

https://dspace.org/

Testing machine learning algorithms for classifying authority in a hybrid institutional complex

Manoim, Rosa

2024
University of the Witwatersrand, Johannesburg
https://hdl.handle.net/10539/45756
Manoim, Rosa . (2024). Testing machine learning algorithms for classifying authority in a hybrid institutional complex [Master`s dissertation, University of the Witwatersrand, Johannesburg]. WIReDSpace. https://hdl.handle.net/10539/45756

Downloaded from a DSpace repository. https://dspace.org/

TESTING MACHINE LEARNING ALGORITHMS FOR CLASSIFYING AUTHORITY IN A HYBRID INSTITUTIONAL COMPLEX

ROSA MANOIM

A research report submitted to the Faculty of Humanities, University of the Witwatersrand, Johannesburg, in partial fulfilment of the requirements for the degree of Master of Arts in eScience.

Completed on 27 February 2024, Johannesburg

Abstract

The growing diversity of institutions that make up Hybrid Institutional Complexes involved in global governance has meant growing masses of raw data. Although these forms of institutions are some of the most important contemporary governance bodies, that have not yet been adequately analysed in the literature. Annual Reports, meeting minutes, policy documents and Codes are constantly being produced and published by these institutions, but this data is not in a form useful for statistical analysis. The use of hand-coding techniques for textual data is extraordinarily time consuming, a problem that is exacerbated in a swiftly changing field where data collection and classification could easily fall behind the ongoing shifts in institutional collaboration.

In order to keep up with the increasing complexity of these global governance bodies, research methodology needs to evolve accordingly, and develop new ways of capturing information about these institutions. By harnessing machine learning algorithms and especially deep learning networks for classifying textual-data, social scientists are able to deepen their research, particularly by creating new, usable datasets from the output documents of the institutions they research. This report demonstrates how the output documents released by the institutions in the global private security governance institutional complex can be successfully classified by machine learning algorithms.

This research report focuses on developing, and then assessing the effectiveness of an automated text classification approach. It demonstrates how a deep neural network algorithm can classify textual data from the global private security governance complex with up to 90% accuracy compared to expert labelling of the texts. It further compares traditional machine learning models to deep learning models and finds that traditional models like the random forest algorithm can classify these texts with over 85% accuracy.

Keywords: Deep Learning; Machine Learning; Artificial Neural Network; Language Modelling; International Relations; Global Governance Complex; Hybrid Institutional Complex; Deference; Authority.

Declaration

I, Rosa Manoim (Student number: 725975) am a student registered for Masters of Arts (in e-Science). I hereby declare that this research report:

- Is my own unaided work except where I have explicitly indicated otherwise. I have followed the required conventions in referencing the thoughts and ideas of others.
- Is submitted in partial fulfilment of the requirement for the degree of Master of Arts in e-Science at the University of Witwatersrand, Johannesburg. It has not been submitted before for any other degree or examination at any other university.

Signature:  Date: 27 February 2024

Acknowledgements

The support of the DSI-NICIS National e-Science Postgraduate Teaching and Training Platform (NEPTTP) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NEPTTP.

This research would not have been possible without the guidance and support of my external supervisor, Dr David Lubinsky. Dr Lubinsky stepped in at a point when I had all this data and no way to deal with it and said: 'Why not just use machine learning'. Thank you for your patience with me, and your enthusiasm for this project. You reinvigorated my excitement at a time when I thought the task might be impossible.

A big thank you too, to Professor Rod Alence for introducing me to the world of R and for showing me that data science is compatible with international relations research.

Thank you to Dr Oliver Westerwinter for sharing his original dataset with me and for all the hours spent discussing the hand coding of the Hybrid Institutional Complexes data.

Thank you to my family; Melinda, Irwin, and Lily for your encouragement, help and edits.

Table of Contents

1. Introduction 8

2. Literature Review 10

2.1 Hybrid Institutional Complexes 10

2.1.1 Global Governance Complexes in Liquid Modernity 10

2.1.2 Relationships of Collaboration and Deference of Authority in Hybrid Institutional Complexes (HICS) 12

2.1.3 Gap: Automated Text Classification as a Solution to the Gap in Data on Deference of Authority in HICs 14

2.2 AI and Machine Learning 15

2.2.1 Text classification and Natural Language Processing (NLP) 15

2.2.2 Artificial Intelligence, Machine Learning and Deep Learning 17

2.2.3 The Shift from Traditional Machine Learning to Deep Learning 20

2.2.5 How Do They Work: Neural Networks Architectures 24

2.2.6 Traditional Machine Learning 27

2.2.7 Comparison Literature 30

2.3 Bringing AI to International Relations Research 32

2.3.1 Politics, Text Data and Natural Language Processing 32

2.3.2 Literature Review Conclusions and Research Question 33

3. Methodology 35

3.1 Section Outline 35

3.2 Generating Data 35

3.2.1 Raw Data and Coding Scheme 35

3.2.2. Hand-Coding Problems and Machine Learning as a Potential Solution 38

3.3 Splitting Data into Useable Subsets 39

3.3.1 Training and Validation Splits 39

3.3.2 United Nations (UN) Data Focus 39

3.4 Language and Packages 41

3.5 Preparing the Data for Input into Machine Learning Algorithms 41

3.5.1 Text Pre-Processing 41

3.5.2 Indexing, Tokenization and Standardisation 42

3.5.3 One-hot Encoding Labels 44

3.6 Neural Network Models 45

3.6.1 Model Design and Rationale....45
Model 1....46
Model 2....48
Model 3....51
Model 4....52
Model 5....53
Model 6....54
Model 7....55
Model 8....57
Model 9....58
Model 10....59
Model 11....60
Model 12....62
Model 13....63
Model 14....64
3.7 Traditional Models....66
3.7.1 Random Forests....66
3.7.2 Decision Trees....67
3.7.3 Naïve Bayes....68
4. Findings....70
4.1 Table of Findings....70
4.2 Description of Findings....72
4.2.1 Variable Importance....74
4.2.2 Class Specific Analysis....77
4.3 Summary of Findings....80
5. Limitations of the Study and Proposals for Future Work....80
6. Conclusions....81
References....82
Appendix A....88
R code used in this research report....88

Tables:

Table 1 Institutional Deference Coding Scheme (Pratt 2018, Supplementary Material). 37

Table 2 Example of one hand-coded text in the data table....38
Table 3 Hand-coded sample: Count for each category of reference to the UN....40
Table 4 Re-coded Data with Four Classes of Deference Types....40
Table 5 Example: Labeled Text Row 1 259....40
Table 6 Most Common Words and Their Frequencies....44
Table 7 Database Text with Stop Words and Punctuation Removed Comparison to Original Text....44
Table 8 One-Hot-Encoded Texts Example....45
Table 9 Naive Bayes Predictions Confusion Matrix....69
Table 10 Summary of Models 1-5....70
Table 11 Summary of Models 6-11....71
Table 12 Summary of Models 12-14 and Traditional Models....72
Table 13 Model 12 Architecture and Hyperparameters....74
Table 14 Decision Tree Primary Split Variables and Actual Terms....75
Table 15 Decision Tree Variable Importance: Actual terms....75
Table 16 Random Forest Variable Importance and Actual Terms....76
Table 17 Random Forest model training data Confusion Matrix and Class Error....77
Table 18 Actual Class Labels for Validation Data Set....78
Table 19 Decision Tree Confusion Matrix....78
Table 20 Random Forest Confusion Matrix....78
Table 21 Model 12 Confusion Matrix....79
Table 22 Model 2 Confusion Matrix....79

Figures:

Figure 1 Locating Deep Learning in the AI Umbrella. 19
Figure 2 Image from Palanivinayagam et al (2023), depicting different classification algorithms performance on various datasets....27
Figure 3 The original decision tree from Breiman et al (1984, 2). 28
Figure 4 Keras Model 1 Loss and Accuracy....48
Figure 5 Keras Model 2 Loss and Accuracy....50
Figure 6 Keras Model 3 Loss and Accuracy....52
Figure 7 Keras Model 5 Loss and Accuracy....53
Figure 8 Keras Model 6 Loss and Accuracy....55
Figure 9 Keras Model 7 Loss and Accuracy....56
Figure 10 Keras Model 8 Loss and Accuracy....58
Figure 11 Keras Model 9 Loss and Accuracy....59
Figure 12 Keras Model 10 Loss and Accuracy....60
Figure 13 Keras Model 11 Loss and Accuracy....61
Figure 14 Keras Model 12 Loss and Accuracy....63
Figure 15 Keras Model 13 Loss and Accuracy....63
Figure 16 Keras Model 14 Loss and Accuracy....65
Figure 17 Random Forest Variable Importance Plot (Variable index number on y axis)....76

1. Introduction

From health to climate change, cyberspace to intellectual property, terrorism and security, the majority of global politics today are no longer governed by individual institutions or even interstate-regime complexes (Abbot and Faude 2021 and Eilstrup-Sangiovanni and Westerwinter 2021). These issues that transcend the borders of a single nation state are increasingly "governed by overlapping and interacting sets of formal and informal institutions constituted by varying mixtures of state and non-state actors" (Westerwinter 2022, 2). The term 'hybrid institutional complexes' (Abbot and Faude 2021) describes these diverse global institutional forms that are currently governing most, if not all, issue areas in the world. Within these governance complexes, institutions may conflict with each other or overlap in different ways such as membership, function, and issue area. Institutional overlap in a regime complex may cause problems in governance. Research shows that states or members might engage in "'forum shopping' to avoid complying with international rules they do not like," which they can do "by exploiting gaps in regulatory authority, or by claiming the absence of a clear global standard on a particular issue" (Pratt 2018, 565). This growth in institutional complexity is therefore likely to have a profound impact on the direction of global governance (Abbot and Faude 2021 and Eilstrup-Sangiovanni and Westerwinter 2021) as it may affect compliance with international standards. It is therefore necessary for scholarship to respond to this phenomenon and develop ways to understand these complexes as well as the interactions and relationships between the institutions that constitute the complexes.

Can machine learning help us to better understand these structures and relationships? Chollet et al (2022) explain machine learning as "searching for useful representations and rules over some input data, within a predefined space of possibilities, using guidance from a feedback signal. This simple idea allows for solving a remarkably broad range of intellectual tasks, from speech recognition to autonomous driving".

In a world of increasing complexity and increasing amounts of data as well as the constant improvements to technology and communication systems, research methods which can harness this technology and make use of this data may offer us new knowledge on issues that we do not yet sufficiently understand. Goodfellow et al (2016) argue that "machine learning is the only viable approach to building AI systems that can operate in complicated real-world environments".

Deep learning is a type of machine learning technique that "achieves great power and flexibility by representing the world as a nested hierarchy of concepts, with each concept defined in relation to

simpler concepts, and more abstract representations computed in terms of less abstract ones" (Goodfellow et al, 2016). This "hierarchical architecture" (Zulqarnain et al, 2020) means that the output of one layer can be the input in a higher layer, which allows the model to process low-level features (like word vectors) into higher level or more abstract features. Building concepts through layers and layers built on top of one another in this way leads to a deep network of knowledge, i.e. deep learning. Through 'building complex concepts out of simpler concepts' (Goodfellow et al, 2016) these algorithms can be more powerful at feature representation than other machine learning algorithms (Zulqarnain et al, 2020). Deep learning algorithms have been described by data scientists as "one of the best-performing machine learning algorithms" (Maslej-Krešňáková et al 2020). They receive particular attention for their successes in machine-perception problems in particular in the fields of computer vision and speech recognition (Maslej-Krešňáková et al 2020, Chollet et al 2022, Ballard et al 1983, Berner et al 2023).

The disconnect between the social sciences and mathematical and computer sciences, especially in the field of machine learning, indicates a potential gap which, if addressed successfully, could greatly benefit further social research. The success of machine learning algorithms for classifying or analysing text-data, could mean that scholars of international relations and other social-sciences could potentially draw on these methods to deepen their research, particularly for creating new, usable datasets derived from the output documents of the institutions they research.

The structures that we study in the social sciences are becoming increasingly noisy, 'messy', complex, as are the intellectual questions we are exploring about them. It is therefore harder to rely on existing databases which quickly become outdated. There is a need to produce more user-friendly data on these real-world phenomena, but the hand-coding approach that is usually used is too slow and costly to keep up with the floods of raw data that the internet makes available. An automated classification approach can produce the crucial data needed to better understand our world.

This research tests this theory by using machine learning techniques to classify text data produced by organisations in a global governance complex and assessing the accuracy of different machine learning algorithms.

2. Literature Review

This section reviews the existing literature to ground the discussion, provide a theoretical framework, and place the research in its academic context including identifying gaps in the literature. It is divided into three subsections: Section 2.1 provides an overview of the literature on Hybrid Institutional Complexes and introduces the literature gap which this research hopes to fill through the application of machine learning methods. Section 2.2 provides an overview of the history and literature on AI and Machine Learning and gives a detailed discussion on different machine learning models. Section 2.3 brings these two discussions together by proposing AI solutions to social science research problems and presenting the research question and rationale.

2.1 Hybrid Institutional Complexes

2.1.1 Global Governance Complexes in Liquid Modernity

Historically, scientific research has often worked to study and understand the world through condensing, simplifying, creating rigid generalisable categories. Both data science and the study of international relations have often looked systematically at 'states', or 'organisations' as solid objects of study. However, our contemporary era could more realistically be seen as in fact, lacking in 'solid' categories that can be studied as they are (Bauman, 2000). Rather, actors, organisations, and even state-forms, can no longer be imagined as stable categories for analysis, and could better be understood as systems in flux which are constantly shifting and interacting in unanticipated ways. Complexity needs to be incorporated into scientific research. When international organisations are our objects of study, it is useful to move towards seeing them not as limited, definable singular objects, but as highly complex, ever-changing systems, networks, or *complexes* affected by, and affecting their various components. The growing complexities of (and between) these international organisations are examined by scholars of International Relations including Eilstrup-Sangiovanni and Westerwinter (2021), Abbott and Faude (2021), Clark (2021), and Pratt (2018).

Bauman (2000), describes our current era as "liquid modernity", as opposed to the 'solidity' that structuralist scholars understood of the pre-globalised capitalist world order of recent history.

> *"Fluids travel easily. They 'flow', 'spill', 'run out', 'splash', 'pour over', 'leak', 'flood', 'spray', 'drip', 'seep', 'ooze'; unlike solids, they are not easily stopped - they pass around some obstacles, dissolve some others and bore or soak their way through others still... The extraordinary mobility of fluids is what associates them with the idea of 'lightness'. There are liquids which, cubic inch for cubic inch, are heavier than many solids, but we are inclined*

nonetheless to visualize them all as lighter, less 'weighty' than everything solid. We associate 'lightness' or 'weightlessness' with mobility and inconstancy: we know from practice that the lighter we travel the easier and faster we move. These are reasons to consider 'fluidity' or 'liquidity' as fitting metaphors when we wish to grasp the nature of the present, in many ways novel, phase in the history of modernity." (Bauman 2000, 2)

It is in this context of liquid modernity that scholars are trying to better understand hybrid global governance systems.

Under the contemporary conditions of globalised capitalism and privatisation, it is not 'the state' but rather diverse international regime complexes that are increasingly seen as responsible for governance of global issues that transcend state borders, including the issue of private security.

Safety, in the form of 'security' is an important social and political issue that has been central to societies around the world, albeit in different iterations, throughout history (Bauman, 2000). Earlier history shows communities responsible for protecting themselves from other communities/tribes/empires through allegiances, weapons, and settlement patterns. In more recent history, under the Westphalian nation state world-order, we understood states to be the main provider of security, particularly in the form of violence-based security: armies, police, wars, and negotiations (Weber, 1919). Now, however, security is understood to be no longer limited to the public sphere and is becoming an exponentially privatised commodity. States no longer are thought to have the monopoly over security issues, nor can they regulate private security alone (Abrahamsen and Williams, 2007). The rise of globalised private security indicates that analytically we must move away from traditional assumptions about distinct boundaries (Bauman 2000, Abrahamsen and Williams 2007) i.e. those between 'the state' and 'the private sphere', because "the authority of global private security companies cannot be contained within the traditional distinctions of inside–outside, global–local or public–private, but requires the dissolution of the 'state–territory–authority' marriage". Instead, we must develop a more complex analysis that considers "new networks and interactions between state and non-state actors" (Abrahamsen and Williams 2007, 238).

Therefore, this research sets out to do this work of examining new networks and interactions between the diverse range of actors in a global governance complex.

Abbott and Faude (2021, 2) argue that "most issue areas in world politics today are governed neither by individual institutions nor by regime complexes composed of formal interstate institutions. Rather, they are governed by 'hybrid institutional complexes'". These Hybrid Institutional Complexes (HICs) are comprised of "heterogeneous interstate, infra-state, public–private and private transnational institutions, formal and informal."

The Hybrid Institutional Complex is not simply a collection of institutions but rather it is a *complex* in which the range of institutional forms that it contains have some authority to govern, address common problems, and interact with and affect each other. It is these interactions that make the complex a "governance system" as opposed to a group of institutions (Abbot and Faude, 2021).

Eilstrup-Sangiovanni and Westerwinter (2021) coined the term Global Governance Complex as a broad-reaching concept that captures the variety of international organisations and actors that work to govern a particular policy issue. They define the Global Governance Complex as "a system of governance composed of at least three international or transnational institutions or actors whose mandates, functions and memberships overlap, and that jointly address a specific policy problem" (Eilstrup-Sangiovanni and Westerwinter, 2021).

The variety of institutions and conglomerates that are included in the idea of a Global Governance Complex are understood by Abbott and Faude (2021) as being on a continuum, distinguishable by the level of diversity of the institution in question. Abbott and Faude (2021) see regime complexes comprised of homogenous formal intergovernmental organisations as being a low-diversity global governance complex, with the more diverse complexes that contain a greater variety of institutional forms at the higher end of the continuum. These they define as Hybrid Institutional Complexes (Abbott and Faude, 2021). The level of diversity, and thus complexity, in a global governance complex affects how the complex operates, and thus how we can understand it.

2.1.2 Relationships of Collaboration and Deference of Authority in Hybrid Institutional Complexes (HICS)

The global private security governance complex is identified by Westerwinter (2022) as a 'prime example" of an emerging Hybrid Institutional Complex made up of a heterogenous range of institutions/actors that can influence global politics. As a relatively new institution, that is increasing in significance over time due to the global trend of privatizing security functions, I find it to be a particularly useful case for developing insight into Hybrid Institutional Complexes specifically and contemporary global governance generally. The global private security governance complex that makes up my dataset has 11 institutions that have been active sometime between 2000-2020, that all "seek to regulate the behaviour of private security companies at the global level" (Westerwinter 2022). Westerwinter (2022) describes it as "a relatively young complex in which governance is fraught with uncertainty about the problems at hand and the distributional implications of different solutions". "The institutions in the complex have intersecting mandates and protrude into each other's functional governance spaces. This

provides the basis for inter-institutional collaboration but also holds the potential for competition and conflict" (Westerwinter 2022, 4).

One of the major features of Hybrid Institutional Complexes according to the literature, is the likelihood of overlap and conflicting governance from the multiple institutions in the complex (Eilstrup-Sangiovanni and Westerwinter 2021, Abbott and Faude, 2021). Institutions may try to mitigate these potential global governance problems by recognizing the authority of an institution in that complex and cooperating with that institution in a collaborative way. Authority, in terms of inter-organisational ordering structure or hierarchy is an important part of complexity management, and it affects governance in institutional complexes through shaping whether and how collective outcomes can be achieved and how solutions to problems can be found and implemented (Eilstrup-Sangiovanni and Westerwinter, 2021). "Deference occurs whenever one [International Organisation] accepts another organization's exercise of authority" (Pratt, 2018). Deferring authority to a different organisation could mitigate the governance issues of having overlapping and often contradicting rules for the same jurisdiction, and could mitigate the costs of "inefficient duplication" of work and outputs (Pratt 2018).

Pratt (2018) argues that institutional deference is a mechanism for coordination amongst institutions in a regime complex. He provides the example of the UN Security Council members, who rather than creating their own rules, deferred to the rules from the Financial Action Task Force (FATF) in order to prevent potentially conflicting sets of rules. Thereafter, other international organisations followed suit and also deferred to the FATF. Much existing scholarship on inter-organisational collaboration show that an authoritative organisation as a collaboration partner is an asset (Brosig 2015, Abbot et al 2015, Hale and Roger 2014, Oberthur and Stokke 2011).

Clark (2021) in *Pool or Duel? Cooperation and Competition Among International Organizations* explores the collaboration between international organisations by investigating the reasons that these organisations might 'pool' together their resources rather than compete amongst themselves. Clark (2021) examines the organizational costs and payoffs of sharing various types of resources. This work also engages in questions of authority relationships, and the author argues that in fact "the relations between the most powerful stakeholders in each [international organisation] should dictate whether [international organisations] pool or duel" (Clark 2021,2). His theoretical focus on the organisations' staff and their interests and alignments as pivotal in dictating the direction of the organisations, is overlooked by many of the other works in this cannon.

Westerwinter (2022) examines the private security governance regime complex and provides statistical evidence on how collaboration amongst the institutions in the complex is shaped by informal hierarchy and functional differentiation. His paper develops useful indicators to map collaboration in a global governance complex. His research shows that (informal) hierarchy in a regime complex will "promote the

overall level of collaboration in a complex as well as collaboration between pairs of institutions". He further argues that institutions with functional similarity in hierarchical regime complexes are more likely to collaborate than in non-hierarchical complexes. Westerwinter (2022) develops a methodology for examining three different types of collaboration, namely endorsements, information exchange, and institutionalised partnerships. He also develops a methodology for identifying the different governance tasks undertaken by each institution.

Whilst Westerwinter (2022) looks at authority through the idea of power hierarchies (Henning and Pratt 2021) to evaluate collaboration in a global governance complex, the concept of authority levels is explored differently in the 2018 work by Pratt who focuses much more specifically on cases of organisational 'deference' to the authority of other organisations in a regime complex.

Pratt (2018) examines deference in three different regime complexes, all of which would likely be defined as low-diversity complexes by Abbott and Faude (2021). Pratt's (2018) work measures the use of deference as it is formally documented in the policy documents of the relevant international organisations in the regime complex. He explores whether deference is used by international organisations as a tool for managing overlapping jurisdictions, and dividing labour accordingly. He sheds light on the politics that determine who defers to whom, and why. He looks at two different possible explanations for institutional deference: 'functional efficiency' and 'member state power' (Pratt 2018).

Abbott and Faude (2021) discuss the idea of 'informal hierarchy' within Hybrid Institutional Complexes and engage with the Pratt (2018) paper in particular, as one of the first works that looks at hierarchy beyond states, rather, within institutions. Drawing from Pratt's empirical research, they argue that that greater institutional diversity in a Hybrid Institutional Complex will mean greater informal hierarchy structures in that complex.

While Pratt has studied the cooperation within low diversity regime complexes (2018), scholars of international relations have stressed the need for more detailed engagement with how international organisations in hybrid complexes mitigate conflict through collaboration, whether there is deference of authority, and what the implications of this might be (Abbot and Faude 2021, Ellistrup and Westerwinter 2021).

2.1.3 Gap: Automated Text Classification as a Solution to the Gap in Data on Deference of Authority in HICs

Whilst Pratt relies on statistical software to find the references to the specific organisations within each document, he uses hand coding to classify each reference. Similarly, Westerwinter (2022) uses hand

coding for his cooperation variable, as does Clark (2021) for his 'co-financing' and 'information sharing' variables which he explained were hand-coded based on his "reading of thousands of program documents, press releases, and other pieces of [International Organisation] legislation covering the period between 1944 and 2018".

Although the literature indicates the importance of understanding the relationships between the institutions within global governance complexes, these have not yet been sufficiently studied, and it is not yet known in literature if/how the organisations in a hybrid institutional complex recognise and defer authority to one another.

All the literature I have reviewed speaks to the increasing production of textual material in these international regime complexes, but none have yet used machine learning techniques to make this data available for statistical analysis. The use of hand coding techniques for textual data is extraordinarily time consuming, a problem that is exacerbated in a swiftly changing, 'fluid' (Bauman 2000), field, where data collection and classification could easily fall behind the fast-paced shifts in institutional collaboration.

Using supervised machine learning techniques to classify institutional output documents from the institutions in a hybrid governance complex according to whether, and how strongly, the institutions defer authority to one another, can if successful, open avenues for further understanding into these significant relationships.

2.2 AI and Machine Learning

The literature indicates that automatic text classification techniques have the potential to be extremely helpful in extracting and compiling useful data for deeper research into relationships within the global private security governance institutional complex. The next section of my literature review lays out what I mean by automatic text classification techniques, and engages with some of the key historical and contemporary theoretical discussions around machine learning, and contextualises some of the different text classification models.

2.2.1 Text classification and Natural Language Processing (NLP)

The continuous growth of internet usage results in an increase in the production of data (Dogru et al, 2021). Ikonomakis et al (2005) describe automatic text classification as a "necessity" because of the extremely large quantities of textual documents produced. Often the data are heterogenous, particularly when they are collected from a variety of sources such as websites, news, and organisational reports and therefore have different formats, vocabularies and styles (Ikonomakis et al, 2005).

