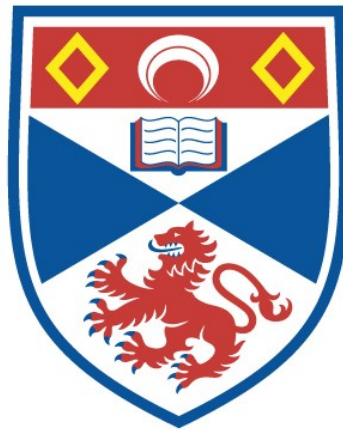


# Processing clinical guideline text for formal verification

Fahrurrozi Rahman

A thesis submitted for the degree of PhD  
at the  
University of St Andrews



2022

Full metadata for this thesis is available in  
St Andrews Research Repository  
at:

<https://research-repository.st-andrews.ac.uk/>

Identifier to use to cite or link to this thesis:

DOI: <https://doi.org/10.17630/sta/935>

This item is protected by original copyright

### **Candidate's declaration**

I, Fahrurrozi Rahman, do hereby certify that this thesis, submitted for the degree of PhD, which is approximately 29,795 words in length, has been written by me, and that it is the record of work carried out by me, or principally by myself in collaboration with others as acknowledged, and that it has not been submitted in any previous application for any degree. I confirm that any appendices included in my thesis contain only material permitted by the 'Assessment of Postgraduate Research Students' policy.

I was admitted as a research student at the University of St Andrews in September 2016.

I received funding from an organisation or institution and have acknowledged the funder(s) in the full text of my thesis.

Date 14 February 2022

Signature of candidate

### **Supervisor's declaration**

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of PhD in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree. I confirm that any appendices included in the thesis contain only material permitted by the 'Assessment of Postgraduate Research Students' policy.

Date 14 February 2022

Signature of supervisor

### **Permission for publication**

In submitting this thesis to the University of St Andrews we understand that we are giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. We also understand, unless exempt by an award of an embargo as requested below, that the title and the abstract will be published, and that a copy of the work may be made and supplied to any bona fide library or research worker, that this thesis will be electronically accessible for personal or research use and that the library has the right to migrate this thesis into new electronic forms as required to ensure continued access to the thesis.

I, Fahrurrozi Rahman, confirm that my thesis does not contain any third-party material that requires copyright clearance.

The following is an agreed request by candidate and supervisor regarding the publication of this thesis:

**Printed copy**

No embargo on print copy.

**Electronic copy**

No embargo on electronic copy.

Date 14 February 2022

Signature of candidate

Date 14 February 2022

Signature of supervisor

## **Underpinning Research Data or Digital Outputs**

### **Candidate's declaration**

I, Fahrurrozi Rahman, hereby certify that no requirements to deposit original research data or digital outputs apply to this thesis and that, where appropriate, secondary data used have been referenced in the full text of my thesis.

Date 14 February 2022

Signature of candidate

# Acknowledgements

I would express my gratitude to my supervisor Dr Juliana Bowles, for her constant guidance, support and patience throughout my PhD. This PhD has helped me growing up, from limited to no knowledge in formal methods and machine learning when I started, to gaining academic skills and becoming an independent researcher. Thank you also for always encouraging me to break out of my bubble.

I would also like to thank Dr Ruth Hoffmann and Dr Beatrice Alex for their patience and guidance during the viva. Their invaluable feedback for this thesis is very much appreciated.

Many thanks also to the academic and administrative staffs at the School of Computer Science, University of St Andrews. My journey could have been rougher without the help of Stuart, Alex, Julie, and Sylvia.

Thanks also to people that I have only met on the Internet throughout online forums and have helped me moving forward with their invisible hands.

Special thanks to Clara, Juanjo, Andy, Ibrahim, Remmy and Deborah for being faithful friends through thick and thin; Trish and Jim for constantly checking on my well being and providing me with gifts especially throughout the lockdown; Marco for the technical help rasterising a pdf file; Emma for our chats that keep me sane; and Jonny, a good brother I never had from Utah.

And lastly, to me, for surviving after all the ups and downs.

**Funding** This work was supported by the Indonesian Endowment Fund for Education (LPDP) 2016-2020.

# Abstract

Clinical guidelines are evidence-based recommendations developed to assist practitioners in their decisions on appropriate care for patients with specific clinical circumstances. They provide succinct instructions such as what drugs should be given or taken for a particular condition, how long such treatment should be given, what tests should be conducted, or other situational clinical circumstances for certain diseases. However, as they are described in natural language, they are prone to problems such as ambiguity and incompleteness. As the guidelines are publicly accessible, we expect them to be foolproof from inconsistencies and missing gaps. This thesis aims to answer a couple questions in regard to the correctness of clinical guidelines: (1) How can we get the main information in clinical guideline texts? (2) How can we check the guidelines in terms of correctness and consistencies? To answer these questions, first, we develop several methods to mark and capture the semantic information in the texts. We start by building a controlled natural language to reduce the complexity of the texts' structure. We show that this approach is easy to set up but obviously unscalable. We then consider machine learning approaches and use semantic role labelling, named-entity recognition and relation classification techniques. To achieve this task, we create a clinical guideline corpus tagged with process labels. We show that even with a small corpus, the baseline performance is promising. We then investigate fine-tuning some state-of-the-art neural model architectures and get better performance. Finally, we create a framework to transform the clinical guidelines into formal statements and check their correctness against some properties using model checkers or constraint solvers. This thesis presents a study and analyses of entity labelling and relation classification in regard to clinical guidelines, as well as formally

checking their correctness, providing insights and future research directions on the improvement of clinical guidelines.



# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Objectives . . . . .	3
1.2	Outline . . . . .	4
<b>2</b>	<b>Related Work</b>	<b>6</b>
2.1	Modelling Software Requirements . . . . .	6
2.1.1	Automatic Test Cases Generation from Requirements . . . . .	9
2.1.1.1	Syntactic analysis . . . . .	10
2.1.1.2	Semantic analysis . . . . .	11
2.1.2	Semantic Annotation in Software Requirements . . . . .	12
2.1.2.1	Ontology of concepts . . . . .	14
2.1.2.2	Syntactic analysis . . . . .	15
2.2	Natural Language Processing in Healthcare Data . . . . .	16
2.2.1	Formal Checking for Clinical Guidelines . . . . .	17
2.2.2	Semantic Annotation in Biomedical Texts . . . . .	18
2.3	Summary . . . . .	22
<b>3</b>	<b>Controlled Natural Language for Clinical Guidelines</b>	<b>23</b>
3.1	CNL in Clinical Guidelines . . . . .	23
3.1.1	Syntactic Analysis . . . . .	26
3.1.2	Semantic Analysis . . . . .	32
3.1.2.1	Parsing sentences in CNL . . . . .	32
3.1.2.2	Case frame generation . . . . .	33
3.2	Summary . . . . .	37
<b>4</b>	<b>Machine Learning for Clinical Guidelines</b>	<b>39</b>
4.1	Semantic Role Labelling . . . . .	39
4.1.1	Semantics in clinical guidelines . . . . .	41

4.1.2	Relationship between concepts . . . . .	42
4.1.3	Syntactic analysis of guideline sentences . . . . .	43
4.1.4	Semantic analysis of guideline sentences . . . . .	45
4.1.5	Features . . . . .	46
4.1.6	Word embedding . . . . .	47
4.1.7	Learning . . . . .	50
4.2	NER & Relation Classification . . . . .	51
4.2.1	NER in biomedical texts . . . . .	52
4.2.2	Baseline model for NER . . . . .	53
4.2.3	Baseline model for relation classification . . . . .	55
4.2.4	Fine-tuning state-of-the-art models . . . . .	58
4.2.4.1	Fine-tuning spaCy model for NER . . . . .	58
4.2.4.2	Fine-tuning BERT model . . . . .	59
4.3	Summary . . . . .	63
<b>5</b>	<b>Experiments and Evaluation</b>	<b>64</b>
5.1	Dataset Analysis . . . . .	64
5.2	Experiments . . . . .	69
5.2.1	Experiments on Semantic Role Labelling (SRL) . . . . .	69
5.2.2	Experiments on Named-Entity Recognition (NER) . . . . .	70
5.2.3	Fine-tuning spaCy and BERT models for NER . . . . .	71
5.2.4	Experiments on relation classification . . . . .	76
5.2.5	Fine-tuning BERT models for relation classification . . . . .	77
5.3	Improving the Models . . . . .	81
5.4	Summary . . . . .	81
<b>6</b>	<b>Formal Model Generation and Verification</b>	<b>83</b>
6.1	Logical Form . . . . .	84
6.1.1	Propositional Logic . . . . .	85
6.1.2	First-order Logic . . . . .	85
6.1.3	Temporal Logic and Linear Temporal Logic . . . . .	86
6.1.4	Computation Tree Logic . . . . .	88
6.2	Constraint Solver and Model Checker . . . . .	89
6.2.1	Transforming and verifying guidelines in UPPAAL . . . . .	90
6.2.1.1	Guideline transformation . . . . .	91
6.2.1.2	Verification . . . . .	97
6.2.2	Transforming and verifying guidelines in PRISM . . . . .	98
6.2.2.1	Guidelines transformation . . . . .	99

6.2.2.2	Verification . . . . .	103
6.2.3	Transforming and verifying guidelines in Z3 . . . . .	104
6.2.3.1	Guideline transformation . . . . .	106
6.2.3.2	Verification . . . . .	109
6.3	Summary . . . . .	111
<b>7</b>	<b>Conclusions</b>	<b>112</b>
7.1	Key Contributions and Findings . . . . .	114
7.2	Future Work . . . . .	115
	<b>Bibliography</b>	<b>118</b>
	<b>Index of Terms</b>	<b>133</b>
<b>A</b>	<b>Penn and Brown Corpora Tagset</b>	<b>135</b>
A.1	Penn Corpus Tagset . . . . .	135
A.2	Brown Corpus Tagset . . . . .	136
<b>B</b>	<b>Annotation Guidelines</b>	<b>139</b>
B.1	Entity Annotation Guide . . . . .	139
B.2	Relation Annotation Guide . . . . .	140

# List of Figures

1.1	The framework for verifying clinical guidelines . . . . .	4
2.1	The left image shows the classical waterfall process in software engineering whereas the right image illustrates the requirement engineering process consisting of requirement elicitation and analysis, specification and validation (Sommerville, 2016). . . . .	7
2.2	Phases of the NAT2TEST strategy (Carvalho, 2016) . . . . .	9
2.3	An example of a requirement frame (Carvalho, 2016) . . . . .	13
2.4	The architecture of S-CASE (Diamantopoulos et al., 2017) . . . . .	13
2.5	Ontology of Concepts (figure readjusted from Diamantopoulos et al., 2017) . . . . .	14
2.6	Syntactic analysis in the Mate Tools pipeline . . . . .	15
2.7	Part of the statechart defined from the Intravenous Catheters (IRC) guideline (Pérez and Porres, 2010) . . . . .	18
3.1	The National Institute for Health and Care Excellence Type 2 Diabetes (T2D) Therapy Algorithm . . . . .	25
3.2	A fragment of the parse tree for the first sentence in the T2D therapy algorithm: " <i>If HbA1c rises to 48mmol/mol(6.5%) on lifestyle interventions: offer standard-release metformin</i> ". . . . .	36
4.1	Semantic role labelling concepts that we use in this study. . . . .	42
4.2	Dependency parse tree for the sentence: " <i>All patients should have appropriate monitoring for clinically significant AEs.</i> " . . . . .	45
4.3	Word embedding capturing the semantic similarity of words . . . . .	48
4.4	Two sentences marked with Stanza i2b2-2010 tags . . . . .	53
4.5	A sentence marked with the BIO format . . . . .	54
4.6	Named-entity recognition model implemented using BiLSTM . . . . .	55
4.7	Relation classification model implemented using BiLSTM . . . . .	56

4.8	A sentence marked with the BIO and the BILUO formats . . . . .	59
4.9	The Transformer architecture (Vaswani et al., 2017) . . . . .	60
5.1	Sentence distribution over token counts. X axis denotes the interval of token size in a sentence. Y axis denotes the number of sentences having a particular token size. . . . .	66
5.2	Heatmaps of the difference between the baseline, BERT and BIO_-ClinicalBERT NER confusion matrix based on the strictness of entity classification. The X-axis shows the evaluation strategy, and the Y-axis shows the named-entities. The numbers are showing F1-scores (in %) . . . . .	75
5.3	Heatmaps of the difference between the baseline, BERT and BIO_-ClinicalBERT relation classification confusion matrix. The numbers are showing F1-scores (in %) . . . . .	79
6.1	Pipeline for verifying a clinical guideline . . . . .	84
6.2	An LTL path for $\Box\Diamond a \rightarrow \Box\Diamond b$ formula . . . . .	87
6.3	A CTL tree for $\forall\Box\forall\Diamond a \rightarrow \forall\Box\forall\Diamond b$ formula . . . . .	89
6.4	A text snippet from the T2D therapy algorithm . . . . .	92
6.5	Generated model for adults with T2D that tolerate metformin . . . . .	94
6.6	Model for adults with T2D that tolerate metformin modified with branching points. . . . .	96
6.7	Percentage of drugs combination for double and triple therapy (Greiver et al., 2021) . . . . .	99
6.8	A simple system equation . . . . .	105

