positive and negative emotions, the presence of special words could improve the model's accuracy.

Argument Component Detection

The final stage of the devised pipeline involves the detection of the individual argument components. Towards this, a Recurrent Neural Network is built with an LSTM model as the hidden layer. As mentioned in Chapter 3, we use pre-trained GloVe embeddings. These are available in 4 types of vectors - 25 dimensions, 50 dimensions, 100 dimensions, and 200 dimensions. We experimented with all of them to determine the best choice for this task. Here, we present the performance of the model along with various considerations about possible modifications.

In order to understand the model's performance better, we define two baseline models. The first one (Baseline-1) predicts the most frequent class every time. This is a standard baseline that is used in both binary and multiclass classification models. The second one (Baseline-2) makes predictions proportional to the class distribution in the training data.

It must be pointed here that the process of training neural networks is stochastic in nature. While evaluating the model, we ran the process for a number of iterations, and an estimate of these is presented here. At an estimated mean of all the iterations, we achieve an F1 score of 0.557.

No.	Word Embedding	Accuracy
Baseline1	-	0.520
Baseline2	-	0.200
1.	25d vectors	0.561
2.	100d vectors	0.616
3.	200d vectors	0.600

Table 7.4: Performance of RNN Model

The loss function of a neural network calculates the prediction error. The process of training a neural network ultimately aims to minimize this function. Hence, given how fundamental it is to a network, a loss function is a good indicator of how well training is done. Here, a graph that compares the loss function values for the training set and validation set is presented. The graph suggests that both the losses decrease at a decent rate. There is no sign of overfitting (at no point is the validation loss greater than training loss) or underfitting. Therefore, it reasonable to suggest that the training conducted is satisfactory. Other common metrics to quantify the performance of neural networks are accuracy and mean square error. Here, we show the accuracy of the model along with the baseline.

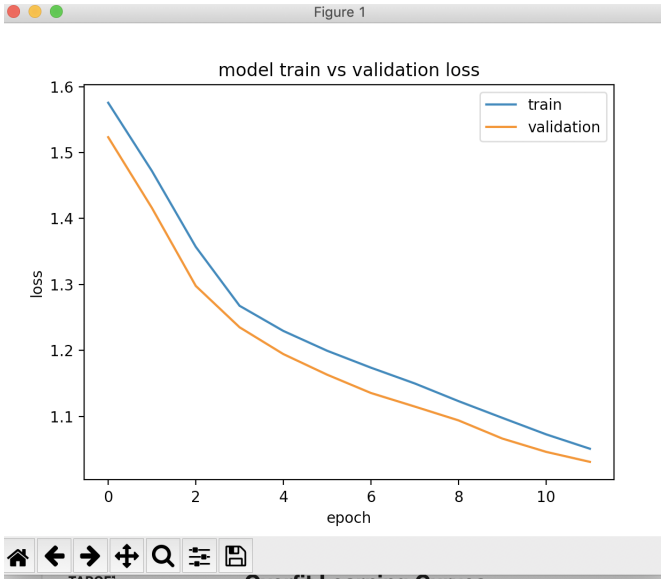

Figure 7.1: Training loss vs validation loss

As observed here, the proposed Recurrent Neural Network's performance is much greater than both the baseline models. The training data, based on the extended Toulmin model, consists of 996 data points. The extended Toulmin has 5 argument components. The class distribution is 169 - Claim, 527 - Premise, 217 - Backing, 17 - Refutation, 65 - Rebuttal. The data is highly imbalanced, and this directly increases the complexity of the classification problem. Furthermore, neural networks need a sufficient amount of data to understand each class's relationship, and therefore, this is a tough challenge. Given that they performed the best, we chose the 100-dimensional GloVe vectors. As is with any neural network model, choosing the correct set of hyperparameters for the model is vital for good performance. Here as well, different values of parameters were tried. For instance, while choosing the values for training epochs and batch size, it was important to ensure that the model is not trained to the point where it begins to overfit, which is a common problem in neural networks.

Existing work that uses neural networks in argument mining includes Mayer et al. (2020). They present results of using a Long Short Term Memory model along with a Conditional Random Field. Using different word embeddings, they obtain F1 scores ranging from 0.60 to 0.70. Niculae et al. (2017) propose an argument parsing model using structured Support Vector Machines and Recurrent Neural Networks. They obtain an F1 score of 0.63 on the RNN by using GloVe embeddings. Therefore, the model proposed here performs similarly to the state of the art approaches.