Whilst it is a difficult and time-consuming task to classify all this data by hand, the fields of natural language processing (NLP) and machine learning provide for automated classification of data. Natural Language Processing refers to human languages (like English) not machine languages (like R or Python). Unlike machine languages, natural languages are "messy- ambiguous, chaotic, sprawling, and constantly in flux" (Chollet et al 2022, 335).

Yet as social scientists understand, it is through language that we learn about the world. "Language is how we store almost all of our knowledge. Our very thoughts are largely built upon language. However, the ability to understand natural language has long eluded machines" (Chollet et al 2022, 335). Therefore, Natural Language Processing tools like those used in this paper are not designed to train computers to *understand* language but rather to use computers to "ingest a piece of language as input and return something useful" (Chollet et al 2022). In this case 'something useful' refers to a classification task that asks: does this text from organisation *i* refer to organisations *j,k,* and/or *l* and if so is this reference an example of deference of authority or simply a passing reference?

Classification tasks involve instructing a computer program to determine the category to which a given input belongs. The typical approach to address such tasks is to instruct the learning algorithm to generate a function: $f: \mathbb{R}^n \rightarrow \{1, \ldots, k\}$

When $y = f(x)$ the model assigns an input, represented by the vector x, to a specific category identified by the numeric code y. Variations of classification tasks exist, such as cases where f outputs a probability distribution across classes. A classic example of a classification task is object recognition, where the input is an image, usually characterized by a set of pixel brightness values, and the output is a numeric code identifying the object depicted in the image (Goodfellow et al, 2016).

Text classification is the process of analysing textual data and separating it according to predefined classes (Dogru et al, 2021). When text classification is done entirely by hand, it is extremely time consuming and costly. It is also understood as being highly susceptible to error because of lack of domain knowledge and human error by the coders (Palanivinayagam et al, 2023). The invention of machine learning models such as Naïve Bayes, random forest and the Support Vector Machine brought about 'a huge revolution in text classification' (Palanivinayagam et al, 2023), where much manual work was replaced by these algorithms because of the time and cost reduction as well as because high levels of classification accuracy were proved in early research (Palanivinayagam et al 2023, Ombabi et al 2017, Ikonomakis et al 2005). Since the early days of machine learning models, automatic text classification models have continuously been studied, improved, and optimized (Palanivinayagam et al 2023, Goodfellow et al 2016). However, Ikonomakis et al (2005) showed almost a decade ago that the effectiveness of automatic text classifiers was not faultless and still needed improvement. Researchers

are now looking towards deep learning models to improve performance of machine learning algorithms doing text classification tasks (Ombabi et al, 2017).

"With the development of big data technology, the study of international relations texts has begun to attract the attention of academic circles, but the accuracy of text classification for international relations needs further research." (Tao, 2022). This research paper responds to this call by assessing the accuracy of different classification algorithms on international relations data.

2.2.2 Artificial Intelligence, Machine Learning and Deep Learning

Although Artificial intelligence (AI) is currently receiving a lot of attention in media and scholarship, it is not a new invention. Early AI successes include automating chess and checkers playing, and automating driving, but programmers found difficulties in hand-coding a 'knowledge base' for computers to rely on to solve problems (Lenat and Guha, 1989). The difficulties that arose from having to hard-code knowledge for the AI algorithms to rely on, brought about the need to develop a system for the AI to "acquire its own knowledge by extracting patterns from raw data" (Goodfellow et al, 2016). This is called machine learning. Examples of classic machine learning algorithms include logistic regression, Naïve Bayes, and decision trees. However, these traditional machine learning algorithms generally cannot solve highly complex tasks, or tasks that are hard to explain by formal rules:

*"In the early days of artificial intelligence, the field rapidly tackled and solved problems that are intellectually difficult for human beings but relatively straight-forward for computers—problems that can be described by a list of formal, mathematical rules. The true challenge to artificial intelligence proved to be solving the tasks that are easy for people to perform but hard for people to describe formally—problems that we solve intuitively, that feel automatic, like recognizing spoken words or faces in images" (*Goodfellow, et al 2016).

The solution to this issue of 'intuitive' problems being too complicated for traditional AI can be found in 'deep learning' techniques. The premise of deep learning is that computers can "learn from experience and understand the world in terms of a hierarchy of concepts, with each concept defined through its relation to simpler concepts" (Goodfellow et al, 2016). Humans therefore don't need to specify every piece of knowledge that the computer needs because it can automatically learn what it needs to from 'experience' or exposure to the data (Goodfellow et al 2016, Aggarwal 2018).

Deep learning can be summarised according to Berner et al (2023, 2) as *"techniques where deep neural networks are trained with gradient-based methods".* Deep Learning is a specific type of machine learning which usually involves layers of geometric transformations stacked on top of one another. Deep

learning has been considered to have achieved 'unprecedented technical success' compared with other forms of machine learning (Chollet et al, 2022) due to the breakthroughs that have recently been achieved (Grohs and Kutniok, 2023) especially in areas in which computers have historically been relatively unsuccessful, in particular 'perception' issues, like extracting useful data from videos (Maslej-Krešňáková et al 2020). While it can be understood that deep learning is currently experiencing a massive hype and surge in public interest, Chollet et al (2022) predict that "the hype may (and likely will) recede, but the sustained economic and technological impact of deep learning will remain. …in the longer term, it will still be a major revolution that will transform our economy and our lives."

What we describe today as deep learning algorithms have developed through different iterations and names. Deep learning algorithms are classically referred to as *artificial neural networks* because historically deep learning was theorised in relation to how learning happens in the brain, in particular how neurons connect and feed into each other in order to 'learn' (Aggarwal 2018, Grohs and Kutyniok 2023, Rosenblatt 1958). Various arguments around this perspective exist in the literature (Goodfellow et al 2016, Aggarwal 2018, Grohs and Kutniyok 2023) and whilst current understandings of deep learning and neural networks have shifted away from the image of the human brain, the "neural perspective on deep learning" still holds some compelling arguments, such as the idea that for neural network algorithms, as within the brain, "having many computational units that become intelligent only via their interactions with each other" (Goodfellow et al, 2016). Modern deep learning, however, draws on a number of fields and no longer references much neuroscience, focusing instead on applied mathematics and information theory (Goodfellow et al, 2016). Chollet et al (2022) pose that "a more appropriate name would have been *layered representations learning or hierarchical representations learning*, or maybe even *deep differentiable models* or *chained geometric transforms*, to emphasize the fact that continuous geometric space manipulation is at their core".

I will use the terms neural networks and deep learning models somewhat interchangeably in this research.

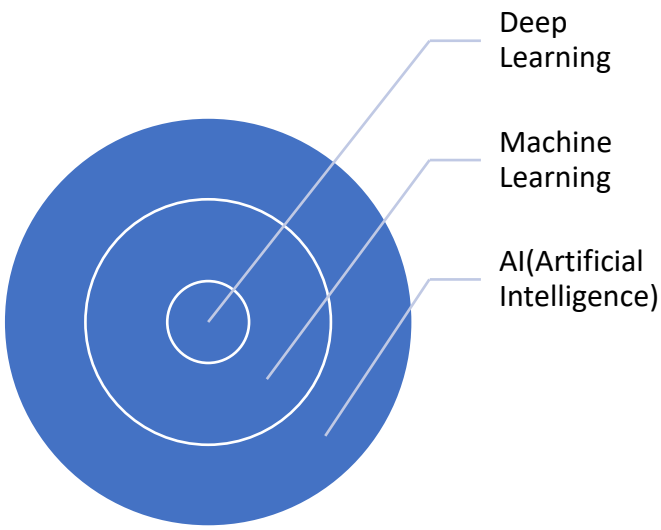


Figure 1 Locating Deep Learning in the AI Umbrella.

The image above illistrates that deep learning fits under the umbrella of machine learning which fits under the umbrella of artificial intelligence.

Artificial Intelligence (AI) can be summarized as "all attempts to automate human cognitive processes (Chollet et al 2022)". Machine learning is a subfield of AI that develops programs called 'models' by 'learning' from training data. In the 2000s machine learning became the most widely used type of AI, although it has existed for much longer (Chollet et al, 2022). AI today is associated with advertising, search engines, identifying speech and images, automating labour, and even assisting in medical diagnoses (Grohs and Kutyniok 2023, Berner et al 2023, Chollet et al 2022). A logistic regression algorithm can determine whether to recommend caesarean delivery (Mor-Yosef et al, 1990) or a simple Naive Bayes algorithm can separate legitimate e-mail from spam e-mail (Sahami et al 1998). Deep learning techniques have helped pharmaceutical companies predict how molecules will interact with each other in new drugs (Dahl et al 2014) and have assisted neuroscientist in their understanding of the brain (Yamins et al 2013 and Knowles-Barley et al 2014). Recent breakthroughs of deep learning algorithms include OpenAI's ChatGPT (Brown et al, 2020) and its subsequent development into GPT-4 (OpenAI, 2023).

> *"Rather than programmers crafting data-processing rules by hand, could a computer automatically learn these rules by looking at data?" (Chollet 2018).*

Referring to machines that 'learn', computer scientists mean that rather than explicitly programming the system, a machine learning system is 'trained' by being presented with relevant examples. In these examples it finds a statistical structure that allows it to determine rules for automating the task (Chollet 2018). According to Goodfellow et al (2016), "from a scientific and philosophical point of view, machine learning is interesting because developing our understanding of it entails developing our understanding of the principles that underlie intelligence".

There are a number of different types of machine learning; including unsupervised, and reinforcement machine learning, but this paper focuses on supervised machine learning which is most useful for classification tasks (Chollet, 2018). In supervised machine learning we input a set of data which are already labelled i.e., we 'train' the machine using a subset of data with answers we have hand-coded, and the machine learning algorithm then 'learns' from the labelled data what the rules are for labelling that data. We then give the machine a way to measure how successful the algorithm is at its task and adjust the algorithm accordingly (Chollet et al 2022, Aggarwal 2018).

Goodfellow et al (2016) rely on the 'improvement' focused definition for machine learning given by Mitchell (1997): *"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."*

AI algorithms generally rely on probability theory, which in its most basic sense means representing uncertainty in a mathematical way (Goodfellow et al, 2016). Probability theory can tell us how AI systems should reason so that we can design algorithms based on this theory, and we can also then use probability to analyse the AI systems that have been built. Information theory is also important for machine learning because it provides tools to enable us to quantify the amount of uncertainty in a probability distribution. "Machine learning must always deal with uncertain quantities and sometimes stochastic (nondeterministic) quantities" (Goodfellow et al, 2016). Uncertainty arises in machine learning models for several reasons including inherent stochasticity in the data, or incomplete data that cannot ever be fully observed or modelling that requires some information to be discarded (Goodfellow et al, 2016). Information theory is based on the simple idea that it is more informative to learn that an unlikely event has occurred than to learn that a likely event has occurred.

When designing the architecture of a machine learning algorithm, particularly a deep neural network, probability distribution needs to be taken into account, especially when datasets get increasingly large, and variables may be increasingly random (Goodfellow et al 2016).

2.2.3 The Shift from Traditional Machine Learning to Deep Learning

Early AI was successful at assisting humans with tasks that had clear mathematical rules that the algorithm could produce the 'answers' to. But artificial intelligence historically has not been so successful at solving the problems that are often subjective and intuitive for people but for which the rules are not easily articulated (such as identifying people in pictures) (Goodfellow et al, 2016). These sort of tasks, when done by humans, require years of subjective and intuitive knowledge and experience that we have gotten just from being in the world. In order for computers to output this sort of 'intelligence' they need a way of getting all the necessary 'knowledge' required for their task.

Deep learning is a potential solution for this problem:

> *"This solution is to allow computers to learn from experience and understand the world in terms of a hierarchy of concepts, with each concept defined through its relation to simpler concepts. By gathering knowledge from experience, this approach avoids the need for human operators to formally specify all the knowledge that the computer needs. The hierarchy of concepts enables the computer to learn complicated concepts by building them out of simpler ones". (Goodfellow et al 2016)*

Building concepts through many layers built on top of one another in this way leads to a *deep* network of knowledge, i.e deep learning. Through 'building complex concepts out of simpler concepts' (Goodfellow et al 2016) deep learning opens up the possibility of harnessing AI for processing information that is multi-dimensional, complex, and hard to define.

Although often thought of as 'new hype', neural networks are argued to have been developed soon after computers themselves in the late nineteen fifties (Aggarwal 2018, Chollet et al 2022) with Rosenblatt's *perceptron* algorithm: a development on the McCulloch and Pitts Neron which was invented in 1943 as an attempt to mimic a biological neuron with a linear model (Rosenblatt 1958), and soon after that, 'ADALINE' the adaptive linear element (Windrow and Hoff, 1960). ADALINE is still famous for its early use of stochastic gradient descent, an algorithm which remains in use in neural networks today (Goodfellow et al 2016). The *perceptron* (Rosenblatt, 1958) is famous for being one of the first models that demonstrated machine learning (Grohs and Kutniok 2023)- in that a simple algorithm could adapt its parameters, based on training data inputted into it, in order to improve its performance. However, it had limited applicability for non-linear problems, leading to some years of a decrease in research interest in neural networks and AI in general (Aggarwal, 2018).

The 'missing piece' getting in the way of AI's success was "an efficient way to train large neural networks" (Chollet et al, 2022). The shift that allowed AI to regain research interest and popularity was the application of the backpropagation algorithm to neural networks in the mid-1980s. This discovery, or rediscovery, can be attributed to several independent researchers including Kelley (1960), or Dreyfus (1962), but Goodfellow et al (2016) and Aggarwal (2028) attribute it in particular to Rumerlhart et al's (1986) work that used symbol embeddings which could later pave the way for more advanced natural language processing networks. However, the most successful demonstration of backpropagation that put AI back on track for research and development was a few years later with LeCun et al's (1989) work at Bell Labs on reading handwritten digits.

Rumelhart et al's (1986) early family tree model was one of the first recorded successful uses of back-propagation techniques and a layered neural network (Goodfellow et al 2016, Aggarwal 2018). This work used the idea of forming an embedding for a symbol that could be interpreted by the network. In 1990, Deerwester et al. used the same idea around embeddings, but they applied it to words in an early Singular Value Decomposition algorithm (Deerwester et al, 1990). Other early applications of neural network to language-tasks include Miikulainen and Dyer's work in 1991 which focused on modelling textual input as a sequence of characters (Miikulainen and Dyer, 1991) which remained popular for quite some time, arguably until the early 2000s when the focus of language modelling returned to word-level models such as Bengio et al's (2003) work which introduced neural language models that focused on producing interpretable word embeddings (similar to what is still used today). Goodfellow et al (2016) describe how these sorts of neural network models have increased in capacity from being able to represent a small set of symbols in the 1980s to the millions of words that contemporary neural networks can handle.

The success of AI is often defined by comparison to human performance, despite the understanding that "humans and computers are inherently suited to different types of tasks" (Aggarwal, 2018). As yet, AI for the most part has not yet been able to compare to human performance, other than in a few fields where AI has matched or even exceeded human capabilities, including game playing, image recognition and arguably also self-driving cars (Aggarwal 2018, Grohs and Kutyniok 2023).

Because it is seen as 'a task that is effortless for humans and many animals but challenging for computers", computer vision has been one of the most widely researched areas for deep learning problems (Ballard et al, 1983). The Convolutional Neural Network's (CNN) success with image recognition is the main accepted example of this. Automated image classification has in fact been one of the fields where deep learning first became successful and is today considered as being able to out-perform human classifiers (Berner et al 2023, Maslej-Krešňáková et al 2020). CNN architecture can be traced back to Hubel and Wiesel's (1959) work around the neurons in the visual cortex of a cat. Thereafter, in 1979, Fukushima developed the 'Neocognitron' an extremely successful use of the Convolutional Neural Network (Fukushima, 1980), which he developed without funding during the period referred to as the 'AI winter' (Goodfellow et al 2016, Aggarwal 2018, Chollet et al 2022) when AI was being overlooked for research funding because of its limitations.

Although deep neural networks have received much less scholarly attention for their uses in text classification and natural language processing tasks, the literature that does test this has been overwhelmingly positive (Zulqarnairn et al 2020, Ombabi et al 2017, Maslej-Krešňáková et al 2020, Chollet et al 2022, Berner et al 2023) and language processing is understood in the literature as one of the growing fields where 'impressive advances' are being made (Berner et al 2023).

In understanding the history of neural networks from the 1950s until now, it is important to note the shifts happening in the world during this period which affected artificial intelligence research and allowed for AI to become what it is today. The success and capabilities of the early neural networks (such as Hubel and Wiesel 1959, Roseblatt 1959, Windrow and Hoff 1960, Dreyfus 1962, Rumelhart et al 1986, etc) were limited by poor computing processing power, limited data, and the cost of computing.

Because deep learning is based on intelligence built by the interactions between a large number of neurons in a system, the size of the data is important. The increase in the size of the artificial networks is made possible only through high performance software and hardware and greater availability of data (Goodfellow et al 2016, Chollet et al 2022, Aggarwal 2018). "The age of 'Big Data' has made machine learning much easier because the key burden of statistical estimation—generalizing well to new data after observing only a small amount of data—has been considerably lightened" (Goodfellow et al 2016, 20). The Big Data era has meant that everything we do online, including shopping, clicking on a website, and communication is collected and stored. In 1999, Graphics Processing Units, (GPUs) were developed which allowed faster processing and a massively increased computational speed, which by 2011 had increased to an extent that deep neural networks came firmly back onto the map, achieving notable results in global competitions (Aggarwal 2018, Goodfellow et al 2016). It is only thanks to the increased availability of data and computational power that neural networks have been able to prevail over the limits of traditional machine learning algorithms (Aggarwal 2018).

Not only have GPUs enabled more efficient processing and data storage, but the increased computation speed has enabled more efficient teasing and tweaking of algorithms (Aggarwal 2018, Chollet et al, Goodfellow et al 2016). GPUs, originally sold for fast gaming graphics, are now flexible enough to be used for a much wider range of tasks, including statistical computing. In 2005, Steinkrau et al implemented a two layer fully connected neural network on a GPU and achieved impressive results, especially relating to speed (Steinkrau et al, 2005).

Together with GPUs, the rise of the internet, the increase in available storage, and 'incremental algorithmic innovations' (Chollet et al 2022) the continued experimentation and gradual advancement in AI research and technology described above have allowed for neural networks to develop into the highly competitive and useful algorithms they are today.

Although deep learning is just one of many AI applications, because of its "spectacular success" (Grohs and Kutniok, 2023) it "has singlehandedly brought about …[this] period of intense interest, investment, and hype in the field of AI" which we are presently experiencing (Chollet et al, 2022). However, because AI is generally understood as quite elusive to its lay users, there has been a growth in attempts to shift towards 'explainable artificial intelligence' namely AI systems that can be explained to human lay users and understood (Miller, 2019). The idea of the 'black box' is often used to describe machine learning

algorithms, in particular deep learning/neural network algorithms. This means that human users don't fully understand the workings of the models, why or how the model is able to do its task, only how well it performed. Miller (2019) shows that interpretability and explanation in AI systems are increasingly important for users to trust and buy into the systems, but as yet most AI systems remain "black boxes".

At present, deep learning algorithms are not generally useful for tasks requiring an explanation but are extremely useful for tasks based on large amounts of data where the desired outcome is a high level of accuracy (rather than explainability).

2.2.5 How Do They Work: Neural Networks Architectures

Allaire (2018), who created the R interface for deep learning (using Keras), summarises deep learning very simply as taking some data, and chaining together various transformations until we get an output (prediction).

Goodfellow et al (2016) describe machine learning as "essentially a form of applied statistics with increased emphasis on the use of computers to statistically estimate complicated functions and a decreased emphasis on proving confidence intervals around these functions". Deep learning is based on a combination of calculus, statistics and linear algebra. Neural networks can also be understood as being a "higher-level abstraction" of traditional machine learning models (Aggarwal 2018). Aggarwal (2018) argues that "the most basic units of computation in the neural network are inspired by traditional machine learning algorithms like least-squares regression and logistic regression" and that their mathematic success comes from the possibility of putting together a number of these basic models and minimising the prediction error by learning from the weights of the different units simultaneously.

There are a variety of deep learning architectures that each encode different assumptions (Chollet et al 2022, Zulqarnain et al 2020). These network architectures include densely connected networks, convolutional networks, recurrent networks, and transformers. The structure of the data and the assumptions of the architecture will determine if a particular type of neural network will work. Network types can also be combined with each other (Chollet et al 2022).

The Multilayer Perception (MLP) is the most popularly referenced example of a deep learning model, also called a feedforward deep network. This is described by Goodfellow et al (2016, 5) as simply "a mathematical function mapping some set of input values to output values. The function is formed by composing many simpler functions. We can think of each application of a different mathematical function as providing a new representation of the input".

Artificial neural networks are based on an architecture of layers: including input, hidden, and output layers. The 'layer' can be understood as a geometric data transformation function that is parametrised by

a set of weights and bias (Allaire, 2018). Neural networks work by feeding batches of data into the model and allowing for the model to incrementally improve its predictions by adjusting its weights (Aggarwal, 2018). The data take the form of 'x' and a known 'y'. The loss function gives the model a way to 'learn' the best way to approach the task. Using gradient descent optimisation, the neural network locates the lowest value (global minimun) to reduce the error. At each step the weights are updated, which is repeated until the steepest descent is found (Allaire, 2018). Backpropagation is used to find the output errors along the network by working backwards through the neural network and comparing the predictions with the 'actual' answer (usually the one-hot-encoded vectors). The algorithm's job is to find the function that best maps the x values onto the y values. Once found, this can be generalised onto new data for solving classification and other problems.

2.2.5.1 Convolutional Neural Networks (CNN)

As previously noted, some of the literature speaks of convolutional networks (CNNs) as one of the most successful deep learning algorithms, and the most popular algorithms for commercial use (Goodfellow et al, 2016). They were initially designed around the idea of the brain by LeCun (1989). CNNs work with data that has a clear grid-structured topology and they are able to scale models to extremely large sizes (Goodfellow et al, 2016). These algorithms are most useful for two-dimensional image processing i.e. images comprised of pixels in a 2D grid or a 3D CNN for video data. However, they may also be used for other types of sequential data like text which "can also be considered special cases of grid structured data with various types of relationships among adjacent terms" (Aggarwal, 2018). Convolution is a specialised linear operation that CNNs use in at least one of their layers (Goodfellow et al 2016, Chollet et al 2022).

2.2.5.2 Recurrent Neural Networks (RNN)

Recurrent neural networks (RNNs) are known to be best for processing one-dimensional sequential data. "Recurrent networks can scale to much longer sequences than would be practical for networks without sequence-based specialization" (Goodfellow et al, 2016). RNNs use parameter sharing i.e. extending and applying the model to examples of different lengths and generalising across them (Goodfellow et al, 2016). This is particularly useful for textual data wherein a piece of information can be found at different positions within the text sequence i.e. dates can be found at the beginning, middle, or end of a sentence. A traditional feedforward neural network would see each word in the sentence as a separate input feature and would learn the rules of the language separately according to the positions of each sentence, but with a recurrent neural network the weights are shared across many time-steps.

There are different types of RNNs including Gated Recurrent Units (GRUs) and Long Short-Term Memory models (LSTMs). LSTM is understood as 'the more powerful of the two' (Chollet et al, 2022).

Long Short-Term Memory (LSTM) means that self-loops produce paths where the gradient flows for a longer duration. The initial LSTM model of Hochreiter and Schmidhuber (1997) has been improved over time. The most notable improvement is that the weight on the self-loop can be controlled rather than fixed. This has been found to be extremely useful for a variety of applications including handwriting and speech recognition, machine translation and image captioning (Goodfellow et al, 2016).

2.2.5.3 Model Design and Assessments

Other than CNNs and RNNs there are other architectures including simple densely connected networks and transformers but for the purposes of this research report the focus is on RNNs and CNNs. Within each of these architectures there are numerous decisions to consider when designing a model, including specifying which layers are necessary, and their parameters and hyperparameters. More in-depth information on these is provided in the methodology section (Section 3.6).

Most literature indicates that accuracy is the main performance measure of classification tasks. Accuracy means the proportion of examples for which the model produces the correct output.

Bartz et al (2023) note that while training and validation of deep learning models are usually based on the loss function, the final evaluation is usually based on accuracy. Accuracy, according to Kedziora et al (2020) is the most important performance measure although Bartz et al (2023) disagree and argue that other performance measures are also important including time, complexity, and robustness or even interpretability. Although there are a wide range of arguments and studies suggesting different processes, Bartz et al (2023) note that "there are no general agreement on how to use training, validation, and test sets as well as the associated performance measures".

2.2.6 Traditional Machine Learning

To differentiate them from neural networks/ deep learning models, algorithms like Naïve Bayes, random forest and decision trees are considered in the literature as *traditional* machine learning approaches.