# List of Tables

2.1	Thematic roles for action and condition statements (Carvalho, 2016) .	12
2.2	Some relationships between concepts in S-CASE. . . . .	15
2.3	Transition-based Medical Recommendations for detecting Interactions (TMR4I) Concepts Summary (Carretta Zamborlini et al., 2016). . . .	21
3.1	Some set of commonly used thematic roles . . . . .	33
3.2	Case frames for a sentence in T2D therapy algorithm . . . . .	37
4.1	A set of widely recognised Semantic Roles (Palmer et al., 2010). The words in <b>cyan</b> fulfilling the role. . . . .	41
4.2	Relationship between concepts that we use in this study. . . . .	43
4.3	Feature sets and their usage for SRL task. The ✓ denotes the features used in a particular task while the ✗ otherwise. . . . .	50
4.4	Some Stanza and Flair biomedical NER corpora . . . . .	52
4.5	Relationships and their example. To declutter the table and for an aesthetic reason, we replace the $\langle e1 \rangle \langle /e1 \rangle$ and $\langle e2 \rangle \langle /e2 \rangle$ notations to mark the first and the second entities by highlighting them in <b>yellow</b> and <b>green</b> respectively. . . . .	57
4.6	Input example for relation classification model . . . . .	58
4.7	BERT models for fine-tuning (including the original Transformer). L is the number of layers, E is the size of the input word embeddings, H is the dimension of the hidden unit size, and A denotes the number of attention heads. . . . .	62
5.1	Data source for our corpus. . . . .	65
5.2	Inter-Annotator Agreement (IAA) for the concept and relationship instances (in %). . . . .	67
5.3	Concepts and relationships in clinical guideline . . . . .	67

5.4	Counts of named-entity instances in the original, BIO, and BILUO format. . . . .	68
5.5	Counts of relationship instances . . . . .	69
5.6	Semantic role labelling evaluation scores for several classifiers (in %). . . . .	70
5.7	Named-entity recognition evaluation scores for several word embeddings and input features (in %) . . . . .	70
5.8	Evaluation F1-score values for NER model (in %) . . . . .	71
5.9	Instance score for NER (in %) . . . . .	72
5.10	Cases for measuring a NER model. GS stands for "gold standard" whereas MP stands for "model prediction". For example, in the case number 2, the actual labels are "O O O O" but the second and the third labels are wrongly predicted as "B-object" and "I-object" by the model. . . . .	73
5.11	Evaluation of F1-score (in %) on relation classification for several GloVe word embeddings with varying dimension size. . . . .	76
5.12	Evaluation F1-score values for NER model (in %) . . . . .	77
5.13	Evaluation score for relation classification (in %) . . . . .	78
6.1	Propositional logic truth table . . . . .	85
6.2	Temporal operators in LTL. $\psi$ denotes a formula. . . . .	87
6.3	UPPAAL model verification result. $\models$ and $\not\models$ stand for <i>satisfied</i> and <i>not satisfied</i> respectively. . . . .	98
6.4	PRISM model verification result . . . . .	104
A.1	Penn Corpus Tagset. . . . .	135
A.2	Brown Corpus Tagset. . . . .	136

# Glossary

**ACR** Albumin Creatinine Ratio.

**ACT** *action*; the action performed if the conditions are satisfied.

**ACTL** Action Computational Tree Logic.

**AGT** *agent*; the entity who performs the action in a sentence.

**ANTLR** Another Tool for Language Recognition.

**APG** ABNF Parser Generator.

**ARD** Annals of the Rheumatic Disease.

**ATC** Air Traffic Control.

**BERT** Bidirectional Encoder Representations from Transformers.

**CAC** *condition action*; the action for enabling a condition.

**CACT** *nested condition action*; the nested condition in the condition clause.

**CDS** Clinical Decision Support.

**CFG** Context-Free Grammar.

**CFV** *condition from value*; the value of CPT before performing CAC.

**CKD** Chronic Kidney Disease.

**CMD** *condition modifier*; the modifier for the condition.



**CNL** Controlled Natural Language.

**CNN** Convolutional Neural Network.

**CPT** *condition patient*; the entity affected by CAC.

**CPU** Central Processing Unit.

**CRC** Colorectal Cancer.

**CSP** Communicating Sequential Processes.

**CTL** Computation Tree Logic.

**CTV** *condition to value*; the value of CPT after performing CAC.

**CUP** Construction of Useful Parsers.

**DFRS** Data-Flow Reactive Systems.

**EHR** Electronic Health Records.

**FE** Frame Element.

**FOL** First-Order Logic.

**GloVe** Global Vector.

**GPU** Graphics Processing Unit.

**HbA1c** Glycated Haemoglobin.

**IAA** Inter-Annotator Agreement.

**ICU** Intensive Care Unit.

**IMR** Intermediate Model Representation.

**IRC** Intravenous Catheters.

**LSTM** Long Short-Term Memory.

**LTL** Linear Temporal Logic.

**LU** Lexical Unit.

**MBT** Model-Based Testing.

**ML** Machine Learning.

**NAT2TEST** Natural Language Requirements to Test Cases.

**NER** Named-Entity Recognition.

**NICE** National Institute for Health and Care Excellence.

**NLP** Natural Language Processing.

**NP** Noun Phrase.

**NUM** the number for the sentence sequence.

**PAT** *patient*; the entity who is affected by the action in a sentence.

**PL** Propositional Logic.

**POS** Part-Of-Speech.

**PROMELA** Process Meta Language.

**RNN** Recurrent Neural Network.

**S-CASE** Scaffolding Scalable Software Services.

**SAT** Boolean Satisfiability Problem.

**SCR** Software Cost Reduction.

**ShARe** Shared Annotated Resources.

**SIGN** Scottish Intercollegiate Guidelines Network.

**SMT** Satisfiability Modulo Theory.

**SMV** Symbolic Model Verification.

**SRL** Semantic Role Labelling.

**T2D** Type 2 Diabetes.

**TMR4I** Transition-based Medical Recommendations for detecting Interactions.

**TOV** *to value*; the value of PAT after action completion.

**UML** Unified Modelling Language.

**UMLS** Unified Medical Language System.

**XML** Extensible Markup Language.

*“Begin at the beginning,” the King said, very gravely, “and go on till you come to the end: then stop.”*

Lewis Carroll, *Alice in Wonderland*

# 1

## Introduction

Clinical guidelines are evidence-based recommendations developed to assist health care providers treating patients with different conditions, and are commonly used for documenting how to treat chronic conditions (Institute of Medicine, 1990). They provide guidance such as what drugs should be prescribed for a particular condition, how long such treatment should be given, what tests should be conducted and when, or other situational clinical circumstances for certain diseases. This makes clinical guidelines an official source where standard treatment procedures are documented, developed and revised over time when necessary. In the United Kingdom, clinical guidelines are published by the National Institute for Health and Care Excellence (NICE)<sup>1</sup> for England, Wales and Northern Ireland, and the Scottish Intercollegiate Guidelines Network (SIGN)<sup>2</sup> for Scotland. Further development of local and regional guidelines would be based on these national ones.

As clinical guidelines are disseminated in human natural language, they are often ambiguous and/or incomplete. For example, a blood glucose lowering therapy for people with Type 2 Diabetes (T2D) does not include recommendations which assume patients are able to recover and improve their condition, or the drugs are recommended

---

<sup>1</sup><https://www.nice.org.uk/>

<sup>2</sup><https://www.sign.ac.uk/>

without explicit dosage information. Furthermore, they usually assume that health practitioners understand how to best follow guidelines. Since one clinical guideline is normally associated to a single disease, it is also a challenge to consult the guideline when treating patients with multiple chronic conditions, also known as multimorbidity, as the health practitioner needs to consider multiple guidelines for the different conditions to be able to provide a more suitable and targeted recommendation.

It is therefore important to produce sound clinical guidelines, and representing clinical guidelines and their underlying intentions in a standard, machine-readable, and machine-interpretable way is crucial for dissemination of clinical knowledge, and can improve the quality of care. As an example, in a multimorbidity case where a patient experiences several diseases at the same time, clinical practitioners may consult several guidelines at the same time to treat such particular patients. In some cases, the recommendations in a guideline may contradict others in another guideline. This figure is becoming more prominent with the increase of chronic diseases.

One way to ensure correctness and consistency in recommendations are by using automated methods to reason about them. One approach in this context is through formal verification, for instance model checking where a system is modelled mathematically and some properties are checked against all states and transitions in the model. Another approach is by specifying what is held as true by the system as logical formulae and using a constraint solver to prove mathematically that the system conforms to the specification.

To formally check a clinical guideline, we can either build a mathematical model and define some properties to check against the model, or write logical formulae to represent the guideline. Nevertheless, this is not a trivial task as it demands some background knowledge from the users or health practitioners in mathematics, logic or in the verification tool being used. Another way to do this is by structuring clinical guidelines according to some predefined grammar rules so that they can be transformed into a formal model automatically. However, this approach can also limit the expressiveness of the guidelines to some extent, hence make it (slightly) less natural and not enticing for health practitioners to use it as someone still needs to convert the original source into the predefined rules.

Recent advances in the field of Natural Language Processing (NLP) and machine learning open possibilities to manipulate texts to achieve some tasks. This enables text processing from a very rigid way by defining sets of rules to learning from data. This is ranging from part-of-speech tagging, parsing, sentiment analysis, recognising textual entailment, and question answering. Using NLP, we can leverage the process of transforming clinical guidelines into formal specifications to minimise the burden of requiring the mathematical knowledge from users while keeping the guidelines written as close as possible to human natural language.

## 1.1 Objectives

This thesis aims to facilitate the formal verification of clinical guidelines, specifically the clinical guidelines written in English. Concretely, we will create a framework that can (semi) automatically transform clinical guidelines into formal specifications. This consists of two main issues: (1) capturing the key concepts in the texts, (2) transforming these concepts into formal models and verifying it against some properties.

To address issue (1), we will study several techniques that mainly come from the software engineering field. As human activities are getting more and more involved with and dependent on computer software, research in automatic software verification and test generation from its textual requirements is getting more attention in recent years. We will also look at current techniques in NLP in tandem with machine/deep learning to help us annotate concepts of interest in clinical guidelines more effectively.

To address issue (2), we need to create procedures that enable the conversion of the concepts from issue (1) to several types of formal specifications. Each specification will tie very closely to the tool it is run on and each has its own degree of expressiveness. The purpose of having various specifications is a proof of concept that the output of (1) can be used to generate any specification form that is sufficient for our current task at hand. We also will create properties in formal languages (logics) that is accepted by different tools to verify the generated formal model.

Figure 1.1 shows the whole framework in our research. This framework displays

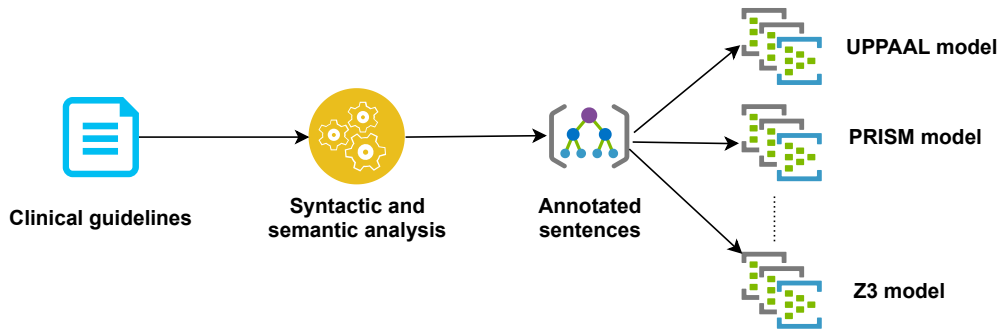


Figure 1.1: The framework for verifying clinical guidelines

the flow when clinical guidelines are first processed by the syntactic and semantic analysis part. This further generates the annotated version of the clinical guidelines. Finally, the last part of the framework shows that the annotation can be converted into several formalism models to suit different needs.

## 1.2 Outline

Over the next chapters, this thesis will go into detail on how we investigate and address the objectives that we want to achieve. This thesis is organised as follows:

**Chapter 2** This chapter lays the related work that sets the foundations and motivations of our work to model clinical guideline texts. We investigate two different domains namely software requirements and biomedical data. We discuss common techniques used to manipulate texts in these two domains and how they can be used to generate formal specification. In the last part of this chapter, we state the approaches that we will adapt in our own work.