Even though our pipeline performs much better than baseline approaches and performs

similarly to other-related approaches within Argument Mining, there is a scope of improvement. For instance, we only test the first stage of the pipeline with actual labelled data. Therefore, testing the success of our pipeline on actual Twitter data is something that requires further work, and this may be considered a limitation of the project. Not only does it require extensively labelled data, but also a proper formal testing framework that can verify the predictions made by the model. This remains an open challenge in argument mining. Currently, the system is targeted towards the actual machine learning models. A good approach from a user perspective could have been the addition of a Graphical User Interface that allows users to view these results. However, we decided that this was out of the scope of the project.

Chapter 8

Conclusion and Future Work

Throughout this project, we have aimed to study and implement different aspects of argument mining. We dealt with the social media domain by proposing a pipeline that identifies argumentative tweets and then further detects the argumentative tweet's individual components. At every stage of the pipeline, technical issues were faced, and we managed to deal with some of them. Using the limited data available, we obtain performance results that outperform baseline approaches and are comparable to other state of the art models. Especially for Argument Identification, our model uses basic features and obtains scores at par with other approaches that use richer features. As is the case with any other Machine learning task, with more training data, our model's performance could potentially increase. As discussed on numerous occasions, the field of argument mining has a constant need for labelled data sets, more so, within the social media domain. Towards this, we created and presented a small data set of labelled tweets for argument identification. Even though small, this data set can help progress the state of argument mining. This poses a challenge for argument mining and the data on Twitter does not naturally lend itself to argument mining. It is raw, unstructured, and noisy. However, given the scale of the platform, the real-world impacts of the mining arguments on Twitter make it a very appealing endeavour.

There has been a sudden rise in the popularity and success of deep learning methods in several sections of AI in the past decade. This wide-ranging success suggests that deep learning methods can potentially solve other problems in AI. The deep learning approaches within Argument Mining have been scarce but nevertheless promising. Within social media, there have not been many approaches to argument mining that employ neural networks. Here, we explored the possibility of using recurrent neural networks in argument mining on social media. This approach

showed that even though the results did not hugely outperform other models, with the increase in the availability of larger data sets, neural networks could very well become a critical method in argument mining. This is especially true for social media because social media platforms generate huge volumes of data on a regular basis.

The field of argument mining has only started to grow, and with more focus, this has the potential to become a crucial part of AI technology systems. A potential line of future work would be to massively scale up and create an appropriately large labelled data set, using as a base the data set presented here. Similarly, another labelled data set, based on the extended Toulmin Model, can be created to detect argument components. It could possibly help advance the approaches presented here. During the creation of the data set, we noticed that some tweets refer to external sources that have high credibility or they themselves speak from a position of authority. This becomes the basis of their argumentation. Thus, an interesting future endeavour would be to explore ethos-based argument mining on Twitter that extracts arguments on the grounds of the speaker's credibility or authority. As pointed out earlier, this project has its limitations when it comes to testing raw Twitter data. A future endeavour could be an extensive testing regime that generates wide-ranging tweets and devises a framework to verify the predictions made by the model. As mentioned in the previous chapter, a good Graphical User Interface of the system could allow better analysis. Building a software product for argument mining and analysis presents a possible use-case of argument technology for commercial purposes.

Bibliography

Bengio, Yoshua, Patrice Simard, and Paolo Frasconi (1994). "Learning Long-Term Dependencies with Gradient Descent is Difficult". In: *IEEE Transactions on Neural Networks* 5.2, pp. 157–166. ISSN: 19410093. DOI: 10.1109/72.279181.

Bentahar, Jamal, Bernard Moulin, and Micheline Bélanger (2010). "A taxonomy of argumentation models used for knowledge representation". In: *Artificial Intelligence Review* 33.3, pp. 211–259. ISSN: 02692821. DOI: 10.1007/s10462-010-9154-1.

Bosc, Tom, Elena Cabrio, and Serena Villata (2016). "Tweeties Squabbling : Positive and Negative Results in Applying Argument Mining on Social Media". In: *Computational Models of Argument*.

Boser, Bernhard E., Vladimir N. Vapnik, and Isabelle M. Guyon (1992). "Training Algorithm Margin for Optimal Classifiers". In: *Perception*, pp. 144–152.

Cabrio, Elena and Serena Villata (2012). "Combining textual entailment and argumentation theory for supporting online debates interactions". In: *50th Annual Meeting of the Association for Computational Linguistics, ACL 2012 - Proceedings of the Conference* 2.July, pp. 208–212.

— (2018). "Five years of argument mining: A Data-driven Analysis". In: *IJCAI International Joint Conference on Artificial Intelligence* 2018-July, pp. 5427–5433. ISSN: 10450823. DOI: 10.24963/ijcai.2018/766.

Cocarascu, Oana and Francesca Toni (2017). "Identifying attack and support argumentative relations using deep learning". In: *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pp. 1374–1379. DOI: 10.18653/v1/d17-1144.