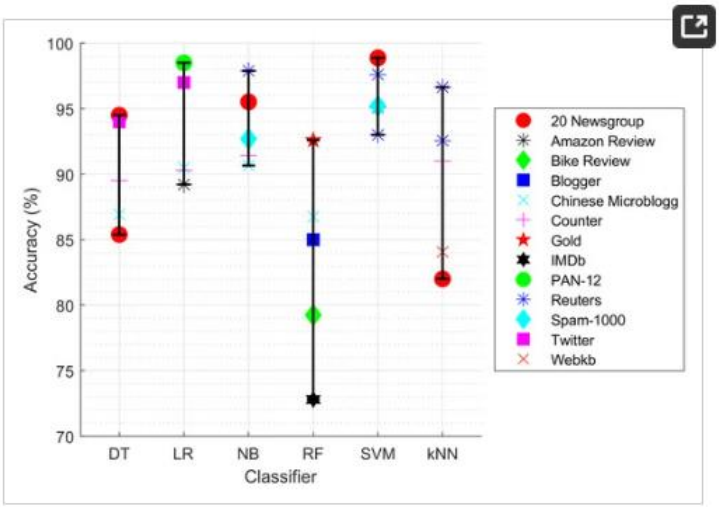


Figure 2 Image from Palanivinayagam et al (2023), depicting different classification algorithms performance on various datasets

Literature has demonstrated that these traditional models can be extremely effective for text classification tasks. (Sebastiani 2002, Palanivinayagam et al 2023). In fact, while deep learning algorithms are still catching up on language classification and sentiment analysis tasks, traditional classification algorithms such as Naïve Bayes are still most common (Mariel et al, 2018). Other traditional classification algorithms suggested in the literature include: Rule Induction, K-Nearest Neighbours, and Support Vector Machines (Ikonomakis et al 2005, Ombabi 2017)).

2.2.6.1 Naïve Bayes

Naïve Bayes is one of the simplest machine learning algorithms (Goodfellow et al 2016, Dogru et al 2021, Ikonomakis et al, 2005). It is best known for the classification task of separating legitimate e-mail from spam e-mail, based on Bayes' theorem (Goodfellow et al 2016, Dogru et al 2021). Despite its simplicity, it is generally very effective at this sort of classification task, but has also been found to not model text well (Ikonomakis et al, 2005) and its performance suffers if the features are not independent (Palanivinayagam et al, 2023). Various adjustments and variations of the traditional model have been created to solve some of these problems (Ikonomakis et al, 2005).

After surveying and comparing a number of different machine learning models Palanivinayagam et al (2023) found that the advantages of Naïve Bayes are good performance on a small training set, and the applicability to perform multiclass classification. The Naïve Bayes algorithm works by selecting single

terms, which is why it is so effective in spam email classification but not generally useful in tasks that require come complex language classification.

2.2.6.2 Decision Trees

The decision tree is quite a common machine learning algorithm for classification tasks, which works by making a sequence of decisions based on the features of the input data in order to classify the data into one of the (predefined) categories. Decision tree algorithms are based on Breiman et al's 1948 work.

The decision tree is also a network, but unlike a neural network, it functions more like a flow chart than a black-box. The branching is decided by mathematical or probabilistic evaluations at each node. The decisions are evaluations based on frequency distributions of the likelihood of events. In text classification tasks this usually means selecting a specific word/term as the splitting attribute.

Breiman et al's 1984 canonical work on 'Classification and Regression Trees', on which a lot of modern algorithms are based, describe the tree methods still used for statistical classification. The authors describe decision trees as originally inspired by social scientists' need to "cope with actual problems and data". Breiman et al's (1984) contribution is to 'strengthen and extend' the original classification algorithm known as 'THAID', developed by Morgan and Messenger in the 1970s (later to be replaced by the more popular CHAID algorithm (Kass 1980).

The original decision tree for classification, depicted below, was based on a medical diagnosis tree classifying heart-attack patients in a medical centre (Breiman et al, 1984):

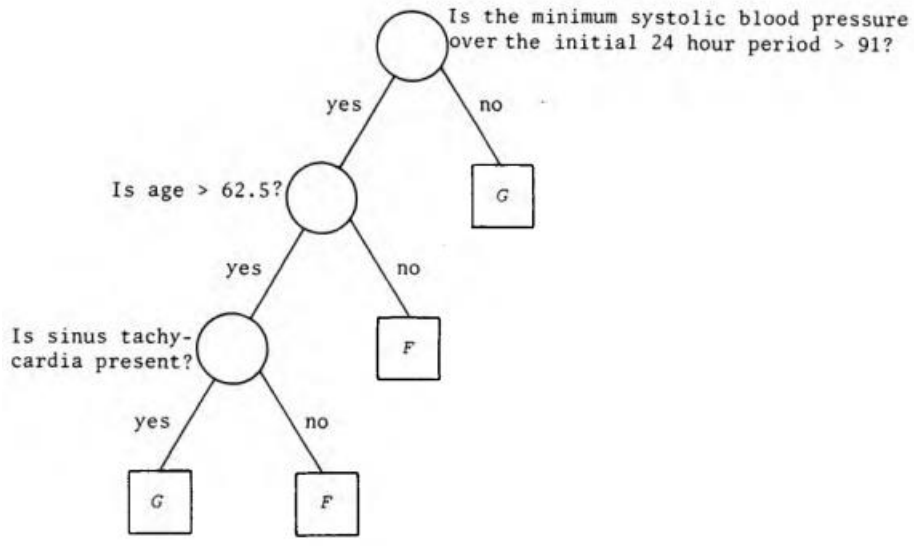


Figure 3 The original decision tree from Breiman et al (1984, 2).

The classification trees, similarly to this medical diagnosis tree are made based on measurements of a specific case, that lead to a prediction on which class the case is in (Breiman et al, 1984).

Breiman et al (1984) lay out a criterion for good classification as 1) producing accuracy in classification and 2) "providing insight and understanding into the predictive structure of the data". Their invention of the decision tree model for classification was to respond to a need for statistical classifiers able to work with larger and more complex data. These could include a variety of data types, non-standard structures, and different relationships between variables in different parts of the measurement space (Breiman et al, 1984). These factors, which challenged data scientists at the time, were identified by Breiman et al (1984) as particularly 'interesting' for their era.

Because of their flow-chart like structure, decision-trees are easy to interpret and visualize (Chollet et al, 2022). Each 'left' side of the tree represents a class label that a document is assigned to. This makes them useful for understanding which terms the algorithm has found to be important for making its classification decisions.

According to Palanivinayagam's (2023) comparative study, decision trees are not a stable classifier, they have large variance, and a small change might result in a 'drastic change' (Shah et al 2020). They are also known to be prone to overfitting on the training data. Shah et al (2020) have also found that decision trees have accuracy problems on smaller data sizes.

2.2.6.3 Random Forests

> *"Important recent problems, i.e., medical diagnosis and document retrieval, often have the property that there are many input variables, often in the hundreds or thousands, with each one containing only a small amount of information. A single tree classifier will then have accuracy only slightly better than a random choice of class. But combining trees grown using random features can produce improved accuracy." (Breiman, 2001).*

In random forest models the idea is that the model quality of decision trees can be improved by merging many individual trees into one overall model (Bartz et al, 2023). Years after Breiman's (1984) contribution of the decision tree machine learning algorithm, researchers including Breiman himself, found that "Significant improvements in classification accuracy have resulted from growing an ensemble of trees and letting them vote for the most popular class" (Breiman, 2001). The random forest algorithm is a collection of a number of specialised decision trees (i.e a 'forest' of many trees), which gives it some advantage over single decision trees. In these forests of trees, each tree "depends on the values of a random vector

sampled independently and with the same distribution for all the trees in the forest" (Breiman, 2001). The randomisation element of the algorithm helps to decorrelate the individual trees by creating a more diverse sample. As described by Breiman (2001), random forests are a favourable model both because of their error rates and because they are particularly "robust with respect to noise". The algorithm randomly selects a subset of features (words/terms) from the full set.

One of the main arguments for using a random forest model is that it can give more accurate results with nonlinear data (Shah et al, 2020). It also has better generalisability to new data than a single decision tree and is more robust to noisy data or irrelevant features than the decision tree (Shah et al, 2020). Chollet et al (2022), who advocate for the advantages of neural networks, argue that random forests are "almost always the second-best algorithm for any shallow machine learning task". Although the random forest model is not as easily interpretable as the single decision tree, it is far more interpretable than deep neural networks which are complete 'black-boxes'. "Random forest has shown great success in many real-world applications" (Shah et al, 2020) but text data often leaves room for some class imbalance in these models. The random forest method reduces the generalisation error in the following ways: Firstly, by using only a random subset of the features for each split on each individual tree, and secondly, by giving each tree a randomly drawn subset of the observations to train. This process is usually followed by bootstrap aggravating (Bartz et al 2023, Breiman 2001). Tuning of random forest models usually includes tuning the parameters of the decision trees themselves (Bartz et al 2023).

Overfitting is described in the literature as being one of the downsides of random forest algorithms, but this problem also affects decision trees and Naïve Bayes models.

2.2.7 Comparison Literature

Comparison of deep learning neural networks with various traditional machine learning models has been undertaken in a variety of studies. However, most recent studies into neural networks focus their attention on the effectiveness of imagine classification, in particular with CNN models. While there are far fewer comparative studies focusing on textual classification tasks, some of the few that do exist are described below:

Dogru et al (2021) in "Comparative Analysis of Deep Learning and Traditional Machine Learning Models for Turkish Text Classification" compare Gauss Naïve Bayes (GNB), random forest (RF), Naïve Bayes (NB) and Support Vector Machine (SVM) with a convolutional neural network (CNN) deep learning model. Their deep learning model achieved higher accuracy than their traditional ones. Another study that finds the neural network model outperforming its traditional counterparts is Mariel et al's (2018) work. Mariel et al (2018) compare a deep learning neural network algorithm with Naïve Bayes and SVM (Support

Vector Machine). They perform sentiment analysis which is a form of text classification focused on text content, on Indonesian text data.

Wu et al (2020) is another such paper that tested a variety of models on a text classification problem. Their paper presents the shortcomings of traditional text classification methods and introduces a deep-learning-based text classification process. The authors tested Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Attention Mechanisms, and others.

Zulqarnain et al (2020) compare three different deep learning models on a text classification task: Deep Belief Neural (DBN), Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN).

Ombabi et al (2017) provide one of the few works using neural networks that that combines sentiment analysis and classification. Their work aims to determine Twitter users topics of interest by combining sentiment analysis and classification and using deep learning methods. Their model was implemented in TensorFlow and achieved 97.3% classification accuracy.

Palanivinayagam et al (2023) argue that machine learning based text classification is a 'leading research area' due to its popular usage in areas like spam detection, hate speech identification, and ratings analysis etc. They survey 224 papers that use different machine learning algorithms to perform text classification. They draw particular attention to the issue of overfitting to the data which affects most of the machine learning algorithms they surveyed, to different extents. Possible ways to address the overfitting issue are presented by other authors of these comparative studies. For example, Zulqarnain et al (2020) argue for the use of a max-pooling layer.

Other comparative studies that focus on textual data argue for particular attention to be paid to the beginning or end of the model design and implementation process. Finding that more than model choice, it is the careful selection of the data itself that needs to be prioritised, Maslej-Krešňáková et al (2020) compare a number of contemporary deep learning models that predict severe anti-social behaviour in online social platforms. Their study examines the effects of different text representations and pre-processing techniques used in the different models. They also indicate the usefulness of parameter and hyperparameter tuning to reduce overfitting.

Looking more deeply at the architecture, parameters, and hyperparameters of machine learning models, Bartz et al (2023) describe the importance of intentional and informed hyperparameter tuning, which some data scientists (Chollet et al, 2022) argue should be left to the algorithm to determine for itself. Bartz et al (2023) compare and assess more than 20 hyperparameters from six machine learning and deep learning methods.

The choice between traditional machine learning algorithms and neural networks for text classification depends on factors such as the specific task, available data, and computational resources, and it is an active area of research and debate in the field (Dogru et al 2021, Mariel et al 2018, Wu et al 2020).

2.3 Bringing AI to International Relations Research

2.3.1 Politics, Text Data and Natural Language Processing

Zhou et al (2022) identify the existing lack of collaboration between policy researchers and computer scientists as a significant research gap. Their paper is one of few that aims to address this gap, which they do by constructing a policy label system based on collaboration between public administration researchers and computer science researchers, and then analyse policy texts through deep learning models.

"Text is arguably the most pervasive—and certainly the most persistent—artifact of political behavior" (Monroe and Schrodt 2009). Monroe and Schrodt give evidence that as early as the 13th century, analysis of political texts has been providing insight into political processes. By exploring the role of written documents in politics throughout history and the evolution of text analysis in the field of political science, they highlight the historical significance of political texts and the analysis thereof. The development of content analysis, the challenges it faced in its early stages, and its subsequent growth with the advent of the World Wide Web are examples of their argument. Text analysis has been, and continues to be, one of the most important issues in the study of politics (Monroe and Schrodt 2009).

Shellman (2008), who also argues for an automated coding approach to political text data, shows that automation can create much more detailed and reliable data sets than human coded data, based on a much wider range of source texts. In addition, he promotes machine processing of textual data as an "aid in the search for the elusive truth" (Shellman, 2008). AI and NLP could assist researchers in "extracting hidden meaning from texts" for example when analysing diplomatic correspondence between political actors involved in a conflict.

Grimmer and Stewart (2013) promote a "sequential, iterative, and inductive approach" to research that uses text as data. This method is grounded in an understanding that the research "process doesn't happen with the context of a single article or book, but across a community of collaborators", which means confirming the results of completed research, upon which one can then test a new hypothesis and set a basis for further research. They argue that in social sciences one can learn the most through improving our own understandings and definitions as we encounter new data. Their work promotes the idea that good research requires "an evolving understanding of the process under study" (Grimmer and Steward 2013).

Although the popular examples and understandings of machine learning and deep learning in particular focus on issues like image recognition or medical diagnostics, Tao (2022) persuasively argues that deep learning can be very useful to the international relations fields. The increasing amount of data available, including international relations data, has led to progress in issues such as predicting violent political conflicts (Tao, 2022). Tao (2022) uses a LSTM text classification model and compares its performance on international affairs text data to the performance of other classification models including a Bayesian simple classifier, KNN classifier and support vector machine classifier. Their deep learning model of international relations text classification achieves an accuracy of 90%.

Despite their successes and achievements, the literature advises caution for utilising machine learning techniques inappropriately, because even if they produce statistically good results, these might be substantively and theoretically inappropriate. Paying attention to the data and its properties is important to ensure that machine learning tools are relevant and appropriate to the project. Monroe et al (2008) look at an example of trying to predict which party politicians are affiliated with by the words in their speeches. They critique machine learning approaches (including random forest) as a solution to this problem for various reasons in particular that it "gets the data generation process backwards", namely, attempting to find a classifier function to map words into a party label, doesn't make theoretical sense when "party is not plausibly a function of word choice. Word choice is (plausibly) a function of party". They continue that the inferential problem is also not correctly addressed: "we (a) observe partisanship and word choice for some subset of our data, (b) observe only word choice for another subset, and (c) wish to develop a model for accurately inferring partisanship for the second subset". They also note that useful information might be unnecessarily discarded: "in many applications we would be using an uncertain probabilistic statement in place of what we know" (Monroe et al 2008).

Compiling a deep learning project often relies on the following two assumptions: a) that our targets can be predicted given our inputs and b) that the data available is "sufficiently informative" to learn the relationship between inputs and targets. Chollet et al (2022) explain that these assumptions remain "mere hypotheses waiting to be validated or invalidated" until there is a working model and warn that "not all problems can be solved with machine learning".

2.3.2 Literature Review Conclusions and Research Question

The literature indicates that theoretically we still know very little about the workings of Hybrid Institutional Complexes. Most of the 'complexity' literature is about regime complexes that primarily consist of intergovernmental institutions and agreements, and thorough analyses on diverse complexes made up of multiple different forms of actors are rare. What does exist on these Hybrid Institutional Complexes is based on individual examples and case studies. As yet, there is no literature on deference in

Hybrid Institutional Complexes thus making this theoretical transfer of the authority and deference literature (Clark 2021, Pratt 2018) to the Hybrid Institutional Complex (Abbott and Faude 2021, Westerwinter 2022), a productive theoretical transfer. Through this, there is much that can potentially be learned about Hybrid Institutional Complexes, as well as about organisational cooperation in institutional environments that does not only consist of intergovernmental agreements.

In machine learning, probability theory is used to model uncertainty, make predictions, and estimate the likelihood of different outcomes. If we can conclude that under conditions of 'liquid modernity' there is tremendous uncertainty about global governance complexes and how they function, then I propose machine learning as part of a solution to navigating that uncertainty. Machine learning provides an efficient way to learn patterns and make predictions and neural networks are particularly useful when the data is noisy. Machine learning algorithms can automatically learn from data, identify relevant features, and make accurate predictions based on those features. They can also be used to estimate the uncertainty associated with those predictions, which can be useful in decision-making scenarios.

The literature in the area of data science and machine learning for text classification is vast and continually evolving. Literature shows that traditional machine learning methods remain valuable, but neural networks have pushed the boundaries of what could be achieved in natural language understanding tasks. The advantages and potential of harnessing machine learning technologies to process and classify textual data could allow for significant progress to be made not only in computer sciences but also in the social sciences. Being able to work with the massive amounts of data available on the internet in an efficient way could allow fields like international relations to develop new theoretical insights that keep up with the shifts and changes in the field.

Texts, i.e. written language, can offer us tremendous insight into the workings, intentions, and relationships of an institution. But without an efficient way to classify these, the sheer amount of raw data is too big. The heterogenous complex of institutions governing private security globally, collectively and individually put out massive amounts of text annually to explain what they do, what their policies are, who they work with and how. A political, transnational, diverse complex like this, which is not represented/led by a single entity, is a socially significant institution for researchers to understand, but it is not a simple task. It requires engaging with each of the individual institutions in the complex, as well as the relationships between them. Although international relations scholars have historically used hand-coding methods to generate and analyse their data, the costs and time involved make this impractical. Without machine learning tools this task is almost impossible, especially as the relationships in institutional complexes shift constantly.

This research therefore tests the effectiveness of machine learning for text classification of the institutional output documents released by institutions in the global private security governance

complex. It further compares traditional machine learning models to deep neural network models by assessing the accuracy of their classifications of the texts.

3. Methodology

3.1 Section Outline

The methodology section below describes the raw data and the classification of deference of authority used in my hand-coding of the training data, and then explains how I build and run the machine learning models, starting with the neural network and then the traditional models:

Section 3.2 describes the raw dataset and outlines how the data for this research is generated. It provides the coding scheme and a detailed description of the hand-coding classification process.

Section 3.3 describes how I split the data into useable subsets for training the machine learning algorithms.

Section 3.4 presents the languages and packages used in this research task.

Section 3.5 explains the data preparation and pre-processing including the automatic pre-processing functions I tested, and the processes of tokenising and indexing the texts into a format readable by the algorithms.

Section 3.6 sets out the architecture used for my initial neural network (Model 1) and then adjusts this architecture to test and improve the model results by presenting 13 other deep learning models with slight variations in the model design and hyperparameters.

Section 3.7 presents three traditional machine learning models – random forest, decision trees, and Naïve Bayes models.

The results of each model are provided under their architecture in this section but the full comparable table of results from all the models is available in the findings chapter (Chapter 4).

3.2 Generating Data

3.2.1 Raw Data and Coding Scheme

As mentioned in the literature review, scholars (Monroe et al 2008) have emphasized the importance of appropriate data for machine learning problems: Grimmer and Stewart (2013) argue that rather than

opting for the most convenient data, researchers should obtain the most appropriate and useful data. In fact, it is argued that the careful selecting of appropriate data is even more important than the algorithm design because "a model's ability to generalise comes almost entirely from the properties of the data it is trained on" (Chollet et al 2022).

The original raw data used in this research comes from Westerwinter's (2022) paper on *Hierarchy, Differentiation, and Inter-institutional Collaboration in the Regime Complex for Global Private Security Governance*, and I respond to that paper's call for further analysis into the hierarchies and forms of collaboration in global governance complexes. I do this by replicating (and adjusting) the coding scheme from Pratt's (2018) work on *Deference and Hierarchy in International Regime Complexes* which uses an ordinal scale to code for instances of institutional deference between different organisations that make up a specific global governance complex.

This work extends Pratt's (2018) work on institutional deference by a) testing its applicability to a more diverse and complex form of regime complex, and contributes to Westerwinter's (2022) research by creating a more statistically valuable measurement of authority than his original 'hierarchy' variable; namely, organisational deference.

Taking guidance from many recent studies into global governance complexes (Westerwinter 2022, Pratt 2018, Clarke 2021, Brosig 2015, Abbot et al 2015, Hale and Roger 2014, Oberthur and Stokke 2011, Abbot and Faude 2021, Abrahamsen and Williams 2007) that statistically examine one or a few individual institutional complexes in order to generate generalisable findings about institutional governance complexes in general, I propose that by focusing on the private security governance complex specifically I will be contributing some knowledge to the studies of the Hybrid Institutional Complexes overall.

Westerwinter's (2022) original dataset is made from the output documents of the 11 institutions that were active at some point between 1987 and 2020, which he identified as part of the global private security governance complex. The data covers the 720 directed dyad pairs of institutions over 2000-2020 in a cross-sectional time series. Organisational output documents from the 11 institutions were collected from the official websites of each organisation through a mix of web scraping and manual downloading from archives, and saved in a time stamped form (Westerwinter, 2022). They were then text-mined through natural language-based entity recognition algorithms to identify the snippets of text in which each of the organisations in the complex has been referenced. This data was collated by Westerwinter into an uncoded dataset (i.e. not yet classified into different categories). The data file contains 13 135 text extracts (each between a couple of sentences to a paragraph in length).

I used the coding structure outlined by Pratt (2018) to look for instances of organisational deference, and hand-coded just over 1 250 of the 13 135 text snippets accordingly. Each time an institution in a regime complex publishes something that articulates its rules or intentions, "it has an opportunity to defer to the

rules of another organization. As a result, the presence or absence of deference in …[the texts] reveals the extent to which institutions successfully coordinate rule making" (Pratt 2018).

The coding scheme rates the references from organisation *i* to organisation *j* on a scale of 1-5, with (1) being 'passing reference', (2) being (a relevant) 'rule reference', (3) being 'intent to coordinate', (4) as 'cooperative action and rule endorsement', and (5) being a proper case of deference namely institution '*i*' explicitly accepting institution '*j*''s authority on an issue (Pratt 2018).

The table below shows Pratt's coding scheme and his examples.

Score	Type	Criteria	Example
1	Passing Reference	IO A refers to IO B's activities or rules on a matter not relevant to the issue area	Commonwealth Secretariat, 1995: "[We] welcomed the adoption by the Organization of African Unity of the Pelindaba Treaty on the Establish- ment of an African Nuclear Weapon Free Zone."
2	Rule Reference	IO A refers to IO B's activities or rules in the issue area	Organization of American States, 2010: "The Commonwealth Secretariat also mounted a three- person [election monitoring] mission under the leader- ship of the Hon. Chris Carter, a former New Zealand Minister."
3	Intent to Coordinate	IO A makes an attempt to coordinate its activities or rules with IO B	World Trade Organization, 1998: "The secretariat was asked to contact the FAO, the secretariat of the Convention on Biological Diversity and UPOV to request factual information on their activities."
4	Cooperative Action and Rule Endorsement	IO A engages in a joint endeavor with IO B or endorses a set of rules or activities undertaken by IO B	World Trade Organization, 1999: "The secretariat cooperates with a number of inter- governmental organizations, notably with WIPO pursuant to the agreement between WIPO and the WTO...and the joint initiative on technical cooperation."
5	Deference	IO A explicitly accepts IO B's authority on a particular issue	Asia Pacific Economic Cooperation, 2002: "[APEC members] are implementing the measures called for in relevant UN Security Council resolutions and are putting in place the legal and regulatory mechanisms to implement Resolution 1373."

Table 1 Institutional Deference Coding Scheme (Pratt 2018, Supplementary Material).

Pratt's coding scheme was created to look for instances of deference of authority on an ordinal scale in policy documents produced by international organisations made up of member states. He focused particularly on the relationship between hierarchy and division of labour between the different international organisations in the regime complex. A lot of his research focuses on the power differentiation between member states that make up each international organisation in the complexes. In shifting the level of analysis to regime complexes that are made up of a more diverse array of institutions than what Pratt studies, it also made sense to broaden the scope of the raw data used; so rather than restricting the data to purely 'policy documents' as Pratt did, I instead include 'all organisational output documents' from the diverse HICs that Westerwinter (2022) uses in his dataset.

Because of the heterogeneity of the institutions in the HIC, not all of them do a lot of policy work (as in Pratt's (2018) case studies) but they are all active and do other governance work, and so the data includes not only policy papers but also meeting summaries, recommendations and standards, research documents, briefings, case studies, best practice collections etc. These output documents are focused on English speaking texts over a certain length. Applying Pratt's original coding scheme to a new, somewhat different set of data raised issues and required some alterations to be made to the coding scheme to ensure it was applicable to the Hybrid Institutional Complex (HIC) data, whilst remaining comparable to Pratt's initial study.

Below is an example of a hand-coded section of the dataset.

Organisation	File Name	Date of document publication	Text Snippet	Deference score	Name of organisation deferred to
International Code of Conduct for Private Security Providers (ICOCA)	FINAL_ICoCA%20Q3%202019%20MoM.txt	2019	Executive Director expressed Association appreciation additional contributions received DCAF Government thanked DRL Swiss Governments past support	4 (cooperative action)	Geneva Centre for the Democratic Control of Armed Forces (DCAF)

Table 2 Example of one hand-coded text in the data table

3.2.2. Hand-Coding Problems and Machine Learning as a Potential Solution

The hand coding process involved going through the snippets and familiarizing myself with the data and background information sufficiently to be able to make coding decisions. Hand-coding just 1264 lines was a long and strenuous process which was possible only through developing a sound qualitative understanding of the context of the data. Often reading the snippets alone seemed insufficient to make a decision on what code to give and it required either further research (looking at the original document itself) or consultation with the dataset's author, Dr Westerwinter.