**Chapter 3** We describe our first technique to annotate the main concepts in clinical guidelines using Controlled Natural Language (CNL). The advantage of using CNL is that the development does not depend on the size of available data. This is very suitable for our situation where our clinical guideline sentences are very limited. The downside of this is that creating grammar rules is not easy as they need to accommodate most (if not all) possible sentence patterns. It is also obvious that the grammar will restrict the expressiveness of the guideline texts. A part of this chapter is published in Rahman and J. K. F. Bowles (2017).

**Chapter 4** To add more flexibility, we adapt NLP techniques for tagging concepts in texts using Semantic Role Labelling (SRL), Named-Entity Recognition (NER) and relation classification. We introduce the concepts that we use throughout the thesis. We investigate several features that we use when training the SRL model. We also describe the neural architecture for our NER and relation classification models. Finally, we discuss the fine-tuning approach from several state-of-the-art deep learning models. A part of this work has been published in Rahman and J. Bowles (2021a).

**Chapter 5** To evaluate the performance of the models that we build, we conduct several experiments. Firstly, we discuss the statistical nature of our dataset to get a basic understanding and to discover patterns and anomalies. This is important due to the small size of our dataset, therefore, we can make rough predictions of what we will get when running the experiments. We then explain the experiments on SRL, NER and relation classification and discuss the performance of our models. As expected, fine-tuning state-of-the-art models gives better performance. However, due to the computation resource needed for fine-tuning, in some cases, this is not always the most favourable option. Some parts of this chapter have been published in Rahman and J. Bowles (2021a) and Rahman and J. Bowles (2021b).

**Chapter 6** We explain the procedures for converting the annotated text from the guidelines into formal specifications in this chapter. Firstly, we provide some background on the underlying formalism and logics used. These logics will be used when specifying the properties to be checked against the formal models. We then discuss the verification step using model checking tools such as UPPAAL (Behrmann et al., 2004) and PRISM (Kwiatkowska et al., 2011) as well as constraint solver such as Z3 (Moura and Bjørner, 2008). Which approach is more suitable depends on the guideline text itself and how much information on treatments is available. A part of this chapter is published in Rahman and J. K. F. Bowles (2017).

**Chapter 7** The summary of our findings with research directions for future work.



*...comme l'on dit, il est bien facile, et même nécessaire de voir plus loin que nos devanciers, lorsque nous sommes montés sur leurs épaules...*

Marin Mersenne

# 2

## Related Work

This chapter describes the background that sets the foundations and motivations of our work to model clinical guideline texts. We start by discussing the approaches done in the field of software requirement modelling from natural language (Section 2.1), which involves automatic semantic annotation and test case generation. We then discuss the importance and challenge of implementing Natural Language Processing (NLP) for healthcare data as well as current related approaches in the field (Section 2.2).

### 2.1 Modelling Software Requirements

In the software engineering process, understanding requirements is one of the most fundamental steps as it drives further all the development stages that follow. Nevertheless, it is also one of the most difficult tasks when building a software system or any system in general (Pressman and Maxim, 2015). In the requirements engineering stage, the scope and the functionality of the system that will be developed are expected to be defined as clearly as possible. Figure 2.1 shows a side-by-side illustration of the waterfall software engineering process and requirement engineering process.

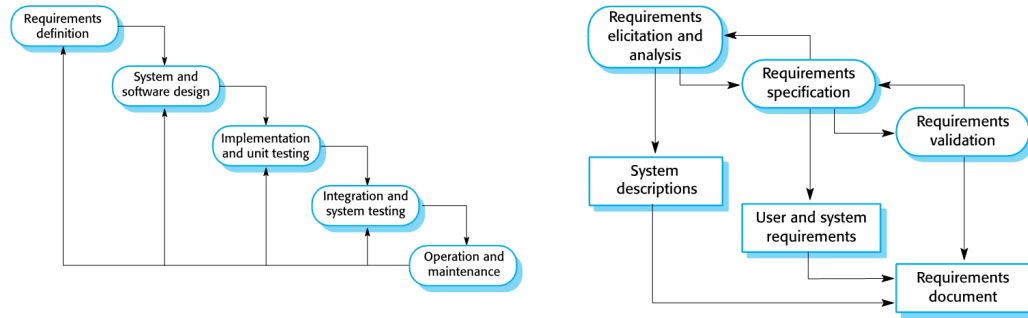


Figure 2.1: The left image shows the classical waterfall process in software engineering whereas the right image illustrates the requirement engineering process consisting of requirement elicitation and analysis, specification and validation (Sommerville, 2016).

Albeit software requirement engineering aims to get a clear, complete and consistent set of requirements, in reality, this is very rarely achieved. One of the reasons for this is that requirements are often written in natural language, hence, they inherit the ambiguous nature of human language. There are other alternatives to express requirements, such as formal mathematical models and graphical models such as use case diagrams and sequence diagrams from Unified Modeling Language (UML; Booch et al., 2005). Mathematical formalisms often require additional understanding and are not straightforward to use. Conversely, semi-formal approaches such as UML are still potential sources of ambiguity or may be incomplete. Even though both cases would minimise the ambiguity in the requirements, in most cases, they can still be difficult to understand by the various stakeholders.

Although it is common to start the next stage of software development after finishing the prior stage as in the waterfall model (Royce, 1970), an incremental development where one stage can still be refined and evolved based on the user’s feedback is also increasingly common especially for more complex systems. Incremental approaches include Agile models such as Extreme Programming (Beck, 1999), Scrum (Schwaber and Beedle, 2002), Kanban (Brechtner, 2015) and DevOps (Kim et al., 2016). The ability to eliminate inconsistencies early can reduce the cost of a later correction or rework, and reduce time spent in implementation and testing phases. The use of formal verification techniques to automatically identify inconsistencies or incomplete requirements is natural at a design stage but not common at natural language requirements.

Research in verifying a software system, or any system in general, has received much attention over the past decade (Beyer, 2020). It is driven by the fact that technology is becoming more and more integrated into our daily life. Hence, the reliability of a system is essential especially in a safety-critical environment or when failures can lead to considerable financial and business losses. Verification is a crucial factor in the hardware industries but it is still received less traction for the software industries in spite of the fact that software debugging in a later software development phase can cost a company billion per year (Brady, 2013). We have witnessed some catastrophes caused by faults in software systems and every year more reports of software failures are identified: from banking to air traffic management, to space and healthcare systems as can be seen in the list below:

- the crash of the space launch vehicle Ariane 5 due to an overflow computation (Gleick, 1996),
- a glitch in the Royal Bank of Scotland (RBS) computer systems caused a man to stay in prison (News, 2012) or blocked from a hotel checkout missing a flight home (Boyce, 2012),
- the recall of 466,000 Toyota cars due to braking issues (Online, 2014),
- the state-wide false alarm about a ballistic missile causing mass hysteria (Rosa, 2018),
- the death caused by the incapability of a self-driving car to identify a pedestrian (McCausland, 2019)

On the other hand, for all exposed catastrophic impacts, we can also see what software systems have been trying to achieve throughout times. We also witness how formal verification can be applied in crucial systems such as in software development for railway signalling systems (Dehbonei and Mejia, 1970), in a medical device control system (Jacky, 1995), in developing an air traffic control (ATC) system (Hall, 1996) and in requirements modelling for spacecraft fault protection system (Easterbrook et al., 1998).

As we are interested in improving the correctness of the requirements capturing phase and removing any sources of errors that may arise at that level, we detail known

approaches across the NLP context. In the next section, we present one approach to analyse requirement texts using a controlled natural language. The work is aimed at generating test cases automatically from requirements. A different approach to analyse requirements using machine learning is explained in Section 2.1.2 (p. 12).

### 2.1.1 Automatic Test Cases Generation from Requirements

One way to ensure the correctness of a system is by testing it under every possible scenario. For critical systems, testing is also aimed at ensuring their safety and reliability. Recent work by Carvalho (2016) shows how test cases can be generated automatically from software requirements written in a controlled natural language. The author argues that despite there being testing techniques that can be used to generate and execute test cases automatically, such as for model-based testing (MBT; Utting and Legeard, 2007), most of the time, the model syntax and semantics are not very well known to be used by engineers and stakeholders. Furthermore, as most requirements are specified in natural language, these models are not available at the start of the project. To overcome these shortcomings, Carvalho (2016) adapts the MBT technique with NLP to extract the models directly from the requirements. This approach is shown in Figure 2.2.

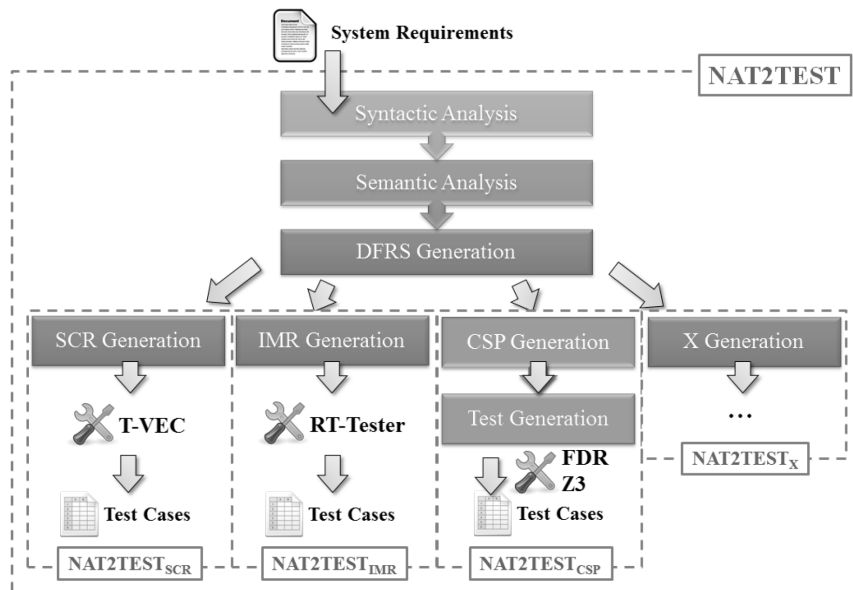


Figure 2.2: Phases of the NAT2TEST strategy (Carvalho, 2016)

Figure 2.2 shows the Natural Language Requirements to Test Cases (NAT2TEST)

strategy. The fundamental idea of NAT2TEST is to generate test cases for Data-Flow Reactive Systems (DFRS), i.e., a class of embedded systems whose inputs and outputs are signals provided by sensors and actuators that can have discrete or continuous time-based behaviour. The output of the strategy can be translated into several formal notations, for example, a formal method based on tables for the specification and analysis of the required behavior of complex software systems called Software Cost Reduction (SCR; Heitmeyer et al., 1998), an abstract syntax called Intermediate Model Representation (IMR) as the input of a test case generator for reactive systems named RT-Tester (Peleska et al., 2011), or process algebras such as Communicating Sequential Processes (CSP; Roscoe, 1997). The translation is done through a series of NLP techniques, namely syntactic analysis and semantic analysis. These strategies are further explained in the subsequent subsections.

#### 2.1.1.1 Syntactic analysis

One way to process system requirements automatically requires that every sentence be written according to some grammar rules. In NAT2TEST, the grammar is defined as SysReq-CNL, a Controlled Natural Language (CNL) designed for editing DFRS requirements. A CNL is a constructed language based on a certain natural language, which has more restriction in terms of lexicon, syntax, and/or semantics while preserving most of its natural properties (Kuhn, 2014). It can be used to improve communication among humans as well as to provide a clearer representation of formal notations. As the expected format of the sentences in this language is clear, we are also closer to a language that can be used in automated tasks.

The CNL is composed of Context-Free Grammar (CFG) with predefined vocabularies from the application domain. The vocabularies fall into several lexical categories or Part-Of-Speech (POS) tags such as *DETER* for determiner, *NSING* and *NPLUR* for singular and plural noun, *ADJ* for adjective, *ADV* for adverb, *CONJ* for conjunction, *PREP* for preposition, and *VBASE* and *VPRE3RD* for verb in base form and verb in 3rd person singular present form. There are also several additional categories to simplify the analysis process, namely *NUMBER* for numbers, *COMP* to mark comparisons, and grammatical keywords such as *AND*, *OR*, *NOT*, *SHALL*, *COLON* and *COMMA*.

To illustrate, the following is an example of a requirement sentence in the original and the SysReq-CNL form (Carvalho, 2016). For every sentence written in the SysReq-CNL form, the syntactic analysis will generate a parse tree as the output which is then used in the semantic analysis step discussed next.

**Original:** "The Priority Logic Function shall assign value 0 (zero) to Command In-Control output when: left Priority Button is not pressed AND right Priority Button is not pressed AND left Command is on neutral position AND right Command is on neutral position."