Dung, Phan Minh (1995). "On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games". In: *Artificial Intelligence* 77.2, pp. 321–357. ISSN: 0004-3702. DOI: https://doi.org/10.1016/0004-

3702(94)00041-X. URL: https://www.sciencedirect.com/science/article/pii/000437029400041X.

Eckle-Kohler, Judith, Roland Kluge, and Iryna Gurevych (2015). "On the role of discourse markers for discriminating claims and premises in argumentative discourse". In: *Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing* September, pp. 2236–2242. DOI: 10.18653/v1/d15-1267.

Habernal, Ivan, Judith Eckle-Kohler, and Iryna Gurevych (2014). "Argumentation mining on the web from information seeking perspective". In: *CEUR Workshop Proceedings* 1341. ISSN: 16130073.

Habernal, Ivan and Iryna Gurevych (2017). "Argumentation mining in user-generated web discourse". In: *Computational Linguistics* 43.1. ISSN: 15309312. DOI: 10.1162/COLI{_}a{_}00276.

Hochreiter, Sepp (1997). "Long Short-Term Memory". In: 1780, pp. 1735–1780.

Lippi, Marco and Paolo Torroni (2016). "Argumentation mining: State of the art and emerging trends". In: *ACM Transactions on Internet Technology* 16.2. ISSN: 15576051. DOI: 10.1145/2850417.

Llewellyn, Clare et al. (2014). "Re-using an argument corpus to aid in the curation of social media collections". In: *Proceedings of the 9th International Conference on Language Resources and Evaluation, LREC 2014*, pp. 462–468.

Mayer, Tobias et al. (2020). "Transformer-based Argument Mining for Healthcare Applications To cite this version : HAL Id : hal-02879293 Transformer-based Argument Mining for Healthcare Applications". In:

Mitchell, Thomas (1997). *Machine Learning*.

Mochales, Raquel and Marie Francine Moens (2011). "Argumentation mining". In: *Artificial Intelligence and Law* 19.1, pp. 1–22. ISSN: 09248463. DOI: 10.1007/s10506-010-9104-x.

Niculae, Vlad, Joonsuk Park, and Claire Cardie (2017). "Argument mining with structured SVMs and RNNs". In: *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*. Vol. 1. DOI: 10.18653/v1/P17-1091.

Park, Joonsuk and Claire Cardie (2015). "Identifying Appropriate Support for Propositions in Online User Comments". In: pp. 29–38. DOI: 10.3115/v1/w14-2105.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). "GloVe: Global vectors for word representation". In: *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*. DOI: 10.3115/v1/d14-1162.

Procter, Rob et al. (Dec. 2013). "Reading the riots: what were the police doing on Twitter?" In: *Policing and Society* 23.4, pp. 413–436. ISSN: 1043-9463. DOI: 10.1080/10439463.2013.780223. URL: https://doi.org/10.1080/10439463.2013.780223.

Simosi, Maria (2003). "Using Toulmin's framework for the analysis of everyday argumentation: Some methodological considerations". In: *Argumentation* 17.2. ISSN: 0920427X. DOI: 10.1023/A:1024059024337.

Slonim, Noam et al. (2021). "An autonomous debating system". In: *Nature* 591.7850, pp. 379–384. ISSN: 1476-4687. DOI: 10.1038/s41586-021-03215-w. URL: https://doi.org/10.1038/s41586-021-03215-w.

Stab, Christian and Iryna Gurevych (2014). "Identifying argumentative discourse structures in persuasive essays". In: *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pp. 46–56. DOI: 10.3115/v1/d14-1006.

Toulmin, Stephen (1958). *The Uses of Argument*. Cambridge University Press.

Toulmin, Stephen E. (2003). *The uses of argument: Updated edition*. DOI: 10.1017/CBO9780511840005.

Walton, Douglas, Christopher Reed, and Fabrizio Macagno (2008). *Argumentation Schemes*. Cambridge: Cambridge University Press. ISBN: 9780521897907. DOI: DOI:10.1017/CBO9780511802034. URL: https://www.cambridge.org/core/books/argumentation-schemes/9AE7E4E6ABDE690565442B2BD516A8B6

Walton, Douglas N (1996). *Argumentation schemes for presumptive reasoning*. English. Mahwah, N.J.: L. Erlbaum Associates. ISBN: 080582071X 9780805820713 0805820728 9780805820720.

Weinberger, Armin et al. (2007). "Scripting Argumentative Knowledge Construction in Computer-Supported Learning Environments". In: *Scripting Computer-Supported Collaborative Learning*. DOI: 10.1007/978-0-387-36949-5{_}12.