I trained myself as a researcher on coding this data. I had the coding scheme and a very small number of examples to follow from Pratt's (2018) materials. I had a thorough qualitative understanding of the purpose of the task and the desired outcomes that I hoped to achieve; namely that I wanted to create a dataset that measured instances of institutional deference to the authority of other organisations in the institutional complex. In reading each line of text I had this question in my mind: does this text snippet from one organisation indicate a level of deference to the authority of one of the other organisations?

First is the simple yes/no question: “is there a reference from org *i* to org *j* present”? This is followed by a more theoretically difficult question: “if yes, then is it a 1,2,3,4, or 5 according to the coding scheme”.

The hand-coding process of creating this data set of 1 264 labelled texts took five months, which was far longer than I had anticipated.

The difficulties in the hand-coding process raised a number of issues relevant to this research:
1. Is this form of data collection and analysis in fact possible and worthwhile to do manually? Or

2. Could machine learning be a solution, and a way to create robust quantitative data out of institutional output texts?

3. Or could machine learning be the wrong approach entirely, due to the complex nature of the data?

If a machine learning approach to this research turns out to be effective, then that has significant implications for international relations research going forward, especially in terms of Hybrid Institutional Complexes which are ever evolving and under-researched.

This leads to the second section of my methodology: using this hand-labelled data for building and training machine learning algorithms that will be able to automatically classify the remaining 11 750 texts:

3.3 Splitting Data into Useable Subsets

3.3.1 Training and Validation Splits

Supervised machine learning generally requires a subset of the data to be hand-coded to train the algorithm. In my case I have a subset of 1 264 lines of text that I have classified by hand, which I then further sub-divide into a training and validation set, which will be used to train, evaluate, and adjust the models. I use a function to randomly split the data into the necessary subsections without any bias. I use 80% of the data for training and 20% for validation, which I have split using a random splitting function. Further research could include a 3rd split for use as a testing set.

3.3.2 United Nations (UN) Data Focus

For the purpose of this research question, my study isolated a smaller subsection of data, the references from any of the 11 organisations in the complex to the United Nations (UN) in general, as this was the section with most available data. I cleaned and processed this subsection of the hand-coded (labelled) data with the following results.

Title of code	Passing reference	Relevant rule reference	Intent to coordinate	Cooperative action and rule endorsement	Deference of authority.
Code	1	2	3	4	5
Number of references in that code.	569	46	61	98	6

Table 3 Hand-coded sample: Count for each category of reference to the UN

This table shows the results without the NA values (ie texts that contain no references to the UN)

6 categories (including those that were not references to the UN) for only 1264 texts would be inefficient for building my models so I further condensed the categories into only 4 categories by recoding the data to make the NA's into 0s and to combine the 2's and 3's into 2s and the 5s and 4s into 3s. Leaving me with the table below:

Explanation of code:	No deference to the UN	Passing reference to the UN	Partial deference to the UN: Relevant rule reference or intent to coordinate	Actual Deference of authority to the UN: Cooperative action/rule endorsement/deference of authority
Code:	0	1	2	3
Number of references in that code	484	569	107	104

Table 4 Re-coded Data with Four Classes of Deference Types

Below is an example from text row 1 259, which shows the text and the label assigned to it. In this case, I had originally hand-labelled it with a 3- 'intent to coordinate' but after recoding down to only the four categories above it is now a 2.

Text(row) Number	Text	Label
1259	develop regular dialogue discuss possible areas cooperation Governments relevant actors including relevant United Nations bodies specialized agencies funds programmes particular Office United Nations High Commissioner Human Rights Global Compact International Labour Organization World Bank International Finance Corporation United Nations Development Programme International Organization Migration well transnational corporations business enterprises national human rights institutions representatives indigenous peoples civil society organizations regional subregional international organizations	2: 'Partial deference to the UN' (Relevant rule reference or intent to coordinate)

Table 5 Example: Labeled Text Row 1 259

This MA research project tests different classification models on this subsection of data with the intention that if I can successfully write a classification algorithm that determines the different levels of deference to the UN organisations, that my further research can extend this model and test it against the other institutions in the complex. Ultimately, a multi-class, multi-level classification model could be a useful tool for this sort of a methodological question, which I propose for further research.

3.4 Language and Packages

The R programming language is suitable for running and evaluating all the machine learning algorithms that I test in this paper. R has packages that provide for the easy and straightforward running of the traditional Machine learning models: Naïve Bayes (Meyer et al, 2023), decision trees (Therneau and Atkinson, 2022), and random forests (Liaw and Weiner, 2022). To date, one of the most accessible, high-level APIs for building and running neural networks is Keras and its popular backend – TensorFlow (Allaire and Tang, 2022). TensorFlow is an interface for expressing, implementing and executing machine learning algorithms. As described by its authors "the system is flexible and can be used to express a wide variety of algorithms, including training and inference algorithms for deep neural network models, and it has been used for conducting research and for deploying machine learning systems into production across more than a dozen areas of computer science and other fields, including speech recognition, computer vision, robotics, information retrieval, natural language processing, geographic information extraction, and computational drug discovery". Keras is a "rapidly emerging" (Sumathi and Alluri, 2021) open-source Application Programming Interface (API) developed by François Chollet, a Google engineer, and the R interface to Keras (Allaire and Chollet, 2022) was thereafter developed by J.J. Allaire. It is relatively easy to use and effective in processing large data volumes (Sumathi and Alluri 2021, Chollet et al 2022). Although Keras uses a Python environment, it can now be implemented entirely through R code, in R Studio. I relied on this to run my neural network models. My subsection of textual data was small enough that I could work effectively on the CPU without need for a separate GPU.

3.5 Preparing the Data for Input into Machine Learning Algorithms

3.5.1 Text Pre-Processing

Machine learning models including deep learning models for textual data can be used to perform pattern recognition (and classification) on texts: words, sentences, paragraphs, but they generally don't use raw text as input. Rather they require language that has been represented in a numeric format. Below I describe how I 'pre-process' my text data into a numerical format appropriate for modelling.

3.5.2 Indexing, Tokenization and Standardisation

Vocabulary Indexing is the process whereby each token is encoded into a numerical representation. This is done by the automatic construction of an index of all the terms found in the training data and the assignment of a unique integer to each entry (Chollet, 2018). A special vector encoding keeps the integer index position in a way that can be processed by the neural network.

Tokenisation refers to breaking down the text into smaller components (tokens) and then representing them according to a unique integer index. Tokens are usually either words, characters, or n-grams. Tensorflow (Allaire and Tang 2022) has a built-in tokenizer function and other packages are available in r that provide for automatic tokenization of text corpuses.

My methodology involved initially pre-processing the texts for modelling in traditional machine learning algorithms, which included using the *tm* package (Feinerer and Hornik, 2023). The *tm* package provides for automatic standardisation, removal of punctuation, decapitalising words, removing stop words, stemming and tokenization and processing these into a matrix where words are represented according to their document and weight.

However, a document term matrix is non-positional, which means that word order does not matter, whereas it does in a corpus. Poor results could have been generated by a model that didn't consider word order and so my methodology shifted to a processing tool that considers word order, by creating a data corpus that holds word position, along with the various document term matrixes that I generated from this.

As Chollet et all (2022, 345) explain,

"The problem of order in natural language is an interesting one: unlike the steps of a time series, words in a sentence don't have a natural, canonical order. Different languages order similar words in very different ways… Even within a given language, you can typically say the same thing in different ways by reshuffling the words a bit... Order is clearly important, but its relationship to meaning isn't straightforward".

A sequence model was a more appropriate text processing model for me to use because word order in this context is important, whereas with bag-of-words processing models the original order is discarded. In the case of my data, institutions reference each other in the output documents in various ways. But it is *how* they reference each other that matters. The word order drastically affects the difference between a passing reference noting the existence of another organisation in the complex or a reference deferring authority to the other organisation in the complex, or a reference claiming that it was deferred to by another organisation (i.e. reversal of the direction of deference that I'm looking for). For this reason,

TensorFlow's inbuilt *tokenizer* function was most useful, specifying *texts_to_sequences* in order to process the text corpus (Allaire and Tang 2022).

When building and testing the neural network model, it became clear that the inbuilt pre-processing functions that are part of TensorFlow (Allaire and Tang 2022) are more appropriate for pre-processing the texts for this task, because they consider important elements that give the model strength, such as the consideration of word sequence/ order in the model. TensorFlow's tokenizer function automatically removes all punctuation and turns the texts into space-separated sequences of words, which are then split into lists of tokens to be indexed or vectorised. The 'tokenizer' function includes an optional argument for specifying the maximum number of most common words to be kept when tokenizing the text, guided by recommendations from the literature, I initially selected the 10 000 most common words.

However, after running my first models I found that the keeping 10 000 words actually lowered the accuracy of the models due to too many words that only had one occurrence and so I later adjusted the tokenizer's specifications to keep the 5 000 most common words.

Table 6 below shows the frequencies of the 40 most common words in the data.

word	**count**
security	1569
united	1100
nations	1026
international	1012
rights	948
human	930
states	661
document	586
national	579
montreux	527
law	495
private	476
sector	467
dcaf	453
support	452
development	398
companies	367
reform	348
ssr	343
state	324
icoc	321
council	314
general	305
working	294
group	291
code	284
good	271
including	267
icoca	266
governance	253

principles	249
military	249
note	247
police	246
practices	245
women	243
armed	227
this	222
conduct	222
undp	214

Table 6 Most Common Words and Their Frequencies

The *tokenizer* function provides for automatic fitting of texts to sequences, which is particularly useful for a sequential model. But in order to effectively run most models, the sequence lengths need to be uniform. After turning the text sequences into integer sequences, I pad the shorter sequences with 0's in order to keep all of them the exact same length. I use Kera's *pad_sequences* function in order to transform the list of sequences into the shape: *number of samples, maximum sequence length* (Allaire and Chollet, 2022). This function then pads the sequences that are shorter than the specified maximum sequence length with 0s. Whilst zero padding does increase the cost (in terms of time and memory), it helps keep the representation size large on Convolutional Neural Networks (Barz et al 2023).

Monroe et al (2008) critiqued the removal of stop words in policy documents, and they show that this practice might result in extremely inaccurate results. Whilst their argument is convincing and should be tested in future studies, my data already had the stop words removed so I was unable to test this. However, additional work could look further into this. Table 7 below depicts an example of the pre-processed text compared to the original raw text.

Pre-processed text in the database, stop-words have been removed and punctuation has been removed.	**Raw text from the original document**
Executive Director expressed Association appreciation additional contributions received DCAF Government thanked DRL Swiss Governments past support	The Executive Director expressed the Association's appreciation for the additional contributions received from DCAF and the UK Government and thanked DRL, the Swiss and UK Governments for their past support.

Table 7 Database Text with Stop Words and Punctuation Removed Comparison to Original Text

3.5.3 One-hot Encoding Labels

After I pre-processed the texts into sequences of 'tokens' of uniform lengths I then pre-processed my labels.

The labels are the classifications that I hand-coded based on the level of deference of authority to the UN that each text represented. These will be the 'answers' used to train the model.

Labels needed to be one-hot-encoded to be compatible with the neural network architecture. That means rather than just being the single deference scores corresponding to each text, they need to be

made into a binary matrix of zeros and ones where zeros are present for all the labels that the text is not labelled as, and a one for the correct label:

In my data set the labels are the numerical values of 0, 1, 2, or 3 (See Table 3). One-hot encoding them involves arranging a matrix of four columns with each column representing each class (ie levels 0, 1, 2, or 3 of deference to the UN). Each row will contain zero-values in the columns that do not reflect the deference score of that row, and the value of one in the column that does contain the deference score of that row. For example: the table below shows that the text in row 1259 (which I showed earlier in Table 5) which under the revised coding scheme correlates to the deference score of 2- meaning the text is classified as 'rule reference [to the UN]' or 'intent to coordinate [with the UN]'. Therefore, the one-hot-encoded label for this text (row 1 259) takes the form of zeros in the columns representing labels 0,1, and 3 and a one in column 2. The one-hot-encoded scores for the next 5 texts are also printed below.

		Classes			
		0	**1**	**2**	**3**
Row (Text) number	1259	0	0	1	0
	1260	0	0	1	0
	1261	1	0	0	0
	1262	0	0	1	0
	1263	1	0	0	0
	1264	1	0	0	0

Table 8 One-Hot-Encoded Texts Example

While the one-hot-encoded labels are appropriate for the neural network, they are in the incorrect structure to be processed by the traditional models and so when I run those I turn them into a factor of levels 0,1,2,3 rather than a one-hot-encoded matrix.

At this stage the texts and labels have been sufficiently pre-processed to use them to train the algorithms. Next, I will describe the machine learning algorithms and their architecture, starting with the deep learning models.

3.6 Neural Network Models

3.6.1 Model Design and Rationale

Keras sequential models (Allaire and Chollet 2022) are made up of an arrangement of layers in which there is one input and one output tensor in each layer. The sequential model consists of a sequence of layer functions. In an artificial neural network, data is transferred from one layer to another in a specific linear order until it reaches the output layer (Sumathi and Alluri 2021). I chose the Keras Sequential

model because it allows one to build on many different kinds of layers and so I am not restricted to a specific model type but can test multiple types and combine.

Keras Sequential models for text classification can be made through a Recurrent Neural Network (RNN) or a Convolutional Neural Network (CNN). I first run a few RNNs and thereafter a few CNNs, as well as a model with both.

I begin with an RNN model because Recurrent Neural Networks (RNNs) are predominantly used for textual/language modelling applications (Aggarwal 2018) and of these, a multi-layer or bidirectional LSTM improves the performance of a simple, single-layer RNN (Aggerwal 2018). I based my initial architecture on the commonly used models for sentence-level-classification which most often take the form of sentiment analysis tasks (the classic example of which is the two-class problem of 'positive sentiment vs negative sentiment').

As described in the literature review (Section 2.2.5), the Long Short-Term Memory (LSTM) Network is a type of RNN. In my methodology it takes the form of a Keras Sequential model with an LSTM layer. The LSTM layer is known to be particularly useful for tasks that involve understanding and processing sequences of words in text data. LSTM layers help models to learn the complex patterns and relationships within the text, which is especially important for tasks where context and word order is important for classification. LSTM algorithms maintain long-term memory, allowing the model to remember information from the beginning of a sequence and make connections with words occurring later in the sequence. They also are useful for handling data with sequences of different lengths.

Below I will describe the model architecture, parameters and hyperparameters for the 14 neural network models I test. Each model is a slight variation of the model(s) before it, in order to incrementally test out the ability for different adjustments to improve the model's fit and accuracy. A tabular summary of all the model's designs and results are available the beginning of Chapter 4.

Model 1

Sequential model with a LSTM layer (32 units) One dense layer with a softmax activation.

My first neural network model has the following architecture: It relies on the sequential API (Allaire and Chollet, 2022), and uses an embedding layer based on the length of the input values (my word index) and an output dimension of 200. It also has a LSTM layer (of 32 units), and 4 densely connected units in the final layer – because of the 4 classes of my predictor variable (namely the deference scores of 0-3 as shown in table 3). I use the *softmax* activation function. The activation function is used to break linearity and transform the input. The *softmax* activation function transforms the score into a probability: the highest number is the predicted class, as shown below:

$$\text{softmax}(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{k} \exp(x_j)}$$

In compiling the model, I use the categorical cross entropy loss function (which is generally accepted as an appropriate loss function for a multi-class-classification problem), and the 'Adam' optimiser. The name 'Adam' refers to 'adaptive moments' because it is an adaptive learning rate algorithm. It is "generally regarded as being fairly robust to the choice of hyperparameters" (Goodfellow et al 2016, 306) and although there is no consensus on what the 'best' optimisation algorithms are (Goodfellow et al, 2016), Adam is the default optimiser in most Keras functions (Bartz et al, 2023).

I chose an embedding dimension of 200 after running some initial experimental models with smaller embedding dimensions. Although a bigger embedding dimension can further the risk of overfitting, in cases where the relationships between the words are important, a larger embedding dimension might be necessary. While simpler tasks like sentiment analysis can use smaller embeddings, tasks like document classification involves a deeper and more nuanced examination of the words and their relationships. I found that a larger embedding was necessary for this task.

I trained this model with 10 epochs and a batch size of 32 and validated against the 20% of validation data I had set aside. The batch size within each epoch represents the number of training examples considered to compute the gradients for one weight update step, based on the gradient descent principles (Goodfellow et al, 2016). Namely, the neural network finds the place with the lowest value (global minimun) to reduce the error.

The loss function I used is categorical cross-entropy which is appropriate for classification tasks with multiple outcome classes. The loss function is a measure of how 'bad' it is to predict y^ given that the true label is y (Bartz et al, 2023).

Below is the summary of this model followed by the loss and accuracy graph generated during training:

```
Model 1
_________________________________________________
 Layer (type)         Output Shape         Param #
=================================================
 embedding (Embedding) (None, 373, 200)    1618600
 lstm (LSTM)          (None, 32)           29824
 dense (Dense)        (None, 4)            132
=================================================
Total params: 1,648,556
Trainable params: 1,648,556
Non-trainable params: 0
_________________________________________________
```

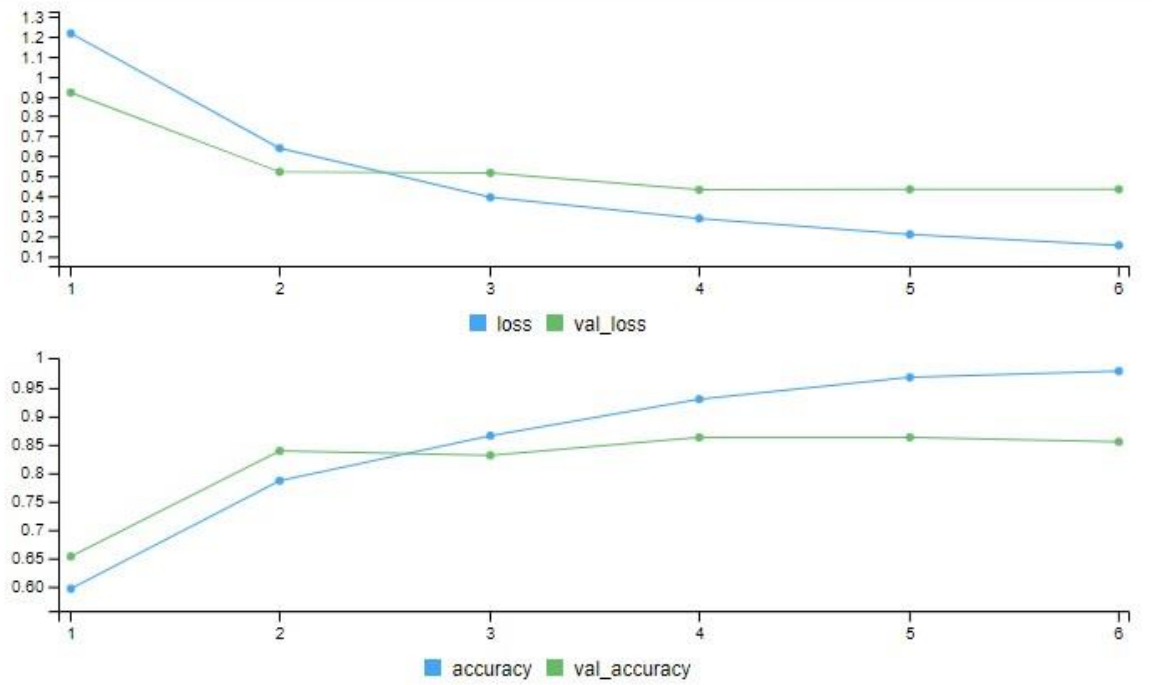


Figure 4 Keras Model 1 Loss and Accuracy

As depicted in Figure 4 above, the maximum validation accuracy for Model 1 is 0.86 which can be found at the 4th epoch. This model ran at a speed of 34.25 seconds.

To evaluate the model, I assess how well the model performs (by measuring accuracy) on data it hasn't seen before, ie data that wasn't used for training it. This model uses the validation set, which is the portion of data set aside and used during optimisation to estimate the model's prediction error.

Model 1 gives an accuracy score of just under 87% on classifying the validation set. This score is lower than the accuracy score from the training data, but more reliable because the model did not see the validation data during its initial learning. As noted previously, ideally, there could also be a third unseen data set, the 'test set' which the model would have never seen at all, but this requires a bigger data set in order to have separated the data into these three groups and still have enough data to be used for training.

Model 2

LSTM layer 64 units

Distinct from the model parameters, which are variables learned from the data and updated during training, the learning algorithm itself has parameters called hyperparameters. These are configurations of the models that influence the model performance (Bartz et al 2023). Hyperparameters can be understood as 'settings' which must be determined outside the learning algorithm itself (Goodfellow et

al). These include number of hidden units, and number of layers, learning rate, batch size, optimiser, kernel size, and dropout rate.

Layers

As described by Bartz et al (2023) the layer parameter determines the number of hidden layers in the neural network (other than the input and output layers which are given in every neural network). The larger value of the layer parameter, the more complex the model, and the more model coefficients are necessary. These result in a higher runtime but often in a higher quality model. However, without regularisation measures or early stopping then there is always an over-fitting risk to consider (Bartz et al, 2023). The layer type is an integer, and defaults to 1. According to Bartz et al (2023, 60), the "influence of layers can be extreme", i.e. "By varying this value, extremely simple (no hidden layer or only one hidden layer with very few neurons) or extremely complex models (thousands of layers and neurons) can be generated".

Units

The 'units' parameter determines the number of neurons in the layer i.e.. the size of the network layer in the hidden layer. This only affects hidden layers because input layers' dimensions are determined by the dimensionality of the data and the number of units in the output layer are dependent on the task, i.e. the four classes of classification in my work means there must always be four units in the output layer.
Just as described above regarding layers, larger values assigned to 'units' means a more complex model but increases the risk of overfitting without regularisation or early stopping (Bartz et al 2023). And as with layers, the influence of units can be 'extreme' (Bartz et al 2023), i.e. it can vary from very few to thousands of neurons. Li et al's (2018) study shows that network depth can strongly influence weight optimisation. Li et al (2018) also show that the relationship between weights and model quality becomes increasingly nonlinear as the network gets deeper and more complex and the difficulty of weight optimisation increases. This can be mitigated by using more neurons per layer, or by using connections that skip layers (Li et al 2019, Bartz et al 2023). Bengio (2012) recommends that the first hidden layer has more neurons than the input layer.

To test this, I ran Model 2 based on the same architecture as model 1 but I increased the LSTM layer to 64 units and trained the model with an increased number of epochs to 20 (which decreased the speed at which the model ran). See below:

```
Model 2
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_1 (Embedding) (None, 373, 200)     1618600
 lstm_1 (LSTM)                (None, 64)                67840
 dense_1 (Dense)              (None, 4)                 260
=================================================================
Total params: 1,686,700
Trainable params: 1,686,700
Non-trainable params: 0
```

Figure 5 below shows the loss and accuracy at each of the 20 epochs.

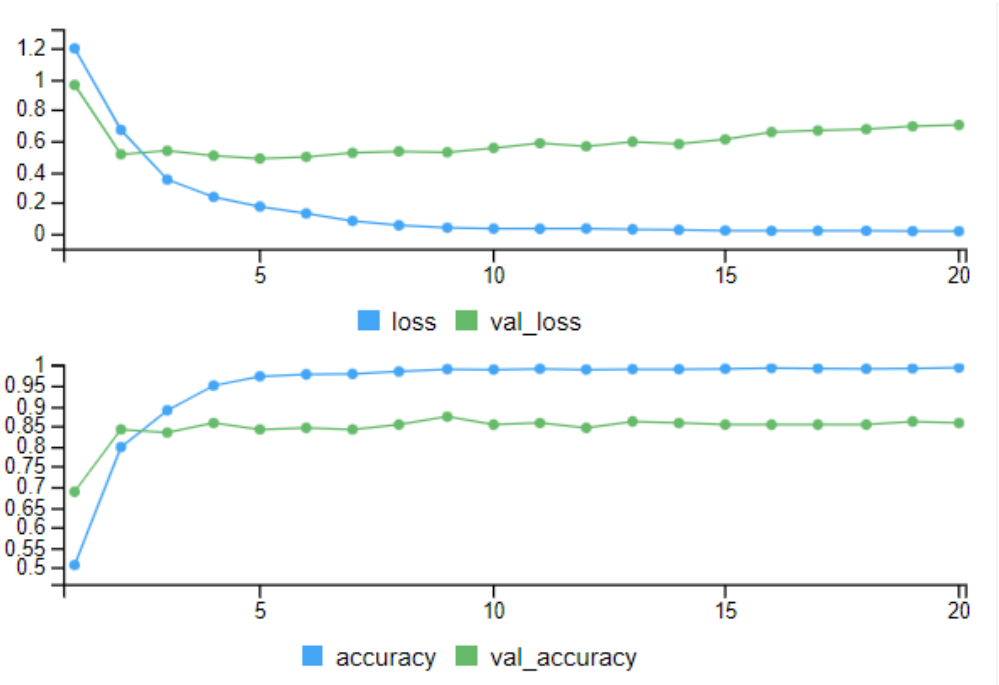


Figure 5 Keras Model 2 Loss and Accuracy

As can be seen in Figure 5, the best accuracy was at the 9th epoch and it decreased thereafter so my later models with equally simple architecture can be run with only 10-15 epochs.