**SysReq-CNL:** "When the left priority button is not pressed, and the right priority button is not pressed, and the left command is on neutral position, and the right command is on neutral position, the Priority Logic Function shall assign 0 to the Command-In-Control output."

The most visible difference between the two forms is the placement of the conditions. In the original sentence, the system action comes first followed by the conditions that should be fulfilled for the action to be executed, whereas the orders are reversed in SysReq-CNL. This rewriting resembles the notion of guard of an action or an if statement structure in programming. Another difference is the reduction and standardisation of the punctuation used in the sentence. The removal of the colon (highlighted in orange red) and the lower casing of AND (highlighted in cyan) are examples of this case.

### 2.1.1.2 Semantic analysis

The semantic analysis step will assign a thematic role, e.g., *agent*, *patient*, *location*, *object*, to every word or group of words that is affected by a verb as described in the case grammar theory (Fillmore, 1968). Two groups of thematic roles are created in association to the action or condition statements in the SysReq-CNL grammar.

Table 2.1 shows the thematic roles and their definition. All thematic roles associated with a verb are then grouped into a case frame. A case frame describes all possible roles accepted by a particular verb. A verb can only have one case frame whereas several verbs can have or share the same case frame. For example, the verb 'add'

	Role	Definition
Action	ACT	the performed action if the conditions are satisfied
	AGT	the entity who performs the action
	PAT	the entity who is affected by the action
	TOV	the value of PAT after action completion
Condition	CAC	the action for enabling a condition
	CPT	the entity affected by CAC
	CFV	the value of CPT before performing CAC
	CTV	the value of CPT after performing CAC
	CMD	the modifier for the condition

Table 2.1: Thematic roles for action and condition statements (Carvalho, 2016)

and 'subtract' in mathematical context have the same arguments, hence they have the same case frame.

The final output of the syntactic analysis step is a structure called a requirement frame. A requirement frame contains as many case frames as the number of verbs in that particular requirement sentence. By taking the value of each thematic role in a requirement frame, NAT2TEST can generate a DFRS which can be further translated into a requirement model in several formalisms. Figure 2.3 shows a requirement frame for the requirement below (the conditions part are highlighted in cyan, the actions in orange red). There are two case frames for the condition: one for the verb *is* and another for the verb *changes*, and two for the action: one for the verb *reset* and another for the verb *assign*.

"When the system mode is idle, and the coin sensor changes to true, the coffee machine system shall: reset the request timer, assign choice to the system mode."

The whole NAT2TEST strategy is for automated test case generation from software requirements. For our study, we adapt the initial stages of NAT2TEST, namely the syntactic and the semantic analyses, for constructing a CNL in a clinical guidelines domain.

## 2.1.2 Semantic Annotation in Software Requirements

Similar to the work of Carvalho (2016) described in Section 2.1.1 (p. 9), Diamantopoulos et al. (2017) try to detect problems in software requirements at an early stage,

<b>Condition #1</b> - Main Verb (CAC): is		
CPT:	the system mode	CFV: -
CMD:	-	CTV: idle
<b>Condition #2</b> - Main Verb (CAC): changes		
CPT:	the coin sensor	CFV: -
CMD:	-	CTV: true
<b>Action</b> - Main Verb (ACT): reset		
AGT:	the coffee machine system	TOV: -
PAT:	the request timer	
<b>Action</b> - Main Verb (ACT): assign		
AGT:	the coffee machine system	TOV: choice
PAT:	the system mode	

Figure 2.3: An example of a requirement frame (Carvalho, 2016)

but follow a very different strategy by using Machine Learning (ML). They adapt the semantic role labelling (SRL; Gildea and Jurafsky, 2002) technique to automate the annotation of software requirement texts as well as the underlying mapping to formal representation. This work is a part of a project called Scaffolding Scalable Software Services (S-CASE)<sup>1</sup> comprising from the front-end part that handles multi-modal requirements input to the back-end that generates source code. Figure 2.4 shows the architecture of S-CASE. The following subsections explain their approach to automate semantic annotation in software requirements which are included in the Req2Specs Module in S-CASE.

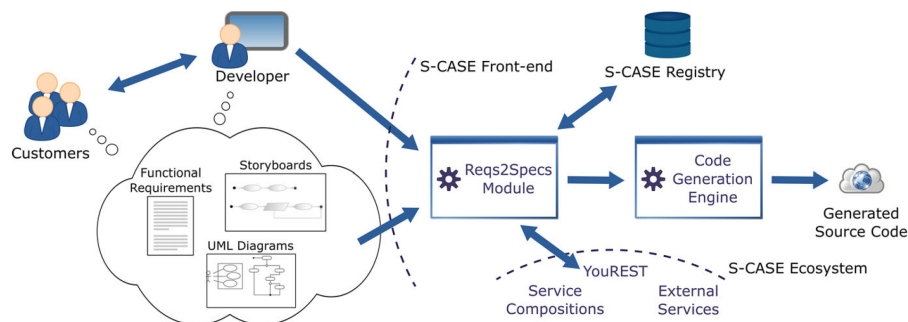


Figure 2.4: The architecture of S-CASE (Diamantopoulos et al., 2017)

<sup>1</sup><https://www.cloudwatchhub.eu/serviceoffers/s-case-scaffolding-scalable-software-services>



### 2.1.2.1 Ontology of concepts

In S-CASE, to annotate the software requirements, a hierarchy of ontologies starting from the most generic to the most specific are defined. These ontologies represent the structure of concepts of the software. For S-CASE, the concepts are designed to capture the relationships between an agent performing some action(s) on some object(s).

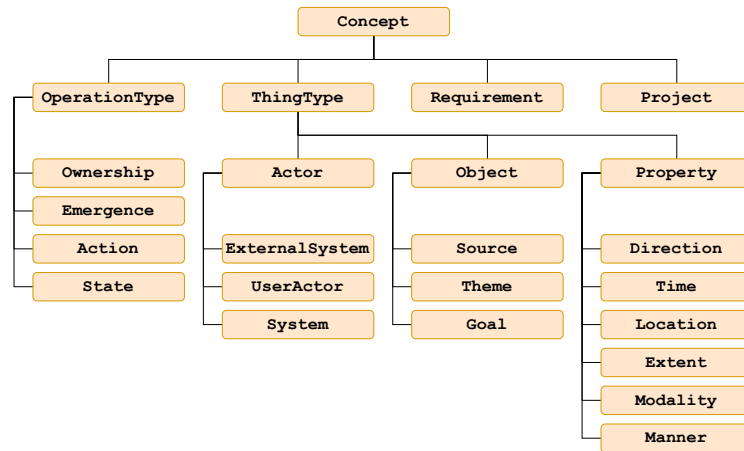


Figure 2.5: Ontology of Concepts (figure readjusted from Diamantopoulos et al., 2017)

S-CASE uses the ontology shown in Figure 2.5, where every entity is seen as a **Concept**. A **Concept** can be further specialised as either a **Project**, a **Requirement**, a **ThingType** or an **OperationType**. The **Project** and the **Requirement** refer to the software project and its requirements, whereas **ThingType** refers to the actor of an action, an object that is acted upon, or the property of an actor or an object, and **OperationType** refers to the actions performed by actors. Table 2.2 shows a subset of relationships between the ontology concepts in S-CASE. Here, we only show what concepts and their relationships that are relevant for our adaptation in our study. Every relationship in the domain has an inverse, that is, for each relationship from concept A to concept B, there is also a reverse relationship from concept B to concept A.

Once the concepts are clearly defined, then the software requirements are annotated accordingly. These requirements are taken from real software projects and in-class assignments from various universities associated with the project. The final annotation for this task only uses **Actor**, **Action**, **Object** and **Property** concepts. This is

because the more detailed concepts introduced, the more difficult it is to differentiate between those concepts. In other words, a fine-grained concept hierarchy will introduce complexity when trying to distinguish them.

Concept	Relationship	Concept
Action	acts_on	Object, Property
Object, Property	receives_action	Action
OperationType	has_actor	Actor
Actor	is_actor_of	OperationType
ThingType	has_property	Property
Property	is_property_of	ThingType

Table 2.2: Some relationships between concepts in S-CASE.

### 2.1.2.2 Syntactic analysis

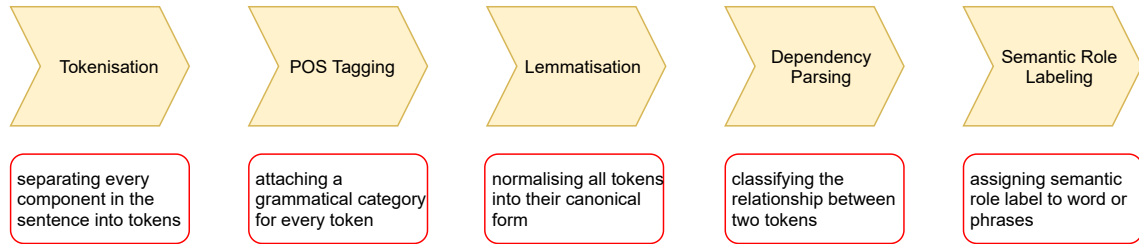


Figure 2.6: Syntactic analysis in the Mate Tools pipeline

To perform the syntactic analysis, Diamantopoulos et al. (2017) use the Mate Tools<sup>2</sup> (Björkelund, Bohnet et al., 2010) that provide functionalities to perform tokenisation, POS tagging, lemmatisation, dependency parsing, and semantic role labelling for a given sentence. The tools have been proven to achieve state-of-the-art performance on the shared task of syntactic and semantic dependencies in multilingual texts (Hajic et al., 2009). Figure 2.6 shows the modules provided in the Mate Tools. For the syntactic analysis, Diamantopoulos et al. (2017) use the tokenisation, POS tagging, and lemmatisation modules, and train their own semantic role labelling module for the software requirements domain. As the syntactic analysis steps are ubiquitous in many NLP preprocessing tasks, we will also use them when building our own system.

The outcome of this step will be used as the features for the semantic analysis step to identify the ontology of concepts in the sentences. Further explanation on this and its implementation in our domain are discussed in Section 4.1.3 (p. 43).

<sup>2</sup><https://code.google.com/archive/p/mate-tools/>

## 2.2 Natural Language Processing in Healthcare Data

In addition to the advances that technology has brought in many fields, e.g., in banking and transportation, it can also bring breakthroughs in the medical field, particularly with the aid of increased recent data processing power. One factor to accelerate the discovery of new insights is the ability to extract valuable information from patient electronic health records (EHRs). Nevertheless, the nuggets of information for the most part are still hidden in free-text form and it remains a challenge to capture this information.

A review study conducted by Demner-Fushman, W. W. Chapman et al. (2009) showed that applying fundamental NLP strategies on EHRs can be used to develop Clinical Decision Support (CDS) systems that benefit health care providers and the general public. A subset of EHRs consisting of medical history, physical examination, and chest radiography results are commonly obtained in the free-text form. The authors argue that the two biggest challenges to process free-text EHRs are the domain-specific abbreviations that clinicians are using and privacy issues. Despite these challenges, the benefits of incorporating CDS systems can range from identifying clinically relevant entities in clinical notes, extracting details on family history from discharge summaries, monitoring of clinical events to detect and prevent adverse events, and processing radiology or pathology reports, to name but a few (Demner-Fushman, W. W. Chapman et al., 2009).

Another study incorporating NLP to improve Colorectal Cancer (CRC) screening and surveillance also shows a promising result (Imler et al., 2015). While it still needs further investigations, another study has shown how text mining on multiple clinical data resources can detect the link between the use of proton pump inhibitors and the increased risk of myocardial infarction (Shah et al., 2015).

The next section discusses some notable work in formal checking for clinical guidelines. Further work on semantic annotation in biomedical texts will be discussed in detail in Section 2.2.2 (p. 18).

### 2.2.1 Formal Checking for Clinical Guidelines

Considerable research in modelling and formalising clinical guidelines have been done over the past years. Bäumlér et al. (2006) applied formal modelling and verification to improve the quality of medical guidelines. The guideline representations are written in Asbru, a predefined language designed especially for the medical domain (Shahar et al., 1998). Asbru is able to express durative actions and the intentions in the guideline plans. With specifications formulated in Action Computational Tree Logic (ACTL; De Nicola and Vaandrager, 1990), the model is then verified using the Cadence Symbolic Model Verification (SMV) model checker (McMillan et al., 2000). The example below illustrates Asbru syntax for the parameter proposition: "high blood-glucose level of any type in the context of therapy for GDM type II for more than 7 days in the period from 24 conception weeks to delivery, using the estimated conception date as a reference point" (Miksch et al., 1997):

```
(STATE(blood-glucose) HIGH GDM-Type-II [[24 WEEKS, 24 WEEKS],  
[DELIVERY, DELIVERY], [7 DAYS, _], CONCEPTION])
```

Another implementation of model checking to verify clinical guidelines is done by Giordano et al. (2006). They use the GLARE language (Terenziani et al., 2001), a domain-independent prototypical system for acquiring, representing, and executing clinical guidelines. With an Extensible Markup Language (XML) intermediary layer which then is translated to Process Meta Language (PROMELA), the model and its properties are verified using the SPIN model checker (Holzmann, 2003).

Pérez and Porres (2010) created a framework to enable authorisation and verification of clinical guidelines. The guidelines are modeled as UML statecharts following common representation patterns found in clinical guidelines from the natural language expressions. Figure 2.7 shows an example of a statechart model from the Intravenous Catheters (IRC) guideline. The statechart model is chosen as an intermediate representation to ease the burden of the non-expert to write formal specifications. The statechart model is then transformed into PROMELA and the properties that the model wants to check are translated into linear temporal logic (LTL; Pnueli, 1977). The verification of the guideline model is then performed using the SPIN model checker.

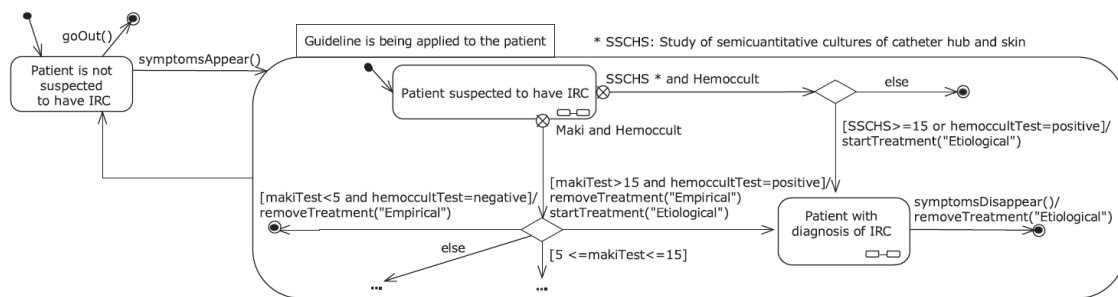


Figure 2.7: Part of the statechart defined from the Intravenous Catheters (IRC) guideline (Pérez and Porres, 2010)

Representing the guidelines as a statechart model or presenting them in some pre-defined languages such as Asbru or GLARE will make the verification process easier. However, this also gives additional restrictions and / or limitations on how the guidelines should be expressed to conform to the model or language being used. Furthermore, none of these is easy to understand for clinicians or health practitioners when developing the clinical guidelines.

Although formal verification is also widely used across other domains, we will restrict its use in the clinical guideline domain for our study. In addition to the model checkers, where the system model is inspected against some properties, we will also consider SAT and SMT solvers as verification tools where the clinical guidelines are defined as a list of constraints that need to be solved. In our case, model checkers are used mainly to verify clinical guidelines that have strict recommendation orders whereas SAT and SMT solvers are more favourable otherwise.

## 2.2.2 Semantic Annotation in Biomedical Texts

Research on finding key concepts in biomedical texts has delivered notable results for the past two decades. Since the creation of the Unified Medical Language System (UMLS) Thesaurus (Lindberg et al., 1993), many systems have been developed to identify and extract those concepts from biomedical texts. Some of these systems are freely available and have been used as the basis for many study in medical texts, such as the MetaMap (Aronson, 2006), cTAKES (Savova et al., 2009), MetaMapLite (Demner-Fushman, Rogers et al., 2017) and Bio-YODIE (Gorrell et al., 2018). A notable shared task for identifying and extracting UMLS disorder from a Shared

Annotated Resources (ShARe) corpus has also been developed in the past (Pradhan et al., 2014; Elhadad et al., 2015).

Other tasks are ranging from recognising protein names, structures and functions (Gaizauskas et al., 2003; Valencia, 2005), drugs (I. Segura-Bedmar, Martinez and M. Segura-Bedmar, 2008), diseases (Gorinski et al., 2019), recommendations in clinical guidelines and their corresponding strength of importance (Read et al., 2016), a wide range of eleven clinical entities (Patel et al., 2018). There is also research on the extraction of protein-protein interactions (Bui et al., 2010), the relations between drugs, genes and cells (Friedman et al., 2001), the relations between drugs and diseases (Shah et al., 2015).

The documents of interest also come from multiple domains, such as an Intensive Care Unit (ICU) department clinical notes (Wang, 2009), the ShARe corpus (Pradhan et al., 2014; Elhadad et al., 2015), large multi-source clinical documents (Patel et al., 2018), or the radiology reports (Alex et al., 2019; Gorinski et al., 2019).

The techniques used to accomplish the task also range from dictionary checking with all possible variations (Aronson, 2006; Demner-Fushman, Rogers et al., 2017; Gorrell et al., 2018), handcraft rule-based technique (Friedman et al., 2001; Alex et al., 2019; Becker, Böckmann et al., 2020), text mining (Shah et al., 2015), as well as machine learning (Bui et al., 2010; Settles, 2004; Pradhan et al., 2014; Elhadad et al., 2015; Read et al., 2016; Gorinski et al., 2019). Some use the combination of these approaches, such as augmenting the rule-based output for the machine learning step or using the rule-based for post-processing the machine learning output (Becker and Böckmann, 2017a).

In terms of extracting concepts in clinical guideline, some studies have been conducted such as parsing clinical guidelines using Named-Entity Recognition (NER) and relation classification to encode diagnosis and treatment recommendations (Taboada et al., 2013), or personalising treatment recommendations from clinical guidelines written in German (Becker and Böckmann, 2017a; Becker and Böckmann, 2017b) which further developed to personalising treatment for patient with CRC (Becker, Böckmann et al., 2020). These studies are conducted fully automated using several

open-source NLP tools, namely OpenNLP<sup>3</sup>, Stanford NLP<sup>4</sup>, SemRep<sup>5</sup> by Taboada et al. (2013) or a combination of rule-based and ML-based using cTAKES<sup>6</sup> by Becker and Böckmann (2017a), Becker and Böckmann (2017b) and Becker, Böckmann et al. (2020).

In addition to the work by Becker and Böckmann (2017a), Becker and Böckmann (2017b) and Becker, Böckmann et al. (2020) for clinical guidelines written in German, Borchert et al. (2020) develop GGPONC (German Guideline Program in Oncology NLP Corpus), a large corpus composed of German medical texts taken from clinical guidelines in oncology. This work is driven by the lack of publicly accessible corpora in medical domain other than English. This corpus contains rich structural information and metadata that can facilitate the development of ML-based NLP algorithms for clinical text in German.

A notable study to detect interactions between recommendations in several guidelines using Semantic Web technologies has been conducted by Carretta Zamborlini et al. (2016). They create the Transition-based Medical Recommendations for detecting Interactions (TMR4I) using Semantic Web to model clinical guidelines as well as to enable components reuse, combination and reason. Table 2.3 (p. 21) shows the concepts that are used in this study. By using these concepts, they can capture interactions in several guidelines such as *contradiction*, *repetition* or *alternative* interactions. TMR4I can then check internal interactions, i.e., interaction between the given recommendations, for example the contradiction between "*lower blood pressure*" and "*increase blood pressure*". It can also check the external interactions based on drug-drug interactions, e.g., Aspirin is incompatible with Ibuprofen.

To evaluate TMR4I, they conducted experiments to check merged guidelines between Duodenal Ulcer (DU) and Transient Ischemic Attack (TIA) as well as Osteoarthritis (OA), Hypertension (HT) and Diabetes (DB). Their experiments show that TMR4I can detect contradictions and give recommendation based on the detected interactions.

---

<sup>3</sup><https://opennlp.apache.org/>

<sup>4</sup><https://stanfordnlp.github.io/CoreNLP/>

<sup>5</sup>[https://lhncbc.nlm.nih.gov/ii/tools/SemRep\\_SemMedDB\\_SKR/SemRep.html](https://lhncbc.nlm.nih.gov/ii/tools/SemRep_SemMedDB_SKR/SemRep.html)

<sup>6</sup><https://ctakes.apache.org/index.html>

Concept	Description	Example
Situation Type	Represents a property and its admissible values	
Transformable Situation Type	Regards the situation that is expected to be changed	<i>Patient's temperature is over 37 degrees</i>
Non-Transformable Situation Type	Regards the situation that holds as filter condition	<i>Patient's age is over 10 years old</i>
Post-Situation Type	Regards the situation that is expected to be achieved	<i>Patient's temperature is below 37 degrees</i>
Care Action Type	Represents the action types that can be performed by health care agents in order to change a situation	<i>Administer aspirin</i>
Transition	Represents the possibility of changing a situation regarding a patient by performing a care action type	<i>Administering aspirin in patient over 10 years old reduces its temperature below 37 degrees</i>
Recommendation	Represents a suggestion to either pursue or avoid a transition promoted by a care action type	

Table 2.3: Transition-based Medical Recommendations for detecting Interactions (TMR4I) Concepts Summary (Carretta Zamborlini et al., 2016).

Complementary to the work of Carretta Zamborlini et al. (2016) to identify and address interactions between multiple clinical guidelines, Wilk et al. (2017) also incorporate temporal aspects of the identified interactions that the former did not have. Furthermore, while Carretta Zamborlini et al. (2016) can detect and suggest *alternative* interactions, Wilk et al. (2017) leverage the former’s work further as they can discover if any interactions still exist by applying the detected *alternative* interactions.

Lastly, the recent work by Grivas et al. (2020) has some resemblance to our research. They build a rule-based information extraction system, EdIE-R, as well as neural systems EdIE-LSTM and EdIE-BERT to mark the *findings* (e.g., tumours or small vessel disease), *modifiers* (e.g., recent, old, deep, or cortical) and *negation* (e.g., if the *findings* are absent or present) in brain MRI radiology reports using NER and relation classification. Their experiments show that in a small dataset environment, the rule-based system outperforms the neural models. For the analogous NER and relation classification method to our work, the concept will be explained further in Section 4.2 (p. 51).



## 2.3 Summary

In this chapter, we presented related work that serves as the foundations of our research to formally check clinical guideline texts. We discussed several approaches to transform software requirements from written text into a formal model. We have also discussed past research on formal checking in clinical guidelines and semantic annotation in biomedical texts.

As far as we are aware, there is no research on process annotation in clinical guidelines. For this reason, we will adapt several approaches that have been discussed in this chapter when building our framework to formally check clinical guidelines. We start from the most rigid approach by making hand-craft rules and building a controlled natural language adapting the work of Carvalho (2016). For a less restrictive approach, we build a machine learning model for semantic role labelling following Diamantopoulos et al. (2017). Finally, we also train several neural network models for NER and relation classification analogous to the work of Grivas et al. (2020).

*Bélise — Veux-tu toute ta vie offenser la grammaire ?*

*Martine — Qui parle d'offenser grand'mère ni grand-père ?*

*Philaminte — Ô Ciel !*

Molière, *Les Femmes savantes*

# 3

## Controlled Natural Language for Clinical Guidelines

This chapter describes the implementation of a controlled natural language to mark the main information in clinical guidelines. This is the first approach followed and we discuss advantages and limitations later on. Part of the work presented in this chapter has been published in Rahman and J. K. F. Bowles (2017).

### 3.1 CNL in Clinical Guidelines

Following Carvalho (2016), we create a Controlled Natural Language (CNL) to standardise the sentence structure of clinical guidelines. The CNL will be used further to parse the guidelines to get their main information. We will take examples from the Type 2 Diabetes (T2D) therapy algorithm shown in Figure 3.1 taken from NICE<sup>1</sup> to guide us in the definition of our CNL.