The increased LSTM layer size slightly improved the models accuracy from Model 1, to just over 87% (0.8 735.)

Model 3

LSTM 64 Units and custom learning rate

Learning Rate

The learning rate is a positive scalar determining the size of the step (Goodfellow et al, 2016). The literature shows that while it is essential to choose an appropriate learning rate in order to effectively train a deep learning model, this often requires some experimentation. Best practice is to start with default learning rates and then assess a range of learning rates in a 'grid search' by evaluating the performance of the different learning rates against each other to find the best one. As previously mentioned, Barz et al (2023) argue that the Adam optimiser often must rely on a learning rate that accounts for its bias correction.

Model 3 therefore uses the same architecture as model 2 (LSTM with 64 units) but I specified a custom learning rate of 0.001 and then trained the model with 11 epochs (see Figure 6). The customised learning rate slightly improved the performance of the model achieving a maximum validation accuracy of over 88%.

```
Model 3
_________________________________________________________
 Layer (type)           Output Shape          Param #
=========================================================
 embedding_3 (Embedding) (None, 373, 200)     1618600
 lstm_3 (LSTM)          (None, 64)            67840
 dense_3 (Dense)        (None, 4)             260
=========================================================
Total params: 1,686,700
Trainable params: 1,686,700
Non-trainable params: 0
_________________________________________________________
```

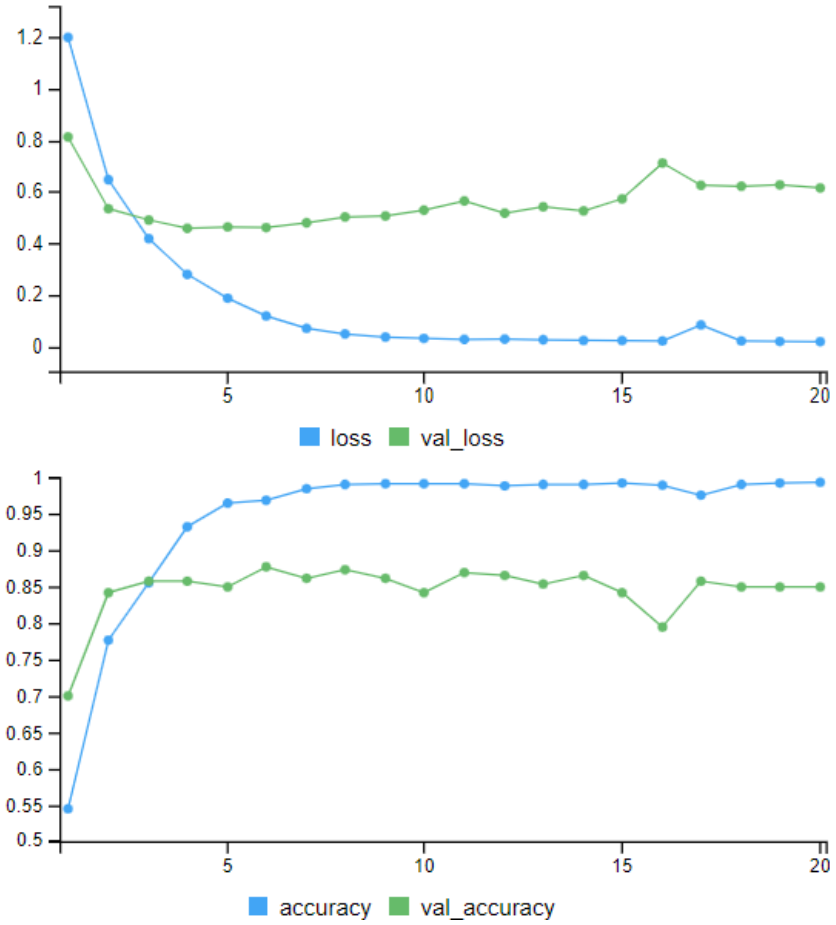


Figure 6 Keras Model 3 Loss and Accuracy

Model 4

LSTM with 96 units

To further test the effect of the adjustment in LSTM layer units, I ran the same model but increased the LSTM layer to 96 units. The results of this were worse than the model with 64 units – dropping the validation accuracy from above 88% on the previous model to below 83% (0.8221).

```
Model 4
_________________________________________________________
 Layer (type)           Output Shape          Param #
=========================================================
 embedding_4 (Embedding) (None, 373, 200)     1618600
 lstm_4 (LSTM)          (None, 96)            114048
 dense_4 (Dense)        (None, 4)             388
=========================================================
Total params: 1,733,036
Trainable params: 1,733,036
Non-trainable params: 0
_________________________________________________________
```

Model 5

LSTM adjusted learning rate

I then re-implemented the model that had been achieving the best accuracy (Model 3 with 64 LSTM units and the custom learning rate of 0.001) but this time I re-adjusted the learning rate. I used a learning rate of 0.0001. (I also trained it on 15 epochs to give it more time for the model to improve. As can be seen in Figure 7, the highest accuracy was found in the 12th epoch). However, this adjusted learning rate was counterproductive and dropped my accuracy to under 85% (0.845 whereas with the original learning rate in model 3 I achieved over 88%.

```
Model 5
_________________________________________________________
 Layer (type)           Output Shape          Param #
=========================================================
 embedding_5 (Embedding)  (None, 373, 200)     1618600
 lstm_5 (LSTM)          (None, 64)            67840
 dense_5 (Dense)        (None, 4)             260
=========================================================
Total params: 1,686,700
Trainable params: 1,686,700
Non-trainable params: 0
_________________________________________________________
```

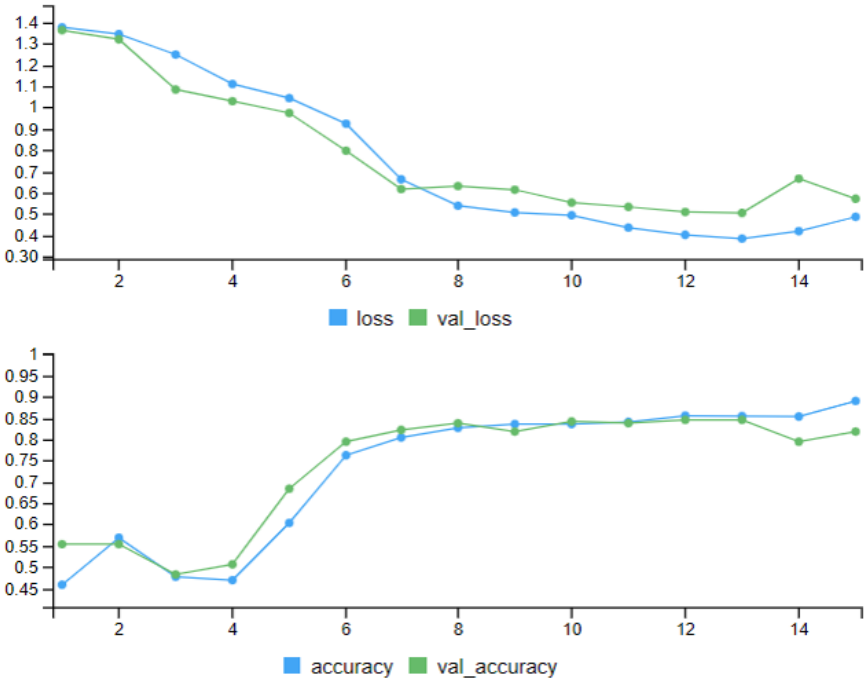


Figure 7 Keras Model 5 Loss and Accuracy

Model 6

CNN model

This model focuses on capturing text features using convolution filters rather than recurrent layers which focus on sequential dependencies.

Because the literature focused so heavily on CNN models, I also tested some of these, by replacing the LSTM layer with a one-dimensional convolution layer and adjusting the width and depth of the layer by specifying the number of filters and kernel size, in various trials with different specifications.

The one-dimensional convolution layer is designed to capture local patterns and relationships within a one-dimensional sequence – such as texts, whereas multidimensional convolution layers are used primarily for multidimensional objects such as image or video data (Chollet et al, 2022).

I also add a global maximum pooling layer to the architecture in order help the model select the most relevant features from the output of the convolution layer, by selecting the maximum value across all the filter outputs in order to reduce dimensionality and select the most important features. The global max pooling layer has also been identified in the literature as being particularly helpful in reducing overfitting (Zulqarnain et al 2020).

The architecture is as follows:

A one-dimensional convolution layer which uses the '*relu'* activation function and contains 128 filters and a kernel size of 5 and included a global max pooling layer. I kept the four unit final dense layer for classification, and also retained the loss (categorical cross-entropy) and optimiser functions (Adam with the learning rate of 0.001).

Activation Function

The relu activation function is the default function and is often a popular choice (Bengio 2012, Bartz et al 2023). The activation function decides how the input values of each node are translated into an output and influences a network's ability to approximate nonlinear functions (Batz et al 2023).

```
Model 6
_________________________________________________
 Layer (type)              Output Shape          Param #
=================================================
 embedding_6 (Embedding)   (None, 373, 200)     1618600
 conv1d (Conv1D)           (None, 369, 128)     128128
 global_max_pooling1d      (None, 128)          0
 dense_6 (Dense)           (None, 4)             516
=================================================
Total params: 1,747,244
```

```
Trainable params: 1,747,244
Non-trainable params: 0
_________________________________________________________________
```

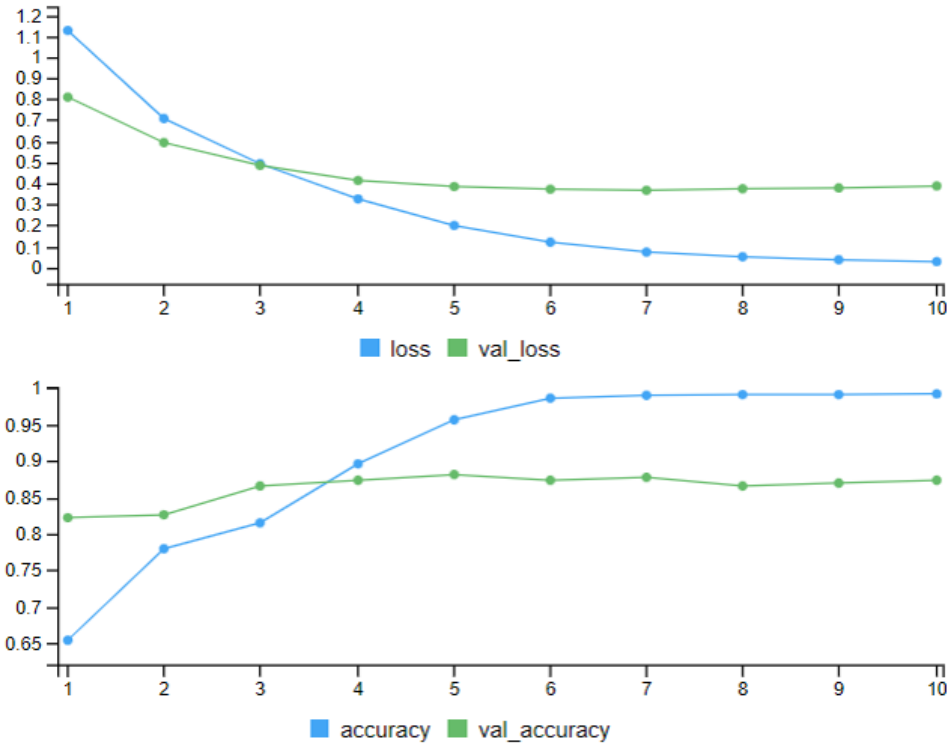


Figure 8 Keras Model 6 Loss and Accuracy

As can be seen in Figure 8, Model 6 achieved good results, a maximum validation accuracy of 88% and a running time of 29 seconds. These results are similar to the best performing model with the LSTM layer (Model 3).

Model 7

CNN, adjusted filters

For convolution layers, the kernel size and filters are important for capturing the relevant patterns in the data.

Model 7 uses the same architecture as model 6 other than an adjusted filter size of 64.

This adjustment slightly lowered the accuracy to 0.869 as can be seen in Figure 9.

Model 7

Layer (type)	Output Shape	Param #
embedding_7 (Embedding)	(None, 373, 200)	1618600
conv1d_1 (Conv1D)	(None, 369, 64)	64064
global_max_pooling1d_1	(None, 64)	0
dense_7 (Dense)	(None, 4)	260

Total params: 1,682,924

Trainable params: 1,682,924

Non-trainable params: 0

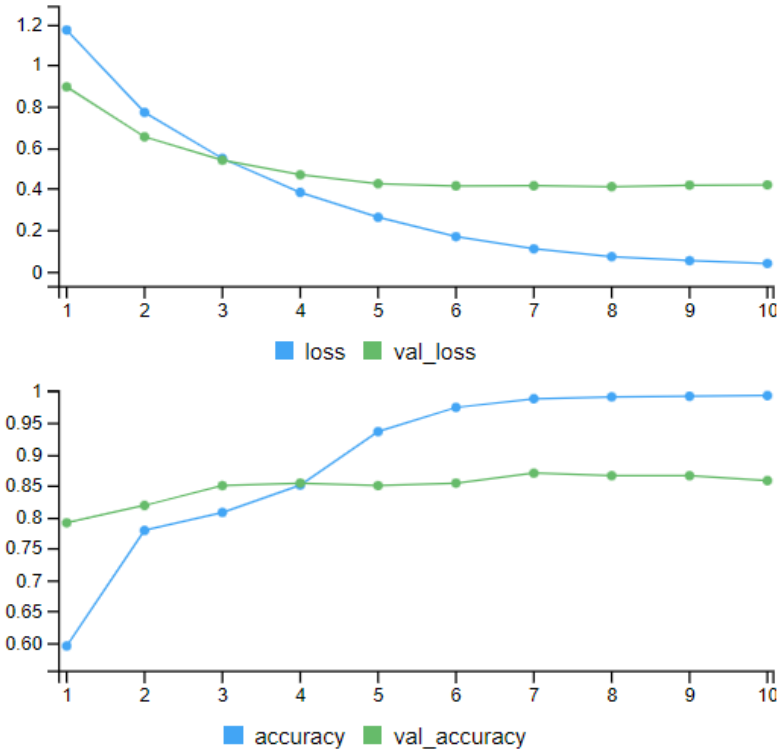


Figure 9 Keras Model 7 Loss and Accuracy

Model 8

CNN decreased kernel size.

Kernal Size

An increased kernel width increases the number of parameters in the model (Goodfellow et al, 2016). A wider kernel results in a narrower output dimension, which would heavily reduce the model capacity if there is no zero-padding in the model. Narrower output reduces the memory cost but the caveat is that the wider kernel required for this output uses more memory for parameter storage, and an increased runtime (Goodfellow et al, 2016). Smaller kernel sizes are better at capturing the relationships between nearby words, i.e. phrases that are important to the classification task. Larger kernels, however, can capture the broader relationships/dependencies and span a greater distance in the text.

Model 8 reverted to the architecture of the initial CNN with 128 filters which had an accuracy of 88%, but this time with a smaller kernel size: decreasing it from 5 to 3.

This model (Model 8) obtains the highest accuracy score so far, nearly reaching 89% (0.8893)

```
Model 8
_________________________________________________________
 Layer (type)              Output Shape           Param #
=========================================================
 embedding_8 (Embedding)  (None, 373, 200)       1618600
 conv1d_2 (Conv1D)        (None, 371, 128)       76928
 global_max_pooling1d_2 (  (None, 128)           0
 dense_8 (Dense)          (None, 4)              516
=========================================================
Total params: 1,696,044
Trainable params: 1,696,044
Non-trainable params: 0
_________________________________________________________
```

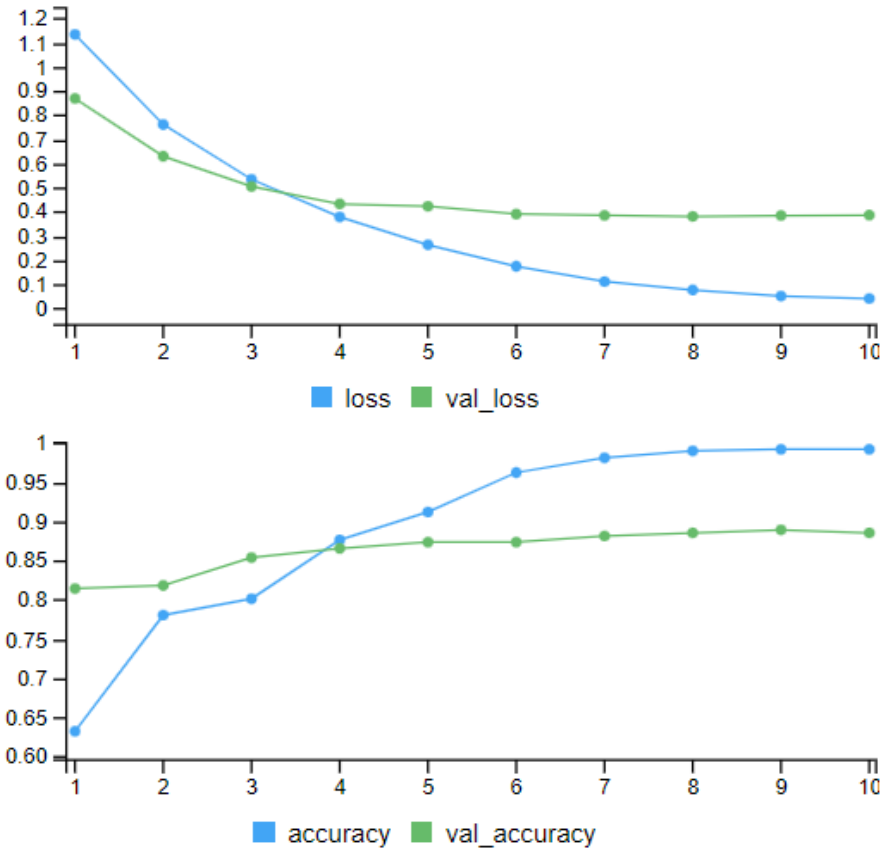


Figure 10 Keras Model 8 Loss and Accuracy

Model 9

CNN increased kernel size

Model 9 uses the same architecture as the model before, other than an increased kernel size of 7 – and the results were exactly the same. Despite the literature on kernel size, my models showed that slightly different kernel sizes made very little difference to my model's predictive accuracy.

```
Model 9
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_9 (Embedding)      (None, 373, 200)          1618600
 conv1d_3 (Conv1D)            (None, 367, 128)          179328
 global_max_pooling1d_3       (None, 128)               0
 dense_9 (Dense)              (None, 4)                 516
=================================================================
Total params: 1,798,444
Trainable params: 1,798,444
Non-trainable params: 0
```

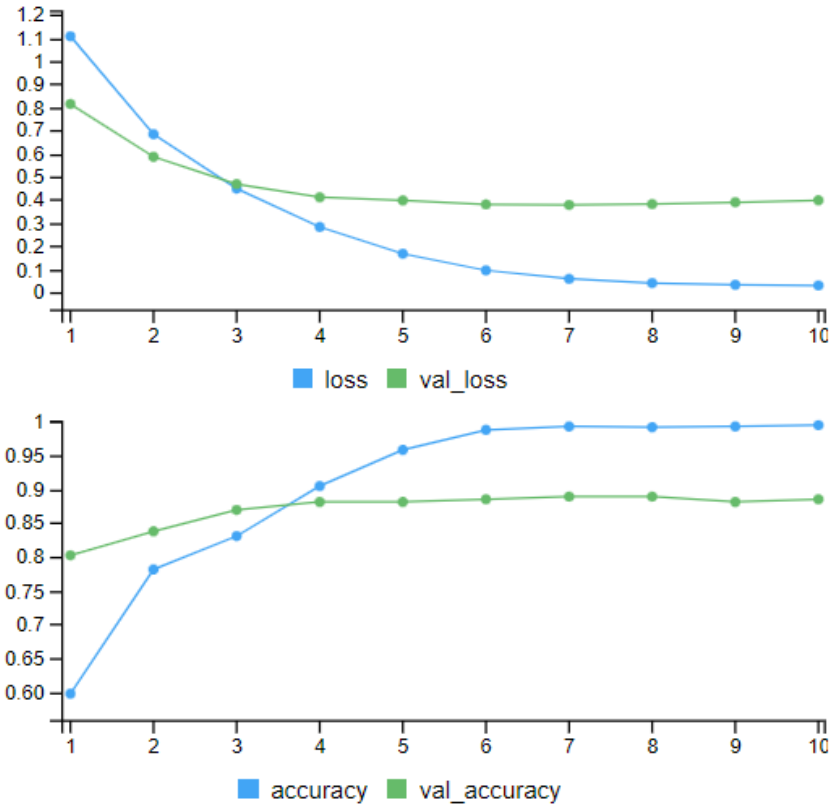


Figure 11 Keras Model 9 Loss and Accuracy

Model 9 also achieved a maximum validation accuracy of 0.8893.

Model 10

CNN increased filter

Thus far, the CNN model with a smaller filter had performed worse than one with a bigger filter size of 128. I then further tested this by running the model with an even larger filter size of 256. It obtained the same accuracy as the previous models of 0.8893

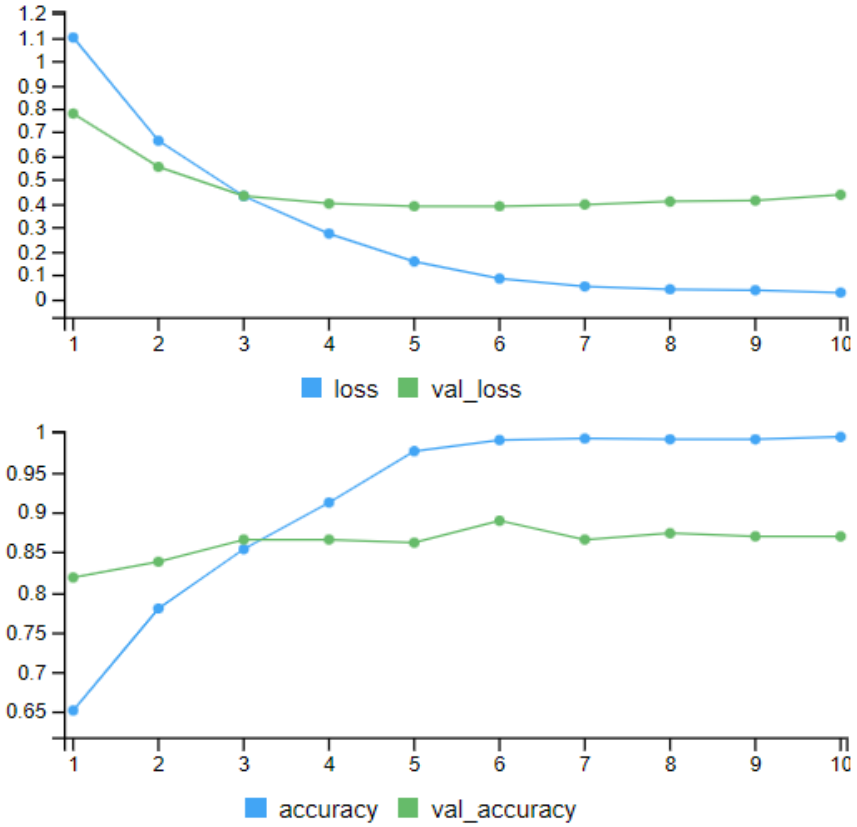


Figure 12 Keras Model 10 Loss and Accuracy

```
Model 10
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_10 (Embedding)     (None, 373, 200)          1618600
 conv1d_4 (Conv1D)            (None, 369, 256)          256256
 global_max_pooling1d_4       (None, 256)               0
 dense_10 (Dense)             (None, 4)                 1028
=================================================================
Total params: 1,875,884
Trainable params: 1,875,884
Non-trainable params: 0
```

Model 11

CNN plus LSTM

After these successful results on both a classic RNN with an LSTM layer, and then with a convolutional layer, I then ran a combination model which keeps the convolutional layer and also adds back the LSTM layer with 64 units. However, it got quite poor results, with under 84% accuracy as can be seen in Figure

13, performing worse than both the LSTM model and the CNN model, and with slower running speed of 1.86 minutes.

```
Model 11
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_11 (Embedding)     (None, 373, 200)          1618600
 conv1d_5 (Conv1D)            (None, 369, 128)          128128
 lstm_6 (LSTM)                (None, 369, 64)           49408
 global_max_pooling1d_5       (None, 64)                0
 dense_11 (Dense)             (None, 4)                 260
=================================================================
Total params: 1,796,396
Trainable params: 1,796,396
Non-trainable params: 0
```

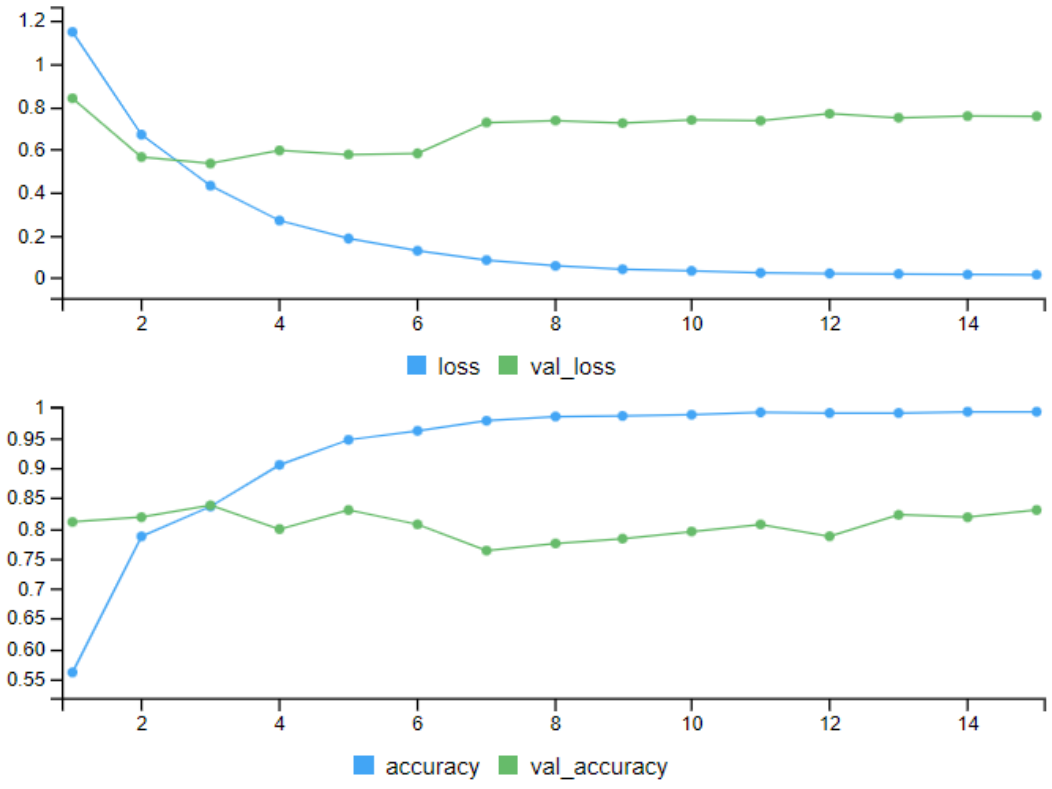


Figure 13 Keras Model 11 Loss and Accuracy

Model 12

CNN with dropout

Dropout Rate

The dropout layer is a hyperparameter used for model regularisation, i.e. to prevent overfitting (Bartz et al 2023). It works by randomly setting a percentage of the layers output features to zero during training so that random nodes/units are dropped out during training (Bartz et al 2023, Chollet et al 2022). Defining the dropout parameter when setting up the network design means defining the probability that a node will be removed.