Before the therapy algorithm is processed, every sentence is rewritten to conform with the CNL. The Context-Free Grammar (CFG) composing the CNL is explained

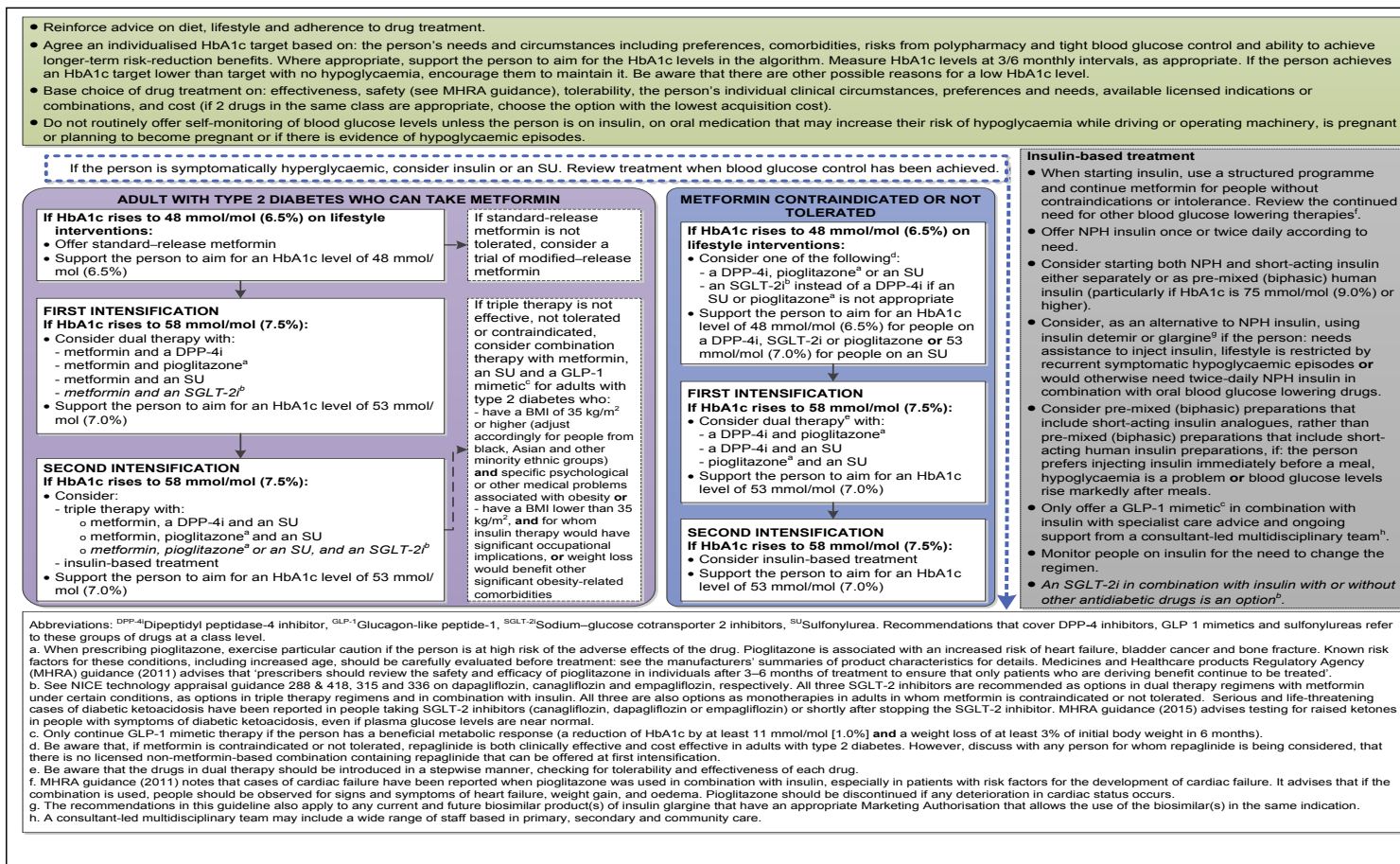
---

<sup>1</sup><https://www.nice.org.uk/guidance/ng28/resources/algorithm-for-blood-glucose-lowering-therapy-in-adults-with-type-2-diabetes-pdf-2185604173>

in the next section. Once all sentences are in agreement with the grammar, we parse the sentences to get the syntactic parse trees using the Modgrammar library<sup>2</sup>. Modgrammar is a Python library for constructing language parsers and interpreters for CFG definitions. By traversing the parse trees following our hand-crafted rules, the semantic analysis module will then construct the case frames. Once we have the case frames, we are able to reason with the information obtained in various ways as shown later in Chapter 6 (p. 83).

---

<sup>2</sup><https://github.com/rembish/modgrammar-py2>



**Insulin-based treatment**

- When starting insulin, use a structured programme and continue metformin for people without contraindications or intolerance. Review the continued need for other blood glucose lowering therapies<sup>f</sup>.
- Offer NPH insulin once or twice daily according to need.
- Consider starting both NPH and short-acting insulin either separately or as pre-mixed (biphasic) human insulin (particularly if HbA1c is 75 mmol/mol (9.0%) or higher).
- Consider, as an alternative to NPH insulin, using insulin detemir or glargine<sup>g</sup> if the person: needs assistance to inject insulin, lifestyle is restricted by recurrent symptomatic hypoglycaemic episodes or would otherwise need twice-daily NPH insulin in combination with oral blood glucose lowering drugs.
- Consider pre-mixed (biphasic) preparations that include short-acting insulin analogues, rather than pre-mixed (biphasic) preparations that include short-acting human insulin preparations, if: the person prefers injecting insulin immediately before a meal, hypoglycaemia is a problem or blood glucose levels rise markedly after meals.
- Only offer a GLP-1 mimetic<sup>c</sup> in combination with insulin with specialist care advice and ongoing support from a consultant-led multidisciplinary team<sup>h</sup>.
- Monitor people on insulin for the need to change the regimen.
- An SGLT-2i in combination with insulin with or without other antidiabetic drugs is an option<sup>f</sup>.

\*Type 2 diabetes in adults: management', NICE guideline NG28. Published December 2015, last updated April 2017.

© National Institute for Health and Care Excellence 2015. All rights reserved.

Figure 3.1: The National Institute for Health and Care Excellence Type 2 Diabetes (T2D) Therapy Algorithm

### 3.1.1 Syntactic Analysis

In the T2D therapy algorithm, we consider that every sentence has the following form:

**number if conditions, the doctor shall: actions**

The number in the beginning of every sentence is needed because the original algorithm is following a specific order. To illustrate this, in Figure 3.1 (the purple area with an automaton like structure, p. 25), a *first intensification* can only happen after the patient has received a *first treatment*. The same with a *second intensification* that only happens later after a *first intensification*.

Some considerations that we take when transforming all sentences into the standardised form are as follows: for every sentence, we add **the doctor** as the agent of the actions. In addition to explicitly showing who is responsible for a therapy, it is also needed to reduce the ambiguity from the missing information of who is giving the therapy versus who is receiving the therapy.

Furthermore, if there is a list of choices (conditions or possible therapies), we transform it into:

**{choice<sub>1</sub>, choice<sub>2</sub>, ... choice<sub>n</sub>}.**

For example, the conditions in "if triple therapy is not effective, not tolerated, or contraindicated" and the advice "consider therapy with a DPP-4i, pioglitazone, or an SU" will be written as: "*if triple therapy is {not effective, not tolerated, contraindicated}*" and "*consider therapy with {a DPP-4i, pioglitazone, an SU}*" respectively. The inclusion of articles as seen in: "*{a DPP-4i, pioglitazone, an SU}*" is to preserve the phrase structure in the original sentence, i.e., a noun phrase in this particular example, and to separate the noun from other categories in a `NounPhrase` (this will be explained further when discussing the `NounPhrase` category in our grammar). This normalisation will group the choices and make it easier to build the case frames from the parse tree.

Since our grammar is controlled and the lexicons are domain-dependent (i.e., a clinical guideline), we have specified in advance the vocabularies and their Part-Of-

Speech (POS) tags. Most of our POS tags follow those proposed by Carvalho (2016) with some additional categories to simplify the parsing of the sentences and the case frames generation, such as `LBrace` and `RBrace` for `{` and `}` respectively, `LPar` and `RPar` for `(` and `)` respectively, and `Percent` for the `%` sign. In addition to the POS tags, we also adapt the context-free grammar of Carvalho (2016) for our CNL with some modifications.

The whole context-free grammar is described in the box below. It follows the Backus-Naur form and regular expression quantification, where the pipe `|` means 'or', the `'?'` indicates zero or one occurrence of the preceding element, the `'*'` indicates zero or more occurrences of the preceding element, and the `'+'` indicates one or more occurrences of the preceding element.

```

Advice -> Number ConditionalClause Comma ActionClause
ConditionalClause -> Conj Condition
Condition -> NounPhrase VerbPhraseCondition
ActionClause -> NounPhrase VerbPhraseAction
VerbPhraseAction -> Shall Colon [VerbAction ToInfClause |
↪ VerbComplement | ChoiceAction ConditionalClause?]+
ChoiceAction -> LBrace PrepComplement [Comma VerbComplement]+ RBrace
VerbPhraseCondition -> VerbCondition Not? VerbComplement
ToInfClause -> To VBase VerbComplement
VerbComplement -> VariableState | ChoiceComplement | PrepComplement
PrepComplement -> [VariableState? Prep] VariableState |
↪ ChoiceComplement
ChoiceComplement -> LBrace VariableState [Comma VariableState]+ RBrace
PrepositionalPhrase -> Prep NounPhrase
VariableState -> AdjPhrase | NounPhrase
NounPhrase -> [Det? Adj* Noun PrepositionalPhrase*]+ [And NounPhrase+]*
VerbAction -> VBase
AdjPhrase -> [Adv?] Adj | VPart
VerbCondition -> VPre3 | VToBePre3
Noun -> Number | NSing | NPlur | NMass

```

The explanation for each symbol in the grammar with its set of lexicons is explained

next. We only use the subset of lexicons taken from the T2D therapy algorithm to illustrate the lexicons.

**Noun** A Noun represents the class of things which may be animate (e.g., doctor) or inanimate (e.g., metformin). It can also be countable as a singular (NSing) or a plural noun (NPlur) or uncountable as a mass noun (NMass). For our grammar, we also put the Number in this category. This inclusion is mainly for practical reason as now it is easier to combine any number and noun into a noun phrase. A number can be negative, real or natural, or a percentage.

```
Number -> '('? '-'? '[0-9]' '.'[0-9]? '%'? ')'?
NSing -> 'insulin' | 'SU' | 'doctor' | 'blood' | 'glucose' | 'control'
      ↪ | 'treatment' | 'metformin' | '\acrshort{hba1c}\index{Glycated
      ↪ Haemoglobin (HbA1c)}' | 'lifestyle' | 'therapy' | 'DPP-4i' |
      ↪ 'pioglitazone' | 'SGLT-2i' | 'patient' | 'GLP-1' | 'mimetic'
NPlur -> 'interventions'
NMass -> 'mmol/mol' | 'level'
```

**VerbCondition** A VerbCondition is the verb denoting the condition in a conditional clause. For example, the inflected verb *is* in "*if triple therapy is not effective*" is a VerbCondition. This is commonly the third person singular present verb (VPre3) or the third person singular present to be (VToBePre3).

```
VPre3 -> 'becomes' | 'rises' | 'takes'
VToBePre3 -> 'is'
```

**AdjPhrase** An AdjPhrase is the phrase associating an attribute to a noun. In our case, this could be a single adjective (Adj), a participle (VPart), or a combination between an adverb (Adv) and an adjective/participle.

```
Adv -> 'symptomatically'
Adj -> 'hyperglycaemic' | 'standard-release' | 'modified-release' |
      ↪ 'effective' | 'dual' | 'triple' | 'insulin-based'
VPart -> 'achieved' | 'tolerated' | 'contraindicated'
```

**VerbAction** Different from a **VerbCondition**, a **VerbAction** is the main verb in an action clause. Because we structure our sentence in the form: **number if conditions, the doctor shall: actions**, every action verb will be in its base form (VBase).

```
VBase -> 'consider' | 'review' | 'offer' | 'support' | 'aim'
```

**NounPhrase and PrepositionalPhrase** A **NounPhrase** is a word or group of words functioning as a noun in the sentence. It can be formed by a noun following a determiner (**Det**) and/or an adjective. The explicit use of a **Det** category is two-fold: keeping the original sentence as it is and separating the main noun in the **NounPhrase** from other categories. A phrase of a preposition followed by a noun phrase forms a **PrepositionalPhrase**. The **NounPhrase** and the **PrepositionalPhrase** can be formed recursively with either one of them. For example, "*HbA1c level*" is a **NounPhrase** and "*on lifestyle interventions*" is a **PrepositionalPhrase**.

```
PrepositionalPhrase -> ['from' | 'to' | 'on' | 'of' | 'for' | 'with']  
  ↪ NounPhrase  
Det -> 'a' | 'an' | 'the'  
NounPhrase -> [Det? Adj* Noun PrepositionalPhrase*]+ ['and'  
  ↪ NounPhrase+]*
```

**ChoiceComplement, PrepComplement and VerbComplement** A **ChoiceComplement** is a group of words that serves as a list of choices. For example, the phrase: "*{metformin and a DPP-4i, metformin and pioglitazone, metformin and an SU, metformin and an SGLT-2i}*" is a **ChoiceComplement**. It is formed by one or several **VariableStates**, which is formed by an adjective phrase or a noun phrase.

A **PrepComplement** is a preposition followed by either a **ChoiceComplement**, an **AdjPhrase**, or a **ChoiceComponent**. An example of a **PrepComplement** is "*dual therapy with {metformin and a DPP-4i, metformin and pioglitazone, metformin and an SU, metformin and an SGLT-2i}*".