A decreased dropout rate, i.e. dropping more units often, allows the remaining units more opportunity to interact in a way that improves model fit, which improves the model capacity (Goodfellow et al 2016).

In order to try to improve the CNN model I defined a dropout rate in the model architecture. I tested a dropout rate of 0.3, 0.4 and 0.5 and found 0.4 to be significantly better.

The architecture for Model 12 repeats the design of the high scoring CNN models, with the only difference being the dropouts. This model is compiled in the same way as the others (with the Adam optimiser and a learning rate of 0.001). I trained it on 15 epochs, but the highest accuracy was found at the 10th epoch. This is the first model that scored up to 90% validation accuracy.

```
Model 12
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_12 (Embedding)    (None, 373, 200)          1618600
 conv1d_6 (Conv1D)           (None, 369, 128)          128128
 dropout (Dropout)           (None, 369, 128)          0
 dropout_1 (Dropout)         (None, 369, 128)          0
 global_max_pooling1d_6      (None, 128)              0
 dropout_2 (Dropout)         (None, 128)               0
 dense_12 (Dense)            (None, 4)                 516
=================================================================
Total params: 1,747,244
Trainable params: 1,747,244
Non-trainable params: 0
_________________________________________________________________
```

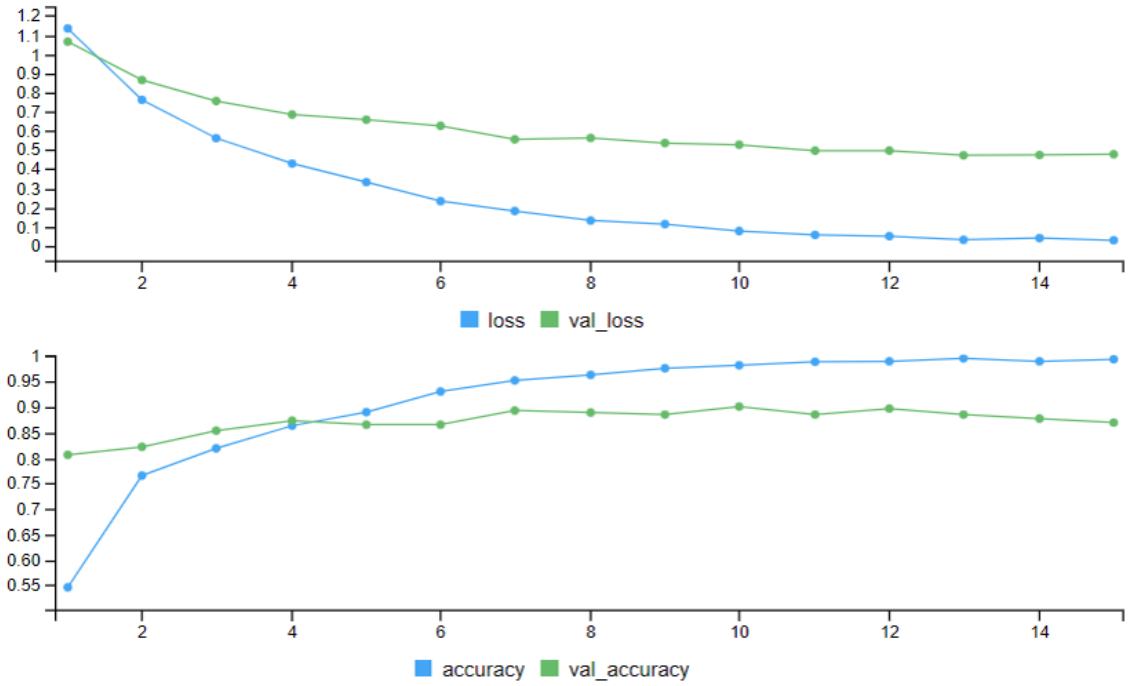


Figure 14 Keras Model 12 Loss and Accuracy

Model 13

CNN with smaller dropout rate

Model 13 is exactly the same as model 12, but this time I used a dropout rate of 0.3. The results of this were worse than with the 0.4 rate, achieving only 0.8853 accuracy

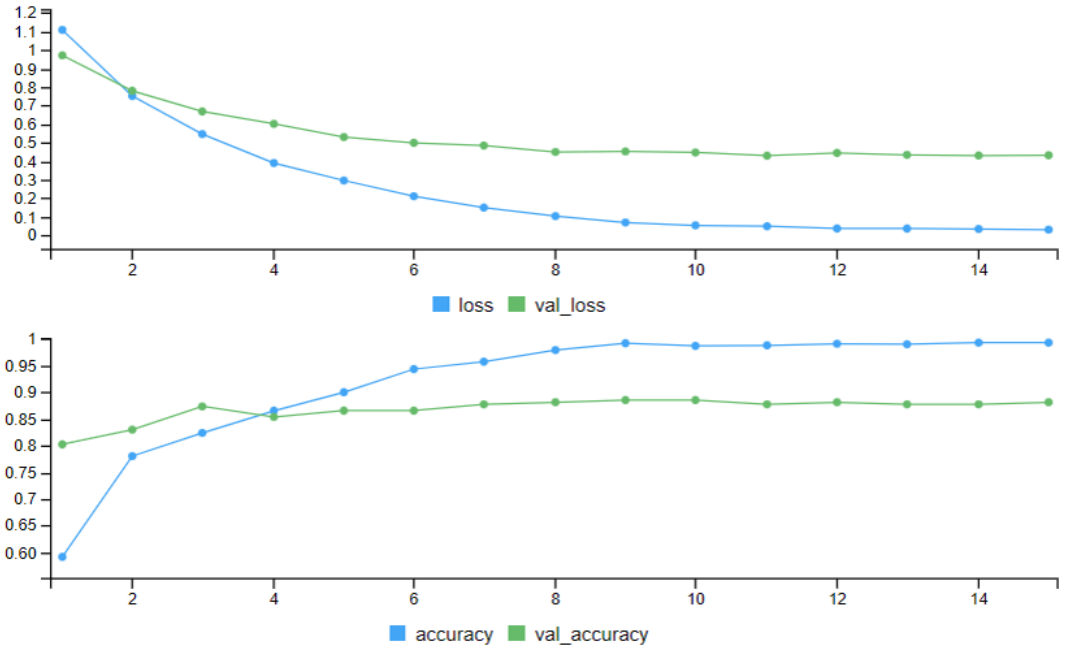


Figure 15 Keras Model 13 Loss and Accuracy

```
Model 13
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_13 (Embedding)    (None, 373, 200)          1618600
 conv1d_7 (Conv1D)           (None, 369, 128)          128128
 dropout_3 (Dropout)         (None, 369, 128)          0
 dropout_4 (Dropout)         (None, 369, 128)          0
 global_max_pooling1d_7      (None, 128)               0
 dropout_5 (Dropout)         (None, 128)               0
 dense_13 (Dense)            (None, 4)                 516
=================================================================
Total params: 1,747,244
Trainable params: 1,747,244
Non-trainable params: 0
_________________________
```

Model 14

More hidden layers

Bengio (2012) recommends the largest possible number of layers. They show that that performance improves with model size, provided appropriate regularisation methods are in place (Bengio, 2012). Layers are expected to interact with units and dropout, because together these parameters determine the number of nodes in the network (Bartz et al, 2023).

Because of the recommendations from the literature, I also tried running a more complex model with more hidden layers.

The architecture is as follows:

One embedding layer, followed by a one-dimensional convolution layer (with 128 filters and kernel size of 5 using the relu activation function), with dropout with a rate of 0.4. This is followed by another one-dimensional convolution layer (with 32 filters, kernel size 3, and relu activation), also with a dropout rate of 0.4. Thereafter there is a global max pooling layer with dropout (rate 0.4) and then a dense layer of 64 units with dropout (rate 0.4) that uses the relu activation). Finally, the final dense layer for classification into the 4 classes (ie 4 units) that uses the softmax activation function.

I then compiled that model as the others, with the Adam optimiser and the custom learning rate of 0.001 and trained it for 15 epochs.

Although this model has the most complex architecture, it scores poorly with an accuracy of under 80% (0.7944) and a slower running speed.

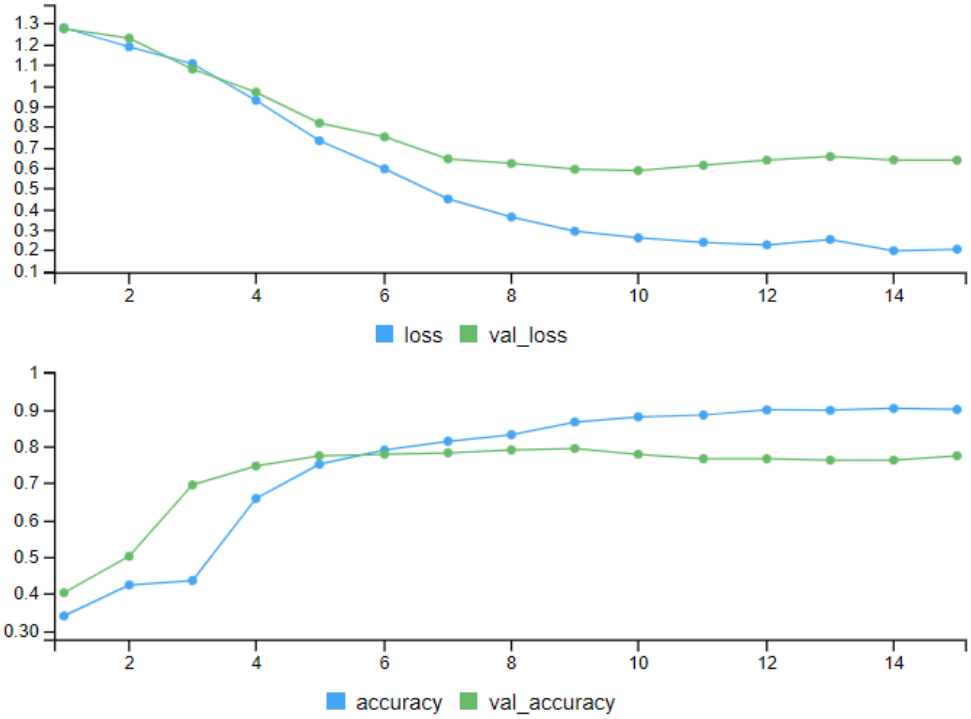


Figure 16 Keras Model 14 Loss and Accuracy

```
Model 14
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_14 (Embedding)     (None, 373, 200)          1618600
 conv1d_8 (Conv1D)            (None, 369, 128)          128128
 dropout_6 (Dropout)          (None, 369, 128)          0
 conv1d_9 (Conv1D)            (None, 367, 64)           24640
 dropout_7 (Dropout)          (None, 367, 64)           0
 conv1d_10 (Conv1D)           (None, 365, 32)           6176
 dropout_8 (Dropout)          (None, 365, 32)           0
 dropout_9 (Dropout)          (None, 365, 32)           0
 global_max_pooling1d_8       (None, 32)                0
 dropout_10 (Dropout)         (None, 32)                0
 dense_14 (Dense)             (None, 64)                2112
 dropout_11 (Dropout)         (None, 64)                0
 dense_15 (Dense)             (None, 4)                 260
=================================================================
Total params: 1,779,916
Trainable params: 1,779,916
Non-trainable params: 0
_________________________________________________________________
```

This shows that adding additional hidden layers and making the model more complex in that way actually negatively affects the results, rather than improving them.

3.7 Traditional Models

To generate comparable results between my models, my data needed to be exactly the same. However, the different requirements of the different types of models do mean that slightly different pre-processing is needed.

Decision trees, random forests and Naïve Bayes are not able to consider word order and word relationships in the same way that neural networks do, rather they focus on the occurrence or lack of the individual tokens (words). For this reason, the data needs to be reshaped. The format of the data processed for the neural network by the tokenizer function takes the form of a matrix object of 373 columns where each column represents a possible word in a document (text) sequence of up to 373 words, and each row shows the token indexes of the specific words in the document (5 000 words). Shorter text sequences were padded with zeros to make sequences of equal lengths as explained in Section 3.5.1.

To reshape the data into a format suitable for training traditional models, I unpadded the data sequences by removing the zero padding- and put them into a matrix of counts; with 5 000 columns (representing each word/token) and 1 011 rows representing each document(text) in the training data, where the counts are represented under each column according to how often they appear in each text. I then made the labels into a factor (rather than using the one-hot encoded labels).

3.7.1 Random Forests

The random forest is extremely simple to run using the randomForest package (Liaw and Weiner, 2022) available in R. The randomForest function implements Breiman's random forest algorithm for classification and regression. In this case, classification was specified, and the only arguments used were the pre-processed training data and the factorised training labels.

The random forest model achieved a prediction accuracy of almost 87% which is far higher than I had hypothesised, considering that it does not account for word order.

This model is a collection of 500 decision trees. The results output in the form of confusion matrices are included below:

Random forest model outcome and statistics from training set:

Type of random forest	classification
Number of trees	500
No. of variables tried at each split	70
OOB estimate of error rate	17.41%

Accuracy	0. 8696
95% Confidence Interval	(0.8217, 0.9085)
No Information Rate	0.4822
P-Value [Acc > NIR]	< 2.2e-16
Kappa	0.7752

Random forest model training confusion matrix

CLASSES	0	1	2	3	CLASS ERROR
0	363	22	2	1	0.06443299
1	19	422	2	4	0.05592841
2	9	46	28	1	0.66666667
3	8	61	1	22	0.76086957

3.7.2 Decision Trees

The decision tree model also needed the unpadded sequences of input data in the form of a data frame and the factorised labels, I use the rpart (Therneau and Atkinson, 2022) library and function to train the decision tree model, which is a variation of Breiman et al (1984). To analyse the predictions generated by the tree I specify type= 'class' to ensure the model performs the class classification task rather than regression.

The decision tree model achieved 81% accuracy on the validation data.

Results below show the first 3 nodes in the decision tree model:

```
Node number 1: 1011 observations,    complexity param=0.3315603
  predicted class=1  expected loss=0.5578635  P(node) =1
    class counts:   388   447    84    92
   probabilities: 0.384 0.442 0.083 0.091
  left son=2 (207 obs) right son=3 (804 obs)
  Primary splits:
      V10 < 0.5 to the right, improve=126.65350, (0 missing)
      V3  < 0.5 to the left,  improve=123.04540, (0 missing)
      V2  < 0.5 to the left,  improve=111.22980, (0 missing)
      V8  < 0.5 to the right, improve= 94.77901, (0 missing)
      V17 < 0.5 to the right, improve= 75.44493, (0 missing)
  Surrogate splits:
      V8   < 0.5 to the right, agree=0.962, adj=0.816, (0 split)
      V41  < 0.5 to the right, agree=0.827, adj=0.155, (0 split)
      V35  < 0.5 to the right, agree=0.821, adj=0.126, (0 split)
      V189 < 0.5 to the right, agree=0.818, adj=0.111, (0 split)
      V80  < 0.5 to the right, agree=0.817, adj=0.106, (0 split)

Node number 2: 207 observations
  predicted class=0  expected loss=0.05797101  P(node) =0.2047478
    class counts:   195     8     3     1
```

```
   probabilities: 0.942 0.039 0.014 0.005

Node number 3: 804 observations,    complexity param=0.09929078
  predicted class=1  expected loss=0.4539801  P(node) =0.7952522
    class counts:   193   439    81    91
   probabilities: 0.240 0.546 0.101 0.113
  left son=6 (56 obs) right son=7 (748 obs)
  Primary splits:
      V21 < 0.5 to the right, improve=54.09035, (0 missing)
      V3  < 0.5 to the left,  improve=48.88284, (0 missing)
      V2  < 0.5 to the left,  improve=46.81114, (0 missing)
      V17 < 0.5 to the right, improve=46.53088, (0 missing)
      V29 < 0.5 to the right, improve=45.42338, (0 missing)
  Surrogate splits:
      V364  < 0.5 to the right, agree=0.947, adj=0.232, (0 split)
      V562  < 0.5 to the right, agree=0.940, adj=0.143, (0 split)
      V174  < 1.5 to the right, agree=0.938, adj=0.107, (0 split)
      V257  < 0.5 to the right, agree=0.938, adj=0.107, (0 split)
      V1665 < 0.5 to the right, agree=0.938, adj=0.107, (0 split)
```

Unlike the neural networks, the decision tree model provides information on variable importance. As depicted above, the tree's primary splits were according to variables 10, 3,2,8, and 17 which are the terms: Montreux (10), Nations (3), United (2), Document (8), and Companies (17). I will expand further on this in the findings chapter.

Decision tree model statistics:

Accuracy	0.8142
95% Confidence Interval	(0.7607, 0.8602)
No Information Rate	0.4822
P-Value [Acc > NIR]	< 2.2e-16
Kappa	0.6852

3.7.3 Naïve Bayes

Because the Naïve Bayes classifier works by identifying single terms, it is unlikely to work on a complex classification task like this project. However, in order to test that, I have included it in this study.

This Naïve Bayes Classifier was built in the e1071 package (Meyer et al, 2023).

Using the exact same data as for the previous models, the Naïve Bayes reached only 5.9% accuracy

Naïve Bayes Overall statistics:

Accuracy	0.0593
95% Confidence Interval	(0.0336, 0.0959)
No Information Rate	0.4822
P-Value [Acc > NIR]	1
Kappa	0.0089

The Naïve Bayes classifier was entirely unable to classify this form of data, achieving less than 6% classification accuracy. To examine this in closer detail the confusion matrix below depicts the model's predictions on the validation set. The matrix shows that the Naïve bayes predicted mostly '3's and was unable to predict any of the other classes with sufficient accuracy. This can be verified by comparing the confusion matrix with the actual labels which are: class 0= 96, class 1 = 122, class 2= 23 and class 3= 12. A more detailed table with this information is in Section 4.2.1

Table 9 Naive Bayes Predictions Confusion Matrix

REFERENCE					
PREDICTION		**0**	**1**	**2**	**3**
	0	0	0	0	0
	1	0	0	0	0
	2	9	6	4	1
	3	87	116	19	11

4. Findings

This section presents a table of comparable results from all the models described in the Methodology section, and then provides some discussion and analysis of the different models. Because of the size of the table I have separated it into 3 smaller tables (Tables 11, 12, and 13) in order to fit into this document. Following the table, Section 4.2. provides a description of some of the notable findings which is followed by a more detailed discussion of the findings relating to variable importance and class-specific analysis.

4.1 Table of Findings

Table 10 Summary of Models 1-5

Description of adjustment	LSTM model	Increase units	Custom learning rate	Further increase units	Change learning rate
Name	**Model 1**	**Model 2**	**Model 3**	**Model 4**	**Model 5**
Type	RNN	RNN	RNN	RNN	RNN
Layers:					
Embedding layer output dimension	200	200	200	200	200
LSTM layer units	32	64	64	96	64
1D Conv layer activation					
Number of Filters					
Kernel size					
Other layers					
Other layers					
Other layers					
Pooling Layer					
Dropout Rate					
Final dense layer units	4	4	4	4	4
Activation	softmax	softmax	softmax	softmax	softmax
Compilation:					
Loss	categorical crossentropy	categorical crossentropy	categorical crossentropy	categorical crossentropy	categorical crossentropy
Optimizer	adam	adam	adam	adam	adam
Customised learning rate			0.001	0.001	0.0001
Training:					
Epochs	6	20	11	11	15
Batch size	32	32	32	32	32
Accuracy	**0.869**	**0.874**	**0.885**	**0.822**	**0.8458**
Time	34.25sec	1.93min	1.11min	2.9min	1.5min

Table 11 Summary of Models 6-11

Description of adjustment	CNN model	Adjust filter size	Decrease kernal	Increase kernal	Increase filter	Combine CNN and LSTM layers
Name	**Model 6**	**Model 7**	**Model 8**	**Model 9**	**Model 10**	**Model 11**
Type	CNN	CNN	CNN	CNN	CNN	CNN and LSTM
Layers:						
Embedding layer output dimension	200	200	200	200	200	200
LSTM layer units						64
1D Conv layer activation	relu	relu	relu	relu	relu	relu
Number of Filters	128	64	128	128	256	128
Kernel size	5	5	3	7	5	5
Other layers						
Other layers						
Other layers						
Pooling Layer	global max	global max	global max	global max	global max	global max
Dropout Rate						
Final dense layer units	4	4	4	4	4	4
Activation	softmax	softmax	softmax	softmax	softmax	softmax
Compilation:						
Loss	categorical crossentropy	categorical crossentropy	categorical crossentropy	categorical crossentropy	categorical crossentropy	categorical crossentropy
Optimizer	adam	adam	adam	adam	adam	adam
Customised learning rate	0.001	0.001	0.001	0.001	0.001	0.001
Training:						
Epochs	10	10	10	10	10	15
Batch size	32	32	32	32	32	32
Accuracy	**0.8814**	**0.8695**	**0.889**	**0.889**	**0.889**	**0.8379**
Time	29sec	27 sec	25sec	34 sec	39sec	1.8min

Table 12 Summary of Models 12-14 and Traditional Models

Description of adjustment	CNN with dropout	Adjusted dropout rate	More hidden layers	Traditional Machine Learning Models		
Name	Model 12	Model 13	Model 14	Decision Tree	Random Forest	Naïve Bayes
Type	CNN	CNN	CNN with more layers			
Layers:						
Embedding layer output dimension	200	200	200			
LSTM layer units						
1D Conv layer activation	relu	relu	relu			
Number of Filters	128	128	128			
Kernel size	5	5	5			
Other layers			1D Conv (64 filters, kernal size 3, dropout)			
Other layers			1D Conv (32 filters, kernal size 3, dropout)			
Other layers			dense (4 units, relu activation, dropout)			
Pooling Layer	global max, with dropout	global max, with dropout	global max, with dropout			
Dropout Rate	0.4	0.3	0.4			
Final dense layer units	4, withdropout	4, withdropout	4, withdropout			
Activation	softmax	softmax	softmax			
Compilation:						
Loss	categorical crossentropy	categorical crossentropy	categorical crossentropy			
Optimizer	adam	adam	adam			
Customised learning rate	0.001	0.001	0.001			
Training:						
Epochs	15	15	15			
Batch size	32	32	32			
Accuracy	**0.901**	**0.885**	**0.794**	**0.8142**	**0.8695**	**0.05928**
Time	57sec	53sec	1.113min			

4.2 Description of Findings

Although the data set tested here is small and only focuses on classifying references to the United Nations, we can see that machine learning algorithms, in particular deep neural networks, are able to classify the textual data into 4 different classes with up to 90% accuracy.

As summarised in the results table(s), Models 1-5 are all different iterations of RNNs with LSTM layers, each with some adjustments in the model's architecture and hyperparameters.

All the RNN (LSTM) models achieve above 80% accuracy. This high result already indicates the potential of deep learning techniques for classifying this sort of data.

Although the CNN models 6-14 in general reached higher predictive accuracy than the RNN models 1-5, Model 3 of the RNNs did perform very well, predicting the validation data with over 88% accuracy. This model had very straightforward architecture, close to a 'default' model design for text classification with the only advancement from 'basic' architecture being a customised learning rate of 0.001, which improved the model's prediction from the RNNs without the customized learning rate.

The CNN models, (number 6-14) scored between 79% and 90% accuracy on classifying the validation data.

Complexifying the model, i.e. combining LSTM and convolution layers (see Model 11) or adding many more hidden layers (see Model 14), did not yield comparatively good results. In fact, these two most complex models of the 14 neural networks I tested, scored the lowest overall. Model 11 and Model 14, which both had more layers than the other 12 deep learning models, achieved approximately 84% and 79% accuracy respectively, whereas most of the simpler neural network models achieved over 85%.

The best performing algorithm overall was Model 12. Using one convolution layer in a sequential neural network model and including a dropout rate in each layer of 0.4 was found to be most effective in maximising the model's predictive accuracy.

As illustrated below in Table 14, which zooms in on Model 12, this model contains the embedding layer that was used in all the models with an output dimension of 200, followed by one one-dimensional convolution layer (with 128 filters and 5 kernels) with a dropout rate of 0.4, followed by a global maximum pooling layer (also with a specified dropout rate 0.4). The final dense layer of 4 units with the softmax activation function (I also specify a dropout rate of 0.4) is for classification into the 4 classes of deference.

Description of adjustment	CNN with dropout
Name	**Model 12**
Type	CNN
Layers:	
Embedding Layer Output Dimension	200
1D Convolution Layer Activation	relu
Number of Filters	128
Kernel Size	5
Pooling Layer	global max, with dropout
Dropout Rate	0.4
Final Dense Layer Units	4, with dropout
Activation	softmax
Compilation:	
Loss	categorical cross-entropy
Optimizer	adam
Customised Learning Rate	0.001
Training:	

Epochs	15
Batch size	32
Accuracy	**0.901**
Time	57sec

Table 13 Model 12 Architecture and Hyperparameters

This model was able to achieve 90% accuracy when testing it on the validation data set, a highly significant result which indicates great statistical potential for these classification methods.

Regarding the traditional models, the random forest model also achieved impressive results especially considering that, unlike neural networks, it does not account for word order. Although the literature had implied that the deep learning models could score significantly higher predictive accuracy results than traditional models, the results here show that in fact, some of the traditional methods can compete quite successfully with deep learning methods. The random forest model, which is a 'random' collection or 'forest' of many decision trees (in this case 500), is able to achieve such good results by having the many individual trees 'vote' for the preferred outcome class (Breiman, 2001). The randomisation aspect of the model means that the sample is more diverse and can generalise better to noisy data.

Looking at the successes of both the random forest model and the neural network with dropout, one can deduce that the 'randomisation' element in machine learning models seems to be particularly important for this dataset. Although random forest algorithms are substantially different to deep neural networks with dropout, there are some levels of similarities. The dropout rate – i.e. the probability of dropping a number of randomly selected nodes in a neural network, helps prevent overfitting by forcing the layers to adjust based on a more probabilistic approach.

That being said, the single decision tree model was also statistically quite successful - achieving an over 80% predictive accuracy on the validation data set, despite using only one decision tree, and therefore not including the randomisation element of the random forest algorithm.

Not surprisingly, the Naïve Bayes model was not accurate and almost only predicted '3s' in the validation set, giving an accuracy of 6%, proving that this is the wrong type of machine learning algorithm for this task and that simply inputting the data into 'any' machine learning algorithm is not viable.

4.2.1 Variable Importance

Unlike the 'black box' deep learning algorithms which do not show how the model came to its result, the traditional models can output some of the reasons for their classification decisions. As can be seen in the tree output in section 3.7.2, the root node at the top of the tree which is the first node to split the entire

tree uses variable 10, and its next splits are based on variables 3,2,8, and 17. By and cross referencing these indices in the tokenizer index we can see that these are the terms:

Variable Index	Term
10	'montreux'
3	'nations'
2	'united'
8	'document'
17	'companies'

Table 14 Decision Tree Primary Split Variables and Actual Terms

It makes sense that the terms like 'united and 'nations' i.e., reflecting United Nations were a big part in determining the tree's primary split and it is interesting to note that those were not included in the decision tree's variable importance output: decision trees also qualify other variables that aren't necessarily used to split the tree, as 'important' for their classification tasks.

The terms determined by the single decision tree model as most important are:

Variable Index	Term
10	'montreux'
8	'document'
21	'icoc'
41	'pmscs'
17	'companies'
118	'parties'
9	'national'
35	'practices'

Table 15 Decision Tree Variable Importance: Actual terms

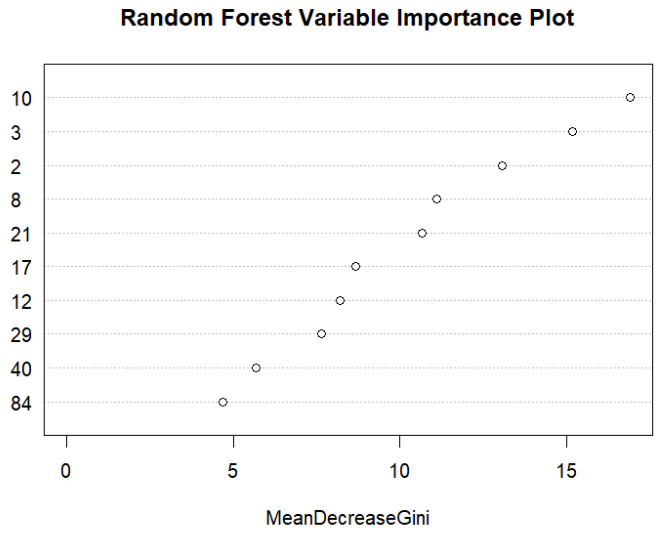


Figure 17 Random Forest Variable Importance Plot (Variable index number on y axis)

Similarly, with the random forest model, by cross referencing the variable importance plot (above) with the numeric indexes in the tokenizer term list, we can see that the model found the following terms (in descending order) most important:

Variable Index	Term
10	'montreux'
3	'nations'
2	'united'
8	'document'
21	'icoc'
17	'companies'
29	'icoca'
40	'undp'
84	'association'

Table 16 Random Forest Variable Importance and Actual Terms

'Montreux' refers to the Montreux Document Process which was later transformed into the Montreux Document Forum (MDF) which was one of the major international institutions that was created in the early 2000s to regulate private security, during the historic shift noted by Westerwinter (2022) after 'the Blackwater scandal', the high profile killing of Iraqi civilians by a group of private military contractors employed by an American private military company called Blackwater USA. Another of the key institutions in the global Private Security Governance complex is the International Code of Conduct for Private Security Service Providers' Association (ICoCA) which is represented in the terms 'icoc' (term 21)

and ‘icoca’ (term 29). ICOCA and the MDF are identified by Westerwinter (2022) as two of the biggest non-United Nations institutions in the complex. It is possible therefore that the term ‘document’ here also refers to the Montreux Document Forum. ‘Document’ is used in the decision tree early splits as well as identified by both decision tree and random forest in their Variable Importance output. The term ‘PMSCS’ refers to ‘private military and security companies’ which these institutions govern. This term was identified as important by the single decision tree but not the random forest but both models identified ‘companies’ as an important term. UNDP, which the random forest found an important term, references a United Nations affiliated body, The United Nations Development Programme. The traditional models are not able to identify ‘united’ and ‘nations’ as part of the same (sequential) phrase and therefore includes them as different variables. Despite this, the decision tree and random forest models were able to classify the majority of texts accurately.

It is also interesting to note that as can be seen in Table 6 Most Common Words and Their Frequencies’, Montreux is only the 10th most frequent word, with ‘security’, ‘united, ‘nations’, ‘international’, ‘rights’, ‘human’, ‘states’, ‘document’ and ‘national’ all occurring more frequently in the texts. We can therefore see that the traditional models therefore do not split based simply on the most common words but rather find more informative ways of choosing which terms to prioritise.

4.2.2 Class Specific Analysis

The accuracy above refers to the proportion of validation examples correctly classified. By this standard 45% accuracy can be achieved by assuming all responses were Group 2. Looking into the nature of the errors is therefore necessary in order to identify whether or not any category or response was particularly poorly coded by the models. We can determine this by examining the confusion matrix – a record of how accurately each class of response was categorised.

Looking at the confusion matrix of the random forest training model below, we can see that the large majority of the data falls into classes 0 and 1, namely, if the United Nations (UN) was not referenced at all (0) or if it was simply a passing reference (1) rather than any deference of authority (2 or 3) (See Table 4 for the class descriptions).

Table 17 Random Forest model training data Confusion Matrix and Class Error

CLASSES	0	1	2	3	CLASS ERROR
0	363	22	2	1	0.06443299
1	19	422	2	4	0.05592841
2	9	46	28	1	0.66666667
3	8	61	1	22	0.76086957

This makes sense in terms of the success of a traditional model like a random forest or decision tree which should be good at classifying according to the existence or not of a reference to the UN, rather than the more complex classification task of determining how significant that reference is. By simply classifying most references as 1s, the model is able to score a high classification accuracy despite having quite a large error rate on the classification of the higher categories (2 and 3).

The table below (18) presents the actual class labels for the validation data and the tables that follow present confusion matrices of the predictions from some of the models which show how many of each class labels were predicted by each model on the validation data set. Cross referencing them with the actual labels below gives the class-specific information.

Table 18 Actual Class Labels for Validation Data Set

validation data set- actual labels				
	class 0	**class 1**	**class 2**	**class 3**
actual labels	96	122	23	12

Decision tree and random forest confusion matrices- predictions on validation data set

Table 19 Decision Tree Confusion Matrix

REFERENCE					
PREDICTION		**0**	**1**	**2**	**3**
	0	89	9	5	2
	1	7	106	11	6
	2	0	0	7	0
	3	0	7	0	4

Table 19 above shows that of the 96 texts that fall into class 0 ‘no reference to the UN’, 89 were correctly predicted by the decision tree model, and that 4 of the 12 texts that deferred authority to the UN (class 3) were predicted correctly.

Table 20 Random Forest Confusion Matrix

REFERENCE					
PREDICTION		**0**	**1**	**2**	**3**
	0	93	5	1	0
	1	3	106	12	9
	2	0	0	9	1
	3	0	1	1	2

Neural Network models 12 and 2 confusion matrices- predictions on validation data set:

Table 21 Model 12 Confusion Matrix

REFERENCE					
PREDICTION		0	1	2	3
	0	93	4	2	0
	1	2	112	7	4
	2	0	3	14	1
	3	0	2	2	7

The most successful neural network model (above) was able to get 7/12 category 3 labels and 14/23 category 2 labels correct.

Table 22 Model 2 Confusion Matrix

REFERENCE					
PREDICTION		0	1	2	3
	0	90	5	1	0
	1	6	113	7	5
	2	0	1	13	2
	3	1	4	1	5

The predictive accuracy of the random forest model was 86%: and the neural network Model 12 was 90% (See Table 13). However, regarding the classification categories 2 and 3, the random forest was only able to get 2 out of 12 of the category 3 labels correct, and 9 out of the 23 category 2 labels correct in the validation set (Table 20).

The neural network models generated slightly better predictions, in particular regarding categories 2 and 3. However even the highest scoring of the models, Model 12, was only able to predict class 2 with 60% accuracy and class 3 with 58% accuracy (Table 21). Because the data was so heavy on classes 1 and 2 in this subset, we cannot yet know for sure how successfully our model will be able to classify classes 2 and 3 with a larger training set. Because of the training/validation data split, the validation data used for these predictions comprised of only 253 labelled texts.

However, with the results we have, we can still see that a neural network model like Model 12, or even a straightforward model like Model 2 (Table 22) is better at predicting the more complicated classes regarding deference of authority, i.e., these algorithms are better able to differentiate between a passing reference to the UN and a reference deferring some power to the UN.

4.3 Summary of Findings

As a sample study investigating the possibility of developing a machine learning model for classifying textual data published by the 11 private security governance institutions according to levels of deference from organisation *k* to organisation *j,* achieving a classification model of 90% accuracy on a small data subset strongly indicates the potential for machine learning for these kind of classification tasks. Although the models in this research were trained and tested on just 1 264 hand-labelled texts, applying the model to the unlabelled data would generate a dataset of over 13 000 texts classified according to levels of deference. This dataset would make an extremely useful research tool for scholars of global governance.

The results of this research show that it is possible to use machine learning classification techniques to create a new data set that describes the levels of deference of authority between organisations in a Hybrid Institutional Complex. Getting up to 90% accuracy on the validation data using deep learning methods indicates quite clearly the potential of these methods for classification tasks based on large amounts of textual data from international governance complexes, which could open up many avenues for international relations research.

5. Limitations of the Study and Proposals for Future Work

As shown in the class specific analysis, accuracy is not always the best measure for testing a model's reliability because even the models with up to 90% classification accuracy were unable to accurately classify a lot of the more complex classes of data. Because the validation set only contained 12 class 3 labels and 23 class 2 labels, the 90% classification accuracy mostly relies on the correct classification of the simpler classes 0 and 1, which contained the majority of data points.

Additionally, by selecting accuracy as the primary measure against which to test the algorithms, this study relies on the accuracy of my hand-coding of the training data to be 100% accurate itself, which is impossible. As described in the methodology section, the hand classification process was difficult and due to expected human error cannot be entirely reliable.

Further, because overfitting is a big issue for machine learning models in general, despite testing some control mechanisms for reducing this, this study would be much more reliable if it used a test set to test the models on completely unseen data. However, one of the main limitations of this study is the relatively small size of hand-coded data with only 1264 hand labelled text sequences. Because of

insufficient training and testing data I was only able to split the data into a training and validation set, and not able to keep aside data for testing.

Further studies should therefore also include performing cross validation of data or hand coding another +400 lines of text and thus being able to include a separate test set in the evaluations.

A further limitation that resulted from the small size of training data was that the study only investigated a small thematic subset of the data, namely the data referencing the United Nations. However, this leaves potential for future studies which could include a multi-level, multi-class classification model that can classify the data for deference of authority by the 11 different organisations in the complex to each other, and according to the original 5 different deference levels.

Another limitation came from the original raw dataset having been somewhat pre-processed already, so I was not able to test the model based on inclusion of stop words for example. Further research could also test other classification algorithms that I excluded from this study due to its small scope, including the Transformer, and the Support Vector Machine.

6. Conclusions

This research proposed and tested different machine learning techniques for classification of the organisational output documents published by the institutions in the Private Security Governance Institutional Complex. By creating, running, and testing different classification models this paper has shown that machine learning algorithms can be successfully used to consolidate and classify a large and diverse range of international relations textual data, and that machine learning could be a useful tool for finding information about the relationships of authority between institutions in a Hybrid Institutional Complex. The most accurate classification model tested in this paper was a convolutional neural network model, but recurrent neural networks and a random forest algorithm also produced statistically significant results.

This paper classified 1 264 texts (from the 11 organisations in the governance complex) according to levels of deference to the authority of the United Nations, with up to 90% accuracy. The successful outcome of this small sample indicates great potential for these machine learning classification algorithms to further classify the levels of deference from each institution in the institutional complex to the other institutions and ultimately develop valuable quantitative data about relationships of authority in a global governance complex like this.

References

Abbott, K. and Faude, B. (2021) Hybrid institutional complexes in global governance. *Review of International Organizations* 17(2), 263–291 (2022).

Abbott, K. W., Genschel, P., Snidal, D., and Zangl, B. (2015). Orchestration: Global Governance through Intermediaries. In *International Organizations as Orchestrators*, ed. Kenneth W. Abbott, Philipp Genschel, Duncan Snidal and Bernhard Zangl. Cambridge: Cambridge University Press. 3–36.

Abrahamsen, R., and Williams, M. (2007). Securing the City: Private Security Companies and Non-State Authority in Global Governance*. International Relations* 21(2) 237-253.

Aggarwal, C. C. (2018). Neural Networks and Deep Learning: A Textbook. Springer: New York. https://doi.org/10.1007/978-3-319-94463-0

Allaire J, Chollet F (2022). _keras: R Interface to 'Keras'_. R package version 2.11.0, <https://CRAN.R-project.org/package=keras>.

Allaire J, Tang Y (2022). _tensorflow: R Interface to 'TensorFlow'_. R package version 2.11.0, <https://CRAN.R-project.org/package=tensorflow>.

Allaire, J.J, (2018) “Machine Learning with R and Tensorflow”. Key Note address by JJ Allaire at rsudio::conf2018. Available on Posit PBC Channel at: https://www.youtube.com/watch?v=atiYXm7JZv0&t=1s

Bauman, Z. (2000). *Liquid modernity*. Oxford: Polity Press.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade: Second Edition* (437-478). Springer: Berlin

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, *3*(6).

Berner, J., Grohs, P., Kutyniok, G., and Petersen, P. (2023). The Modern Mathematics of Deep Learning. In Grohs, P., and Kutyniok, G. (2023). *Mathematical Aspects of Deep Learning.* Cambridge: Cambridge University Press

Breiman, L. (2001) Random Forests. *Machine Learning.* 45:5–32. https://doi.org/10.1023/A:1010933404324

Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J. (1984). *Classification and Regression Trees.* Routledge: New York. https://doi.org/10.1201/9781315139470

Brosig, M. (2015). *Cooperative Peacekepping in Africa. Exploring Regime Complexity*. New York: Routledge.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*. 33, 1877-1901. Available from https://doi.org/10.48550/arXiv.2005.14165Chollet, F. (2018). *Deep Learning with Python.* Manning: Shelter Island. ISBN 9781617294433

Chollet, F., Kalinowski, T., and Allaire, J. (2022). Deep learning with R. Second Edition. Manning: Shelter Island. ISBN 9781633439849

Clark, R. (2021). Pool or Duel? Cooperation and Competition Among International Organizations. *International Organization*, 75(4), 1133-1153.

Dahl, G. E., Jaitly, N., & Salakhutdinov, R. (2014). Multi-task neural networks for QSAR predictions. *arXiv preprint arXiv:1406.1231*.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, *41*(6), 391-407.

Dogru, H., Hameed, A., Tilki, S., and Jamil, A. (2021). Comparative Analysis of Deep Learning and Traditional Machine Learning Models for Turkish Text Classification. Conference Paper available from Applied Artificial intelligence Lab on ResearchGate.net.

Dreyfus, S. (1962). The numerical solution of variational problems, Journal of Mathematical Analysis and Applications 5(1):30-45 ISSN 0022-247X

Eilstrup-Sangiovanni, M, and Westerwinter, O. (2021). The global governance complexity cube: Varieties of institutional complexity in global governance. *Review of International Organizations* 17 (2), 233–262 (2022).

Feinerer I, and Hornik K (2023). tm: Text Mining Package. R package version 0.7-11, https://CRAN.R-project.org/package=tm.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, *36*(4), 193-202.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press: Cambridge. Online version available at http://www.deeplearningbook.org.

Grimmer, J., and Stewart, B. M. (2013). Text as data: the promise and pitfalls of automatic content analysis methods for political texts. Political Analysis. 21: 267–297. doi: 10.1093/pan/mps028

Grohs, P., and Kutyniok, G (Eds). (2023). *Mathematical Aspects of Deep Learning*. Cambridge University Press: Cambridge.

Hale, T and Roger, C. (2014). Orchestration and Transnational Climate Governance. *Review of International Organizations* 9(1):59–82.

Henning, C., & Pratt, T. (2020): Hierarchy and Differentiation in International Regime Complexes: A Theoretical Framework for Comparative Research. Working Paper.

Hochreiter, S., and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

Hubel, D., and Wiesel, T. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, 124(3): 574–591.

Ikonomakis, M., Kotsiantis S., and Tampakas, V. (2005). "Text Classification Using Machine Learning Techniques," Proceedings of the 9th WSEAS International Conference on Computers. 8(4):966-974.

Kass, G. V. (1980). An Exploratory Technique for Investigating Large Quantities of Categorical Data. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*. 29(2):119–127.Kedziora, D. J., Musial, K., and Gabrys, B. (2020). Autonoml: Towards an integrated framework for autonomous machine learning. *arXiv preprint arXiv*:2012.12600.

Kelley, H. J. (1960). Gradient theory of optimal flight paths. *American Rocket Society Journal* 30(10):941-954 https://doi.org/10.2514/8.5282

Knowles-Barley, S., Jones, T. R., Morgan, J., Lee, D., Kasthuri, N., Lichtman, J. W., & Pfister, H. (2014). Deep learning for the connectome. *GPU Technology Conference* (26).

LeCun, Y. (1989). Generalization and network design strategies. *Connectionism in perspective*, *19*(143-155), 18.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), 541-551.

Lenat, D., and Guha, R.Y. (1988). The world according to CYC. (MCC Technical Report Number ACA-AI-300-88.)

Liaw, A., and Wiener, M. (2002). Classification and Regression by randomForest. R News 2(3), 18--22.

Mariel, W. C., Mariyah, S., and Pramana, S. (2018). Sentiment analysis: a comparison of deep learning neural network algorithm with SVM and naïve Bayes for Indonesian text. *Journal of Physics: Conference Series.* 971 (012049). DOI: 10.1088/1742-6596/971/1/012049

Maslej-Krešňáková, V., Sarnovský, M., Butka, P., and Machová, K. (2020). Comparison of Deep Learning Models and Various Text Pre-Processing Techniques for the Toxic Comments Classification. *Applied Sciences* 10 (23): 8631. https://doi.org/10.3390/app10238631

Meyer D, Dimitriadou E, Hornik K, Weingessel A, Leisch F (2023). _e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien_. R package version 1.7-13, <https://CRAN.R-project.org/package=e1071>.

Miikkulainen, R., & Dyer, M. G. (1991). Natural language processing with modular PDP networks and distributed lexicon. *Cognitive Science*, 15(3), 343-399.

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence.* 267: 1–38.

Monroe, B. L., and Schrodt, P. A. (2017). Introduction to the special issue: the statistical analysis of political text. *Political Anal*ysis. 16, 351–355. doi: 10.1093/pan/mpn017

Monroe, B. L., Colaresi, M. P., & Quinn, K. M. (2008). Fightin' Words: Lexical Feature Selection and Evaluation for Identifying the Content of Political Conflict. *Political Analysis,* 16(4), 372–403. http://www.jstor.org/stable/25791946

Mor-Yosef, S., Samueloff, A., Modan, B., Navot, D., & Schenker, J. G. (1990). Ranking the risk factors for cesarean: logistic regression analysis of a nationwide study. *Obstetrics & Gynecology*, *75*(6), 944-947.

Oberthur, S. and Stokke, O.S. (2011). Conclusions: Decentralized Interplay Management in an Evolving Interinstitutional Order. In *Managing Institutional Complexity: Regime Interplay and Global Environmental Change*, ed. Sebastian Oberthur and Olav Schram Stokke. Cambridge: MIT Press. 313–341.

Ombabi, H., Lazzez, O., Ouarda, W., and Alimi, A. M. (2017) "Deep learning framework based on Word2Vec and CNN for users interests classification," Sudan Conference on Computer Science and Information Technology (SCCSIT) 1-7. doi: 10.1109/SCCSIT.2017.8293054.

OpenAI. (2023). GPT-4 technical report. *arXiv2303-08774.*

Palanivinayagam, A., El-Bayeh, C. Z., and Damaševičius. R. (2023). Twenty Years of Machine-Learning-Based Text Classification: A Systematic Review. *Algorithms* 16(5): 236. https://doi.org/10.3390/a16050236

Pratt, T. (2018). Deference and Hierarchy in International Regime Complexes. *International Organization*s, 72(3), 561-590.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6): 386.

Rumelhart, D., Hinton, G. and Williams, R. (1986). Learning representations by back-propagating errors. *Nature* 323, 533–536 https://doi.org/10.1038/323533a0

Sahami, M., Dumais, S.T., Heckerman, D.E., & Horvitz, E. (1998). A Bayesian Approach to Filtering Junk E-Mail. *AAAI Conference on Artificial Intelligence*.

Shah, K., Patel, H., Sanghvi, D., and Shah, M. (2020) A Comparative Analysis of Logistic Regression, Random Forest and KNN Models for the Text Classification. *Augmented Human Research*. 5(12). https://doi.org/10.1007/s41133-020-00032-0

Shellman, S. M. (2008). Coding Disaggregated Intrastate Conflict: Machine Processing the Behavior of Substate Actors Over Time and Space. *Political Analysis*, *16*(4), 464–477. http://www.jstor.org/stable/25791950

Steinkrau, D., Simard, P.Y., and Buck, I. (2005). Using GPUs for machine learning algorithms. *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*. 2:(1115-1120).