A **VariableState**, a **ChoiceComplement** or a **PrepComplement** will form a **VerbComplement**.



```

VerbComplement -> VariableState | ChoiceComplement | PrepComplement
PrepComplement -> [VariableState? Prep] VariableState |
↳ ChoiceComplement
ChoiceComplement -> LBrace VariableState [Comma VariableState]+ RBrace
VariableState -> AdjPhrase | NounPhrase

```

**ConditionalClause, Condition and VerbPhraseCondition** A verb phrase is a group of words with a verb and its dependents all functioning as a predicate in the sentence. To make the parsing process easier, in our CNL we separate the verb phrase for the conditional clause and the action clause. A **VerbPhraseCondition** is a verb phrase used in a conditional clause. For example, the clause: "*rises to 58 mmol/mol (7.5%)*" is a **VerbPhraseCondition**.

A **Condition** is composed by a noun phrase and a verb phrase. It represents the condition for an action to happen. The phrase: "*HbA1c level rises to 58 mmol/mol (7.5%)*" is an example of a **Condition**.

Finally, an *if* and a **Condition** will form a **ConditionalClause** as in "*if HbA1c level rises to 58 mmol/mol (7.5%)*".

```

VerbPhraseCondition -> VerbCondition Not? VerbComplement
Condition -> NounPhrase VerbPhraseCondition
ConditionalClause -> 'if' Condition

```

**ChoiceAction** Similarly to how a **ChoiceComplement** denotes a list of choices available for things, a **ChoiceAction** is used to represent a list of choices for an action. For example, the phrase: "*{triple therapy with {metformin and a DPP-4i and an SU, metformin and pioglitazone and an SU, metformin and pioglitazone and an SGLT-2i, metformin and an SU and an SGLT-2i}, insulin-based treatment}*" shows an example of a **ChoiceAction**.

```

ChoiceAction -> LBrace PrepComplement [Comma VerbComplement]+ RBrace

```

**ToInfClause, ActionClause and VerbPhraseAction** A `ToInfClause`, which occurs only in a `VerbPhraseAction`, is a clause composed by a *to*, a `VBase` and a `VerbComplement`. An example is the clause: "*to aim for an HbA1c level of 48 mmol/mol (6.5%)*".

A `VerbPhraseAction` is a clause that always started by *shall* as in "*shall: offer standard-release metformin, support to aim for an HbA1c level of 48 mmol/mol (6.5%)*". It only occurs in an `ActionClause`.

A combination of a noun phrase and a verb phrase will form an `ActionClause`. At the moment, the noun phrase will always be written as *the doctor*. An example of an `ActionClause` is "*the doctor shall: offer standard-release metformin, support to aim for an HbA1c level of 48 mmol/mol (6.5%)*".

```
ToInfClause -> To VBase VerbComplement
ActionClause -> NounPhrase VerbPhraseAction
VerbPhraseAction -> 'shall' ':' [VerbAction ToInfClause |
↳ VerbComplement | ChoiceAction ConditionalClause?]+
```

**Advice** Finally, a complete sentence in therapy a algorithm is encapsulated as an `Advice`. It starts with a number (to mark the order), followed by a conditional clause, a comma, and an action clause.

```
Advice -> Number ConditionalClause Comma ActionClause
```

As a proof of concept, we apply the grammar rules that we have defined for extracting the main information in the sentences taken from the T2D guideline in Figure 3.1 (p. 25). We do this for the part where the patients can take metformin (the purple area with an automaton like structure). As the grammar rules are developed based on the sentence structures in this particular section of T2D guideline, the coverage is trivially 100%.

## 3.1.2 Semantic Analysis

### 3.1.2.1 Parsing sentences in CNL

After defining the grammar for our CNL, the next step will be parsing the sentences in accordance to this new grammar. First, we need to rewrite the sentences from the guidelines. Below is an example of an original guideline sentence and the rewritten one following the CNL. This sentence is taken from the T2D therapy algorithm in Figure 3.1 (p. 25).

**Original:** "If HbA1c rises to 48 mmol/mol (0.6%) in lifestyle interventions:

- Offer standard-release metformin
- Support **the person** to aim for an HbA1c level of 48 mmol/mol (0.6%)"

**CNL:** "**1** if HbA1c level rises to 48 mmol/mol (6.5%) on lifestyle interventions, **the doctor shall:** offer standard-release metformin, support to aim for an HbA1c level of 48 mmol/mol (6.5%)."

When rewriting the original sentence into CNL form, we remove the patients receiving the recommendation (highlighted in **orange red**) because it is implicitly given to them. We also add the sentence number and the phrase **the doctor shall:** (highlighted in **cyan**) to signify the beginning of the recommendation action. There are several grammar parser libraries, such as Another Tool for Language Recognition (ANTLR)<sup>3</sup>, Construction of Useful Parsers (CUP)<sup>4</sup> or ABNF Parser Generator (APG)<sup>5</sup>. For this task, we use the Modgrammar library as it is simpler and also written in the programming language that the author is most familiar with, i.e., Python. Given a list of sentences, Modgrammar will parse and generate a parse tree for each sentence which then is used in the next step.

---

<sup>3</sup><https://github.com/antlr/antlr4>

<sup>4</sup><http://www2.cs.tum.edu/projects/cup/>

<sup>5</sup><https://github.com/ldthomas/apg-7.0>

### 3.1.2.2 Case frame generation

**Case grammar theory and thematic roles** In case grammar theory (Fillmore, 1968), every word or group of words that is affected by a verb will be given a thematic role, e.g., *agent*, *theme*, *instrument* and many others. The case grammar is further developed as the FrameNet<sup>6</sup> project. In FrameNet, sentences are annotated with *semantic frames* that describe the type of event, relation or entity and the participants in it.

An example of a semantic frame is the **Health\_response**<sup>7</sup> that is realised by a Lexical Unit (LU) word such as *allergic*, *allergy*, *sensitive*, *sensitivity*, *susceptible*, and *susceptibility*. This semantic frame describes an event involving a *PROTAGONIST* that is sensitive to a *TRIGGER*, in a *BODYPART* to some *DEGREE* and *MANNER*, which are called as the Frame Element (FE). The examples below illustrate two LUs, **allergic** and **sensitive**, for the **Health\_response** semantic frame with the FEs as their syntactic dependents:

[*PROTAGONIST* Daisy Duck] is **allergic** to [*TRIGGER* cats].

[*PROTAGONIST* Minnie Mouse] is **sensitive** to [*TRIGGER* pollen].

A *thematic role* captures the semantic commonality between the verb's or predicate's arguments in the sentence. For example, *Daisy Duck* and *Minnie Mouse* in the previous examples are affected by the health trigger. Therefore, they have the **Theme** as their thematic roles. Another example is *cats* and *pollen* are both the triggers for the health response. Hence, they both have the **Actor** as their role.

Thematic Role	Definition
<b>Agent</b>	The volitional causer of an event
<b>Theme</b>	The participant most directly affected by an event
<b>Instrument</b>	An instrument used in an event
<b>Beneficiary</b>	The beneficiary of an event
<b>Source</b>	The origin of the object of a transfer event
<b>Goal</b>	The destination of an object of a transfer event

Table 3.1: Some set of commonly used thematic roles

<sup>6</sup><https://framenet.icsi.berkeley.edu/fndrupal/>

<sup>7</sup>[https://framenet2.icsi.berkeley.edu/fnReports/data/frameIndex.xml?frame=Health\\_response](https://framenet2.icsi.berkeley.edu/fnReports/data/frameIndex.xml?frame=Health_response)

Table 3.1 shows some commonly used thematic roles. The same word in two different sentences can have different roles. The following examples illustrate this phenomenon where *Pluto* serves as the **Agent** in one sentence and the **Theme** in another.

Minnie Mouse is **allergic** to [<sub>Agent</sub> Pluto].

[<sub>Theme</sub> Pluto] is **sensitive** to milk.

**Thematic roles for the CNL in clinical guidelines** We adapt the eight thematic roles defined by Carvalho (2016) in Table 2.1 (p. 12) and also add two additional roles, hence we have ten roles in total. The two additional roles are CACT for handling the nested condition and NUM for marking the sequence of the sentence. The NUM role is useful in our case because the therapies are given in a sequence, and we need to keep track of the order of their occurrences. For example, a value of 1 for NUM means it is from the first sentence in the guideline, whereas 1.1. means it is the first alternative/branch recommendation for the first sentence. Similar with Carvalho (2016), the thematic roles can be grouped into *action clause* and *conditional clause*. The explanation of each thematic role and how it is captured from the defined grammar in Section 3.1.1 (p. 26) is as follows:

- Action clause thematic roles
  1. *Action* (ACT): the action to be performed if the conditions are met. It is taken from the `VBase` or the `ToInfClause` in the `VerbPhraseAction`. For our case, these are *consider*, *review*, *offer*, *support*, and *aim*.
  2. *Agent* (AGT): the actor of ACT. It is taken from the `NounPhrase` found in the `ActionClause`. There is only one agent, i.e., the doctor.
  3. *Patient* (PAT): the entity affected by ACT and not the patient receiving the guideline. It is taken from the `VariableState` in the `VerbComplement`.
  4. *To Value* (TOV): the value given to PAT. It is taken from the `VariableState` in the `PrepositionalComplement` or `ChoiceComplement`.
  5. *Nested Condition Action* (CACT): the additional condition for ACT. For every ACT found, a list of CACT is created whose value could be empty,

i.e., an empty list. It is taken from the `ConditionalClause` found in the `VerbPhraseAction`.

- Conditional clause thematic roles
  1. *Condition Action* (CAC): the action for the condition. The values are taken from all `VerbCondition` inside the `ConditionalClause`.
  2. *Condition Patient* (CPT): the entity related to the condition. Because in the guideline there is only one entity related to the condition(s), its value is taken from the `NounPhrase` found in the `ConditionalClause`.
  3. *Condition To Value* (CTV): the new value of CPT. It is taken from the `VariableState` in the `VerbComplement` or `ChoiceComplement`, or from the `Noun` in the `PrepComplement`.
  4. *Condition Modifier* (CMD): the modifier for the condition. The value is taken from the first `Noun` in the `PrepComplement` if there are more than one.

**Constructing the case frame** After the parse trees from the sentences are generated, we traverse them one by one to construct the case frame. Figure 3.2 shows a fragment of the parse tree generated by Modgrammar for the first sentence in the T2D therapy algorithm.

The algorithm for generating the case frame is shown as the `generate_case_frame` function in Algorithm 1 (p. 36). For every element in a given parse tree, we first check if it is a `Number`. This first step is to get the sentence order and store it in the `case_frame`'s attribute. If the element is a `ConditionalClause`, another function is called to get CPT, CAC, CMD and CTV and they are stored as attributes of the `case_frame`. And lastly, if the element is an `ActionClause`, AGT, ACT, PAT, TOV, and CACT are returned from calling a function. They are then stored as the `case_frame`'s attributes. After leaving the function, the `case_frame` object contains the case frame for the particular sentence. The transformation of the `case_frame` into formal specification will be discussed further in Chapter 6 (p. 83).

Advice	VerbPhraseCondition
Number	VerbCondition
- 1	VPre3
ConditionalClause	- rises
Conj	VerbComplement
- if	PrepComplement
Condition	Prep
NounPhrase	- to
NounPhraseS	VariableState
<REPEAT>	NounPhrase
<GRAMMAR>	NounPhrases
Noun	<REPEAT>
NSing	<GRAMMAR>
- HbA1c	Noun
<GRAMMAR>	Number
Noun	- 48
NMass	<GRAMMAR>
- level	Noun
<REPEAT>	NMass
	- mmol/mol

Figure 3.2: A fragment of the parse tree for the first sentence in the T2D therapy algorithm: "If HbA1c rises to 48mmol/mol(6.5%) on lifestyle interventions: offer standard-release metformin".

---

**Algorithm 1:** Algorithm for case frame generation

---

```

1 def generate_case_frame(parse_Tree):
2     case_frame = CaseFrame()
3     for x in parse_Tree do
4         if x == Number then
5             case_frame.set_num(x)
6         else if x == ConditionalClause then
7             CPT, CAC, CMD, CTV = get_condition_elements(x)
8             case_frame.set_condition(CPT, CAC, CMD, CTV)
9         else if x == ActionClause then
10            AGT, ACT, PAT, TOV, CACT = get_action_elements(x)
11            case_frame.set_action(AGT, ACT, PAT, TOV, CACT)
12    return case_frame

```

---

Table 3.2 below shows the case frames generated from the sentence: "1 if HbA1c level

*rises to 48 mmol/mol (6.5%) on lifestyle interventions, the doctor shall: offer standard-release metformin, support to aim for an HbA1c level of 48 mmol/mol (6.5%)."*

	NUM	1
Conditions	CPT	['HbA1c level']
	CAC	['rises']
	CTV	['48 mmol/mol (6.5%)']
	CMD	['lifestyle interventions']
Actions	AGT	['the doctor']
	ACT	['offer', 'aim']
	PAT	['standard-release metformin', 'HbA1c level']
	TOV	[[], [48 mmol/mol (6.5%)]]
	CACT	[[], []]

Table 3.2: Case frames for a sentence in T2D therapy algorithm

## 3.2 Summary

In this chapter we discussed the CNL that we developed for marking the key concepts in terms of clinical processes in clinical guideline texts. As we discussed in Section 2.2.1 (p. 17), compared to Asbru language (Shahar et al., 1998; Bäumlner et al., 2006), GLARE (Terenziani et al., 2001; Giordano et al., 2006) or UML statecharts (Pérez and Porres, 2010), the CNL is very close to human natural language therefore it is not difficult to understand for practitioners when developing the guideline. Our CNL adapts the one created by Carvalho (2016) developed for formally checking software requirement texts.