Sumathi, D., Alluri, K. (2021). Deploying Deep Learning Models for Various Real-Time Applications Using Keras. In: Prakash, K.B., Kannan, R., Alexander, S., Kanagachidambaresan, G.R. (eds) *Advanced Deep Learning for Engineers and Scientists*. EAI/Springer Innovations in Communication and Computing. Springer, Cham. https://doi.org/10.1007/978-3-030-66519-7_5

Tao, X. (2022). Classification of International Relations Based on Deep Learning. *IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers* (IPEC). 10.1109/IPEC54454.2022.9777594

Therneau T, Atkinson B (2022). _rpart: Recursive Partitioning and Regression Trees_. R package version 4.1.19, <https://CRAN.R-project.org/package=rpart>.

Westerwinter, O. (2022). Hierarchy, Differentiation, and Inter-institutional Collaboration in the Regime Complex for Global Private Security Governance. Working Paper. February 25, 2022

Weber M (1994 [1919]) Politik als Beruf. In: Mommsen WJ, Schluchter W, Morgenbrod B (eds) Max-Weber-Studienausgabe I/17, Wissenschaft als Beruf, Politik als Beruf. Tübingen: Mohr, 35–88.

Wu, H., Liu, Y., Wang, J. (2020). Review of Text Classification Methods on Deep Learning. *Computers, Materials & Continua*. 63(3): 1309-1321. https://doi.org/10.32604/cmc.2020.010172

Yamins, D. L., Hong, H., Cadieu, C., and DiCarlo, J. J. (2013). Hierarchical modular optimization of convolutional networks achieves representations similar to macaque IT and human ventral stream. *Advances in neural information processing systems*. 26.

Zulqarnain, M., Ghazali, R., Mohmad Hassim, Y. M., and Rehan, M. (2020). A Comparative Review on Deep Learning Models for Text Classification. *Indonesian Journal of Electrical Engineering and Computer Science.* 19(1): 325-335.

Appendix A

R code used in this research report

(Database available from author)

```
#libraries ----
library(tidyverse)
library(neuralnet)
library(readxl)
library(tensorflow)
library(keras)
library(tfdatasets)
library(tm)
library(caret)
library(readr)
library(dplyr)
library(randomForest)
library(rpart)
library(e1071)


# Functions ----
#pivot wide function
make_wide_data = function(d){
  # Takes the originally coded data frame and adds one column per deferred to
organisation
  orgs = d$def_org
  split_orgs = strsplit(orgs,", *")
  unique_orgs = remove_NAs(unique(unlist(split_orgs)))
  types = d$def_type
  split_types = strsplit(types,", *")
  unique_types = remove_NAs(unique(unlist(split_types)))
  for(org in unique_orgs){
    d[,org] <- NA
  }
  for(i in 1:nrow(d)){
    orgsi = split_orgs[[i]]
    typesi = split_types[[i]]
    if(length(orgsi) != length(typesi)){
      print(paste("incorrect lengths",i))
    } else if(!is.na(orgsi) && !is.na(typesi)){
      for(j in 1:length(orgsi)){
        d[i,orgsi[j]] = typesi[j]
      }
    }
  }
  d
}

remove_NAs = function(x){
  x = x[!is.na(x)]
  x
```

```
}

#sequences to occourence function
sequences_to_occurence <- function(seq_matrix){
  num_terms = max(seq_matrix)
  n = nrow(seq_matrix)
  oc_matrix = matrix(0,n,num_terms)
  for(i in 1:n){
    termsi = seq_matrix[i,]
    termsi = termsi[termsi!=0]
    counts = table(termsi) # count how many times each term occurs
    oc_matrix[i, as.numeric(names(counts))] = counts
  }
  oc_matrix
}

#get data
path = "C:/Users/rosam/OneDrive/Documents/"
filename = "Deference_data_private_security_governance_12072023_RM.xlsx"
fulldata <- read_excel(paste0(path,filename), sheet = "Deference data")
#make wide using that function we made
widefulldata <- (make_wide_data(fulldata))

#getting data into the right format, removing unnecessary columns and blank
rows
defdata <- widefulldata[which(widefulldata$def_org!="NA"),]
defdata <- defdata[ ,-c(10, 5, 9) ]
defUN_multi_data <- defdata[c(4,8)]
summary(defUN_multi_data)
defUN_multi_data$UN <- as.numeric(defUN_multi_data$UN)
table(defUN_multi_data$UN)

#recode NA's to zeros, 2's and 3's to 2s and 5s to 4s to 3s
defUN_multi_data$UN[is.na(defUN_multi_data$UN)] <- 0
defUN_multi_data$UN[defUN_multi_data$UN ==3] <- 2
defUN_multi_data$UN[defUN_multi_data$UN >3] <- 3
summary(defUN_multi_data)
table(defUN_multi_data$UN)

data <- defUN_multi_data
names(data) <- c("text", "labels")
summary(data)


#Preprocess data ----

#convert labels to one-hot-encoding
one_hot_labels <- to_categorical(data$labels, num_classes = 4)  # 4 is the
number of classes

#text preprocessing
tokenizer <- text_tokenizer(num_words = 5000) #decreased the no. of words to
5000 and accuracy improves
tokenizer$fit_on_texts(data$text)
sequences <- texts_to_sequences(tokenizer, data$text)
word_index <- tokenizer$word_index
```

```
max_seq_length <- max(sapply(sequences, length))
padded_sequences <- pad_sequences(sequences, maxlen = max_seq_length)

#print these out and add to document:
# Get word counts
word_counts <- tokenizer$word_counts
# Convert word_counts to a data frame
word_counts_df <- data.frame(word = names(word_counts), count =
as.integer(word_counts), stringsAsFactors = FALSE)
# Sort the data frame by count in descending order
sorted_word_counts <- word_counts_df[order(-word_counts_df$count), ]
# Print the top 25 words and their frequencies
top_words <- head(sorted_word_counts, 40)
head(tokenizer$index_word,15)

print(top_words, row.names = FALSE)
head(word_index, 25)
max_seq_length
min(sapply(sequences, length))
length(padded_sequences)



# Split the data into training and validation sets
set.seed(123)
train_indices <- sample(1:nrow(data), 0.8 * nrow(data))
train_data <- padded_sequences[train_indices, ]
train_labels <- one_hot_labels[train_indices, ]
val_data <- padded_sequences[-train_indices, ]
val_labels <- one_hot_labels[-train_indices, ]



# Run Keras Sequential models ----

#MODEL 1
#lstm model 32 units
# Create a Sequential model
model <- keras_model_sequential(name = "model1")

# Add an Embedding layer
model %>%
  layer_embedding(input_dim = length(word_index) + 1, output_dim = 200,
input_length = max_seq_length)

# Add an LSTM layer
model %>%
  layer_lstm(units = 32)

# Add a Dense layer for classification (adjust units for the number of
classes)
model %>%
  layer_dense(units = 4, activation = "softmax")  # 4 is the number of classes

summary(model)
```

```
# Compile the model
model %>% compile(
  loss = "categorical_crossentropy",
  optimizer = optimizer_adam(),
  metrics = c("accuracy")
)


# Train the model
Sys.time()
a= Sys.time()
history <- model %>% fit(
  x = train_data,
  y = train_labels,
  epochs = 6,
  batch_size = 32,
  validation_data = list(val_data, val_labels)
)
b= Sys.time()
b-a
history$metrics
max(history$metrics$val_accuracy)
summary(model)



##Model 2
#LSTM model 64 units, more epochs

# Create a Sequential model
model <- keras_model_sequential(name = "model2")

# Add an Embedding layer
model %>%
  layer_embedding(input_dim = length(word_index) + 1, output_dim = 200,
input_length = max_seq_length)

# Add an LSTM layer with more units
model %>%
  layer_lstm(units = 64)  # Increase the number of units

# Add a Dense layer for classification
model %>%
  layer_dense(units = 4, activation = "softmax")  # 4 is the number of classes

# Compile the model
model %>% compile(
  loss = "categorical_crossentropy",
  optimizer = optimizer_adam(),
  metrics = c("accuracy")
)

# Train the model with more epochs
Sys.time()
a= Sys.time()
history <- model %>% fit(
```

```
  x = train_data, y = train_labels,
  epochs = 20,
  batch_size = 32,
  validation_data = list(val_data, val_labels)
)
b= Sys.time()
b-a

history$metrics
summary(model)
max(history$metrics$val_accuracy)



##Model 3
#LSTM model 64 units. custom learning rate

# Create a Sequential model
model <- keras_model_sequential(name = "model3")

# Add an Embedding layer
model %>%
  layer_embedding(input_dim = length(word_index) + 1, output_dim = 200,
input_length = max_seq_length)

# Add an LSTM layer with more units
model %>%
  layer_lstm(units = 64)  # Increase the number of units

# Add a Dense layer for classification
model %>%
  layer_dense(units = 4, activation = "softmax")

# Compile the model with a custom learning rate
model %>% compile(
  loss = "categorical_crossentropy",
  optimizer = optimizer_adam(learning_rate = 0.001),  # Adjust the learning
rate
  metrics = c("accuracy")
)

# Train the model with less epochs
Sys.time()
a= Sys.time()

history <- model %>% fit(
  x = train_data, y = train_labels,
  epochs = 11,  # 11 epochs
  batch_size = 32,
  validation_data = list(val_data, val_labels)
)
b= Sys.time()
b-a

history$metrics
summary(model)
```

```
max(history$metrics$val_accuracy)



## Model4 LSTM with 96 units

# Create a Sequential model
model <- keras_model_sequential(name = "model4")

# Add an Embedding layer
model %>%
  layer_embedding(input_dim = length(word_index) + 1, output_dim = 200,
input_length = max_seq_length)

# Add an LSTM layer with more units
model %>%
  layer_lstm(units = 96)  # Increase the number of units

# Add a Dense layer for classification
model %>%
  layer_dense(units = 4, activation = "softmax")

# Compile the model with a custom learning rate
model %>% compile(
  loss = "categorical_crossentropy",
  optimizer = optimizer_adam(learning_rate = 0.001),
  metrics = c("accuracy")
)


# Train the model
Sys.time()
a= Sys.time()

history <- model %>% fit(
  x = train_data, y = train_labels,
  epochs = 11,  # 11 epochs
  batch_size = 32,
  validation_data = list(val_data, val_labels)
)
b= Sys.time()
b-a

history$metrics
summary(model)
max(history$metrics$val_accuracy)


### Model 5 LSTM 64 Units adjusted learning rate

# Create a Sequential model
model <- keras_model_sequential(name = "model5")

# Add an Embedding layer
model %>%
```

```
  layer_embedding(input_dim = length(word_index) + 1, output_dim = 200,
input_length = max_seq_length)

# Add an LSTM layer with 64 units
model %>%
  layer_lstm(units = 64)  # decrease back to 64 units

# Add a Dense layer for classification
model %>%
  layer_dense(units = 4, activation = "softmax")

# Compile the model with a custom learning rate
model %>% compile(
  loss = "categorical_crossentropy",
  optimizer = optimizer_adam(learning_rate = 0.0001),
  metrics = c("accuracy")
)

# Train the model, increase the epochs again
Sys.time()
a= Sys.time()

history <- model %>% fit(
  x = train_data, y = train_labels,
  epochs = 15,  # 15 epochs
  batch_size = 32,
  validation_data = list(val_data, val_labels)
)
b= Sys.time()
b-a

history$metrics
summary(model)
max(history$metrics$val_accuracy)


##Model 6 first Conv Net (filter 128, kernal 5)

# Create a Sequential model
model <- keras_model_sequential(name = "model6")

# Add an Embedding layer
model %>%
  layer_embedding(input_dim = length(word_index) + 1, output_dim = 200,
input_length = max_seq_length)

# Add a 1D Convolutional layer
model %>%
  layer_conv_1d(filters = 128, kernel_size = 5, activation = "relu")

# Add a Global Max Pooling layer
model %>%
  layer_global_max_pooling_1d()

# Add a Dense layer for classification
model %>%
```

```
  layer_dense(units = 4, activation = "softmax")

# Compile the model with the Adam optimizer and a custom learning rate
model %>% compile(
  loss = "categorical_crossentropy",
  optimizer = optimizer_adam(learning_rate = 0.001),  # Adjust the learning
rate
  metrics = c("accuracy")
)



# Train the model
Sys.time()
a= Sys.time()

history <- model %>% fit(
  x = train_data, y = train_labels,
  epochs = 10,  # 10 epochs
  batch_size = 32,
  validation_data = list(val_data, val_labels)
)
b= Sys.time()
b-a

history$metrics
summary(model)
max(history$metrics$val_accuracy)



##model 7 CNN with adjustments on filter
# Create a Sequential model
model <- keras_model_sequential(name = "model7")

# Add an Embedding layer
model %>%
  layer_embedding(input_dim = length(word_index) + 1, output_dim = 200,
input_length = max_seq_length)

# Add a 1D Convolutional layer
model %>%
  layer_conv_1d(filters = 64, kernel_size = 5, activation = "relu")  # Adjust
filters and kernel_size

# Add a Global Max Pooling layer
model %>%
  layer_global_max_pooling_1d()

# Add a Dense layer for classification
model %>%
  layer_dense(units = 4, activation = "softmax")

# Compile the model with the Adam optimizer and a custom learning rate
model %>% compile(
  loss = "categorical_crossentropy",
```

```
  optimizer = optimizer_adam(learning_rate = 0.001),
  metrics = c("accuracy")
)


# Train the model
Sys.time()
a= Sys.time()

history <- model %>% fit(
  x = train_data, y = train_labels,
  epochs = 10,  # 10 epochs
  batch_size = 32,
  validation_data = list(val_data, val_labels)
)
b= Sys.time()
a-b

history$metrics
summary(model)
max(history$metrics$val_accuracy)


##Model 8 CNN, kernal size 3
# Create a Sequential model
model <- keras_model_sequential(name = "model8")

# Add an Embedding layer
model %>%
  layer_embedding(input_dim = length(word_index) + 1, output_dim = 200,
input_length = max_seq_length)

# Add a 1D Convolutional layer
model %>%
  layer_conv_1d(filters = 128, kernel_size = 3, activation = "relu")  # Adjust
filters and kernel_size

# Add a Global Max Pooling layer
model %>%
  layer_global_max_pooling_1d()

# Add a Dense layer for classification
model %>%
  layer_dense(units = 4, activation = "softmax")

# Compile the model
model %>% compile(
  loss = "categorical_crossentropy",
  optimizer = optimizer_adam(learning_rate = 0.001),
  metrics = c("accuracy")
)


# Train the model
Sys.time()
a= Sys.time()
```

```
history <- model %>% fit(
  x = train_data, y = train_labels,
  epochs = 10,  # 10 epochs
  batch_size = 32,
  validation_data = list(val_data, val_labels)
)
b= Sys.time()
a-b

history$metrics
summary(model)
max(history$metrics$val_accuracy)



##Model 9 CNN, kernal size 7
# Create a Sequential model
model <- keras_model_sequential(name = "model9")

# Add an Embedding layer
model %>%
  layer_embedding(input_dim = length(word_index) + 1, output_dim = 200,
input_length = max_seq_length)

# Add a 1D Convolutional layer
model %>%
  layer_conv_1d(filters = 128, kernel_size = 7, activation = "relu")  # Adjust
filters and kernel_size

# Add a Global Max Pooling layer
model %>%
  layer_global_max_pooling_1d()

# Add a Dense layer for classification
model %>%
  layer_dense(units = 4, activation = "softmax")  # 4 is the number of classes

# Compile the model with the Adam optimizer and a custom learning rate
model %>% compile(
  loss = "categorical_crossentropy",
  optimizer = optimizer_adam(learning_rate = 0.001),  # Adjust the learning
rate
  metrics = c("accuracy")
)


# Train the model
Sys.time()
a= Sys.time()

history <- model %>% fit(
  x = train_data, y = train_labels,
  epochs = 10,  # 10 epochs
  batch_size = 32,
  validation_data = list(val_data, val_labels)
```

```
)
b= Sys.time()
a-b

history$metrics
summary(model)
max(history$metrics$val_accuracy)



## Model 10 CNN further increase filter size

# Create a Sequential model
model <- keras_model_sequential(name = "model10")

# Add an Embedding layer
model %>%
  layer_embedding(input_dim = length(word_index) + 1, output_dim = 200,
input_length = max_seq_length)

# Add a 1D Convolutional layer
model %>%
  layer_conv_1d(filters = 256, kernel_size = 5, activation = "relu")  # Adjust
filters to 256

# Add a Global Max Pooling layer
model %>%
  layer_global_max_pooling_1d()

# Add a Dense layer for classification
model %>%
  layer_dense(units = 4, activation = "softmax")  # 4 is the number of classes

# Compile the model with the Adam optimizer and a custom learning rate
model %>% compile(
  loss = "categorical_crossentropy",
  optimizer = optimizer_adam(learning_rate = 0.001),
  metrics = c("accuracy")
)


# Train the model
Sys.time()
a= Sys.time()

history <- model %>% fit(
  x = train_data, y = train_labels,
  epochs = 10,  # 10 epochs
  batch_size = 32,
  validation_data = list(val_data, val_labels)
)
b= Sys.time()
a-b

history$metrics
summary(model)
```

```
max(history$metrics$val_accuracy)


## Model 11 CNN plus LSTM

# Create a Sequential model
model <- keras_model_sequential(name = "model11")

# Add an Embedding layer
model %>%
  layer_embedding(input_dim = length(word_index) + 1, output_dim = 200,
input_length = max_seq_length)

# Add a 1D Convolutional layer
model %>%
  layer_conv_1d(filters = 128, kernel_size = 5, activation = "relu")

# Add an LSTM layer
model %>%
layer_lstm(units = 64, return_sequences = TRUE)

# Add a Global Max Pooling layer
model %>%
  layer_global_max_pooling_1d()

# Add a Dense layer for classification
model %>%
  layer_dense(units = 4, activation = "softmax")  # 4 is the number of classes

# Compile the model =
model %>% compile(
  loss = "categorical_crossentropy",
  optimizer = optimizer_adam(learning_rate = 0.001),
  metrics = c("accuracy")
)


# Train the model
Sys.time()
a= Sys.time()

history <- model %>% fit(
  x = train_data, y = train_labels,
  epochs = 15,  # 15 epochs
  batch_size = 32,
  validation_data = list(val_data, val_labels)
)
b= Sys.time()
b-a


history$metrics
summary(model)
max(history$metrics$val_accuracy)
```

```
##model 12 CNN with dropout (0.4)

# Create a Sequential model
model <- keras_model_sequential(name = "model12")

# Add an Embedding layer
model %>%
  layer_embedding(input_dim = length(word_index) + 1, output_dim = 200,
input_length = max_seq_length)

# Add a 1D Convolutional layer with Dropout
model %>%
  layer_conv_1d(filters = 128, kernel_size = 5, activation = "relu") %>%
  layer_dropout(rate = 0.4)  # Adjust the dropout rate as needed

# Add a Global Max Pooling layer with Dropout
model %>%
  layer_dropout(rate = 0.4) %>%
  layer_global_max_pooling_1d()

# Add a Dense layer with Dropout for classification
model %>%
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 4, activation = "softmax")

# Compile the model with the Adam optimizer and a custom learning rate
model %>% compile(
  loss = "categorical_crossentropy",
  optimizer = optimizer_adam(learning_rate = 0.001),
  metrics = c("accuracy")
)


# Train the model
Sys.time()
a= Sys.time()

history <- model %>% fit(
  x = train_data, y = train_labels,
  epochs = 15,
  batch_size = 32,
  validation_data = list(val_data, val_labels)
)
b= Sys.time()
a-b

history$metrics
summary(model)
max(history$metrics$val_accuracy)


#model 13 CNN with adjusted dropout (0.3)

# Create a Sequential model
model <- keras_model_sequential(name = "model13")
```

```
# Add an Embedding layer
model %>%
  layer_embedding(input_dim = length(word_index) + 1, output_dim = 200,
input_length = max_seq_length)

# Add a 1D Convolutional layer with Dropout
model %>%
  layer_conv_1d(filters = 128, kernel_size = 5, activation = "relu") %>%
  layer_dropout(rate = 0.3)  # Adjust the dropout rate

# Add a Global Max Pooling layer with Dropout
model %>%
  layer_dropout(rate = 0.3) %>%
  layer_global_max_pooling_1d()

# Add a Dense layer with Dropout for classification
model %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 4, activation = "softmax")  # 4 is the number of classes

# Compile the model
model %>% compile(
  loss = "categorical_crossentropy",
  optimizer = optimizer_adam(learning_rate = 0.001),
  metrics = c("accuracy")
)


# Train the model
Sys.time()
a= Sys.time()

history <- model %>% fit(
  x = train_data, y = train_labels,
  epochs = 15,  # 15 epochs
  batch_size = 32,
  validation_data = list(val_data, val_labels)
)
b= Sys.time()
a-b

history$metrics
summary(model)
max(history$metrics$val_accuracy)


##Model 14 - more hidden layers

# Create a Sequential model
model <- keras_model_sequential(name = "model14")

# Add an Embedding layer
model %>%
  layer_embedding(input_dim = length(word_index) + 1, output_dim = 200,
input_length = max_seq_length)
```

```
# Add a 1D Convolutional layer with Dropout
model %>%
  layer_conv_1d(filters = 128, kernel_size = 5, activation = "relu") %>%
  layer_dropout(rate = 0.4)  # Adjust the dropout rate as needed

# Add another 1D Convolutional layer with Dropout
model %>%
  layer_conv_1d(filters = 64, kernel_size = 3, activation = "relu") %>%
  layer_dropout(rate = 0.4)

# Add another 1D Convolutional layer with Dropout
model %>%
  layer_conv_1d(filters = 32, kernel_size = 3, activation = "relu") %>%
  layer_dropout(rate = 0.4)

# Add a Global Max Pooling layer with Dropout
model %>%
  layer_dropout(rate = 0.4) %>%
  layer_global_max_pooling_1d()

# Add a Dense layer with Dropout
model %>%
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 64, activation = "relu")

# Add the final Dense layer for classification
model %>%
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 4, activation = "softmax")

# Compile the model
model %>% compile(
  loss = "categorical_crossentropy",
  optimizer = optimizer_adam(learning_rate = 0.001),
  metrics = c("accuracy")
)

# Train the model
Sys.time()
a= Sys.time()

history <- model %>% fit(
  x = train_data, y = train_labels,
  epochs = 15,  # 15 epochs
  batch_size = 32,
  validation_data = list(val_data, val_labels)
)
b= Sys.time()
a-b

history$metrics
summary(model)
max(history$metrics$val_accuracy)


# Traditional Models ----
```

```
#convert the prepossessed data into matrix
sequences_to_occurence <- function(seq_matrix){
  num_terms = max(seq_matrix)
  n = nrow(seq_matrix)
  oc_matrix = matrix(0,n,num_terms)
  for(i in 1:n){
    termsi = seq_matrix[i,]
    termsi = termsi[termsi!=0]
    counts = table(termsi) # count how many times each term occurs
    oc_matrix[i, as.numeric(names(counts))] = counts
  }
  oc_matrix
}


#Turn sequences into occurrence matrix
occ_matrix = sequences_to_occurence(padded_sequences)

# Train-test split
set.seed(123)
train_indices_rf <- sample(1:nrow(data), 0.8 * nrow(data))
train_data_rf <- occ_matrix[train_indices_rf, ]
train_labels_rf <- factor(data$labels[train_indices_rf])
val_data_rf <- occ_matrix[-train_indices_rf, ]
val_labels_rf <- factor(data$labels[-train_indices_rf])


# model 15 RANDOM FOREST ----
# Train the Random Forest model
rf_model <- randomForest(train_data_rf, train_labels_rf, type =
"classification")

# Predict using the Random Forest model
rf_predictions <- predict(rf_model, val_data_rf)

# Calculate accuracy
rf_accuracy <- sum(rf_predictions == val_labels_rf) / length(val_labels_rf)
print(paste("Random Forest Accuracy:", rf_accuracy))

summary(rf_model)
confusionMatrix(rf_predictions, val_labels_rf)
varImpPlot(rf_model, n.var = 10, main= "Random Forest Variable Importance
Plot")



# Decision Trees ----

# use the Reshaped sequences from the random forest pre-processing)

# Train the Decision Tree model
tree_model <- rpart(train_labels_rf ~ ., data = as.data.frame(train_data_rf),
method = "class")

# Predict using the Decision Tree model
```

```
tree_predictions <- predict(tree_model, as.data.frame(val_data_rf), type =
"class")

# Calculate accuracy
tree_accuracy <- sum(tree_predictions == val_labels_rf) /
length(val_labels_rf)
print(paste("Decision Tree Accuracy:", tree_accuracy))

summary(tree_model)
tree_model

confusionMatrix(tree_predictions, factor(val_labels_rf))


#Naive Bayes model----


# Naive Bayes model
nb_model <- naiveBayes(train_data_rf, train_labels_rf)

# Make predictions on the validation set
nb_predictions <- predict(nb_model, val_data_rf)


# Calculate accuracy
nb_accuracy <- sum(nb_predictions == val_labels_rf) / length(val_labels_rf)
print(paste("Naive Bayes Accuracy:", nb_accuracy))

# Confusion Matrix
confusionMatrix(nb_predictions, val_labels_rf)



## print confusion matrix for keras model
# Make predictions on the validation data
predictions <- model %>% predict(val_data) %>% k_argmax()

cm <- table(as.vector(predictions),(data$labels[-train_indices_rf])) #print
confusion matrix
print("Confusion Matrix:")
print(cm)
```