The advantage of using CNL in our work can be summed up as follows:

- It is user-friendly as it is very close to our natural language
- It is suitable for restricted domain where the sentence structure variation is small
- Its development does not depend on big dataset

However, the CNL also has several shortcomings:

- It needs linguistic knowledge to develop the grammar



- It is difficult to develop a grammar that can catch all possible sentence structures. For example, our CNL requires explicit sentence number to keep the order for giving the recommendations. In contrast, some guidelines may not have ordering, hence the grammar will need to behave differently.
- It is not scalable when the dataset becomes bigger as the lexicon will also grow and may affect the existing grammar

In the next chapter, we will explain more flexible approaches to extract main components using machine learning. The transformation of annotated guidelines using the CNL will be discussed further in Chapter 6 (p. 83).

UN. — *Je le vois bien, ce do : il est là ! Mais pour ce qui est de le jouer, je suis désolé, professeur, mais je ne me rapelle plus comment on fait.*

DEUX. — *Écoutez, monsieur, vous n'êtes vraiment pas doué. Comment on fait ! On appuie dessus, tout simplement !*

Roland Dubillard, *La Leçon de piano et autres diablogues*

# 4

## Machine Learning for Clinical Guidelines

This chapter explains the Machine Learning (ML) approaches for tagging the process concepts in clinical guidelines. Concretely, we will discuss the semantic role labelling approach implemented with classic machine learning classifiers. We will also discuss the named-entity recognition and relation classification techniques implemented by neural models. Part of the work presented in this chapter has been published in Rahman and J. Bowles (2021a).

### 4.1 Semantic Role Labelling

Our approach using a Controlled Natural Language (CNL) in the previous chapter is deemed to be unscalable as either we need to rewrite (new) sentences to conform the grammar rules, or modify the grammar rules to accommodate the new sentences. Although we do not follow all the thematic roles shown in Table 3.1 (p. 33), we can also argue that it is difficult to agree with the standard set of roles as well as to determine the definition of each role formally.

For example, by definition in Table 3.1 (p. 33), an INSTRUMENT is a *thing* used in an event. However, the examples below show there are at least two possible cases for an INSTRUMENT: an *intermediary* instruments that can act as a subject (sentence 1-3), and an *enabling* instrument that cannot (sentence 4).

1. Donald Duck pricked the potatoes with a fork.
2. The fork pricked the potatoes.
3. Donald Duck ate the potatoes with a fork.
4. \*The fork ate the potatoes.

Another difficulty concerns the AGENT role. In most cases, an AGENT is animate, volitional, sentient and casual. However, most Noun Phrase (NP)s can act as an AGENT without fulfilling those criteria. *The fork* in sentence 2 above is an example of an unanimated, involuntary and insentient agent.

To tackle the limitation of the CNL that we created and its thematic roles, we build a learning model that can automatically mark the roles that arguments of a predicate can take in a sentence. This task is widely known as semantic role labelling (SRL). In SRL, we want to know "*Who* did *What* to *Whom*, and *How*, *When* and *Where*?" (Palmer et al., 2010). The sentence below shows the semantic roles for the verb *threw* that are realised by the words *John*, *a ball*, *Mary* and *in the park*.

[*AGENT* John] threw [*THEME* a ball] to [*PATIENT* Mary] [*LOCATION* in the park].

The semantic roles can be more general or abstract than what can be found in the thematic roles. Take the example of the PROTO-AGENT and PROTO-PATIENT semantic roles. These roles receive more *agent-like* and *patient-like* arguments. Therefore, if an argument demonstrates more agent-like properties, e.g., being animate, volitional and sentient involved in the event as well as causing a change of state in another entity, it is more likely that argument to be labeled as PROTO-AGENT. If it has more characteristics such as changing state affected by another entity then it is more likely to be labeled as PROTO-PATIENT. This way, classifying an NP as a PROTO-AGENT or a PROTO-PATIENT can be done by considering its characteristics between the two criteria.

Table 4.1 shows some common semantic roles with their definitions and examples. Although considerable research has been done in SRL (Kogan et al., 2005; Yaoyun Zhang et al., 2014; Medina-Moreira et al., 2017) we have not found work that deals with processes in guideline sentences. Here, we build a SRL system tailored for our clinical guidelines domain. Our SRL will not mark every argument type for a predicate as we do not need the fine grained semantic roles in our domain. Concretely, we only use the Agent and Patient from Table 4.1 and two additional roles: Action and Modifier. Furthermore, we will call the Patient role as Object. The next subsections will explain the building blocks of our SRL system.

Role	Description	Examples
Agent	Initiator of action, capable of volition	<b>The pilot</b> landed the plane as lightly as a feather.
Patient	Affected by action, undergoes change of state	David trimmed <b>his beard</b> .
Theme	Entity moving, or being "located"	Paola threw <b>the frisbee</b> .
Experiencer	Perceives action but not in control	<b>He</b> tasted the delicate flavor of the baby lettuce.
Beneficiary	For whose benefit action is performed	The Smiths rented an apartment <b>for their son</b> .
Instrument	Intermediary/means used to perform an action	He shot the wounded buffalo with <b>a rifle</b> .
Location	Place of object or action	There are some real monsters hiding <b>in the anxiety closet</b> .
Source	Starting point	We heard the rumor <b>from a friend</b> .
Goal	Ending point	Laura lectured <b>to the class</b> .

Table 4.1: A set of widely recognised Semantic Roles (Palmer et al., 2010). The words in **cyan** fulfilling the role.

### 4.1.1 Semantics in clinical guidelines

To build a learning model to mark the roles of a word or phrase in a guideline sentence, we first need to define the classes of roles that we allow in our domain. Following Diamantopoulos et al. (2017), we select several concepts to represent the process aspects of the guidelines. The design focuses on the concept of an actor doing some action(s) on some object(s) with some modifiers. However, we do not design our concepts in a hierarchy of ontology as in Figure 2.5 (p. 14) because their

aim is to check the concepts using the Web Ontology Language (OWL) whereas ours is not. For the purpose of illustrating the syntax and semantic analysis, we will use the following guideline sentence:

"All patients should have appropriate monitoring for clinically significant AEs."

Figure 4.1 shows the concepts in our SRL system. This set of concepts is the minimum that we can define to mark the labels in a guideline sentence. We can still add some more fine-grained concepts but to do so we would need 1) more data for the learning model and 2) clear annotation guideline to avoid mistakes.

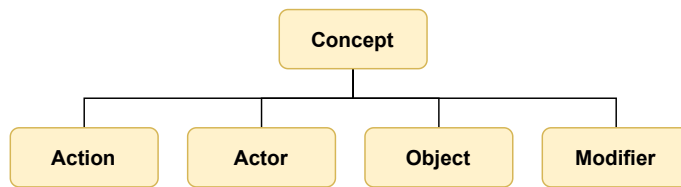


Figure 4.1: Semantic role labelling concepts that we use in this study.

Each concept is defined as follows:

- **Action:** denotes an operation performed on some **Object** by an **Actor** (if it exists). Different from Diamantopoulos et al. (2017), we also consider the ownership type as an **Action**. E.g., "*All patients should **have** appropriate monitoring for clinically significant AEs.*"
- **Actor:** refers to the explicit performer of an **Action**. In many cases, the actor is omitted in the guideline sentences.
- **Object:** denotes the entity over which an **Action** is performed. E.g., "*All patients should have appropriate **monitoring** for clinically significant AEs.*"
- **Modifier:** describes all modifiers of an **Action**, an **Actor**, or an **Object**. E.g., "*All patients should have **appropriate** monitoring for clinically significant AEs.*"

### 4.1.2 Relationship between concepts

When designing the concepts, we also need to introduce the relationships between them. These relationships define the allowed interactions between concepts. We

follow the set of relationships introduced by Diamantopoulos et al. (2017) as shown in Table 2.2 (p. 15) with some modifications. Firstly, there is no hierarchy of concepts so the `OperationType` is substituted with `Action` in `has_actor` and `actor_of`. Next, we remove the `has_property` and `is_property_of` as we do not have the `ThingType` and `Property` concepts anymore. Finally, we introduced two relationships namely `modifies` and `is_modified` for relationships from and to `Modifier`. Table 4.2 shows the adjustment for the relationship set.

Concept class	Relationship	Concept class
Action	<code>acts_on</code>	Object
Object	<code>receives_action</code>	Action
Action	<code>has_actor</code>	Actor
Actor	<code>is_actor_of</code>	Action
Modifier	<code>modifies</code>	Action, Actor, Object
Action, Actor, Object	<code>is_modified_by</code>	Modifier

Table 4.2: Relationship between concepts that we use in this study.

The relationship `acts_on` defines that an `Action` is performed on an `Object`. The inverse relation is `receives_action` that connects an `Object`. In the verb phrase: "*have appropriate monitoring*", we say *monitoring* `receives_action` from *have*.

The performer of an `Action` is defined by the `has_actor` relation to an `Actor`. Likewise, the `Actor` of an `Action` is defined by the `is_actor_of` relation. For example, *have has\_actor patients* in the phrase: "*patients should have appropriate monitoring*".

The last two relations can cover the `Action`, `Actor` and `Object` concepts as its participants. E.g., *monitoring is\_modified\_by appropriate* in the phrase: "*appropriate monitoring*".

As each pair of relations is basically an inverse of one another, we will only use three of them in our end system, namely: `acts_on`, `has_actor`, and `modifies`.

### 4.1.3 Syntactic analysis of guideline sentences

To build the features for the learning model which will be explained further in the next sections, we need to perform the syntactic analysis step. In contrast to Diaman-

topoulos et al. (2017), we use spaCy core large pipeline for English<sup>1</sup> instead of the Mate Tools for this task. spaCy gives state-of-the-art performance for Natural Language Processing (NLP) tasks and also provides configurable NLP pipelines from prototype to production.

The syntactic analysis covers several steps, namely:

- Tokenisation that splits each component in the sentence into a single token. In the sentence: "*All patients should have appropriate monitoring for clinically significant AEs.*", there will be ten tokens, namely: *All*, *patients*, *should*, *have*, *appropriate*, *monitoring*, *for*, *clinically*, *significant*, *AEs*, and ..
- Part-of-speech (POS) tagging that marks up the tokens with a particular part of speech. Following the previous example, the POS tags are as follows: *All/DT*, *patients/NNS*, *should/MD*, *have/VB*, *appropriate/JJ*, *monitoring/NN*, *for/IN*, *clinically/RB*, *significant/JJ*, *AEs/NNS*, *./..*. There are two well-known sets of POS tags: the Brown Corpus developed at Brown University<sup>2</sup> and the Penn Treebank developed at the University of Pennsylvania<sup>3</sup>. The difference between them is that the Brown Corpus has more tags than the Penn Treebank, roughly 2.4 times, which can make it too specific for some tasks. For this reason, we use the Penn Treebank tags set for our case. Appendix A (p. 135) gives the complete tag set in the two corpora.
- Lemmatisation which finds the uninflected base form of each token. A lemma is a word/token that can be inflected into several forms. E.g., *eat* as a verb is the lemma for *eat*, *eats*, *eating*, *ate*, and *eaten*. Using the previous example, the lemmas are as follows: *All/all*, *patients/patient*, *should/should*, *have/have*, *appropriate/appropriate*, *monitoring/monitor*, *for/for*, *clinically/clinically*, *significant/significant*, *AEs/ae*, *./..*
- Dependency parsing which parses the sentence based on the dependency relation of the words, i.e., every word is connected to another by a direct link. Figure 4.2 shows the dependency parse tree for the sentence: "*All patients should*

---

<sup>1</sup>[https://spacy.io/models/en#en\\_core\\_web\\_lg](https://spacy.io/models/en#en_core_web_lg)

<sup>2</sup><http://icame.uib.no/brown/bcm.html>

<sup>3</sup>[https://www.ling.upenn.edu/courses/Fall\\_2003/ling001/penn\\_treebank\\_pos.html](https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html)

*have appropriate monitoring for clinically significant AEs.*". The dependency relationship marks the link between two words. For example, the link connecting *monitoring* to *appropriate* is marked by the relation *nmod*, or  $\langle nmod \rangle \rightarrow \langle monitoring, appropriate \rangle$ , which means that *appropriate* is a noun modifier for *monitoring*.

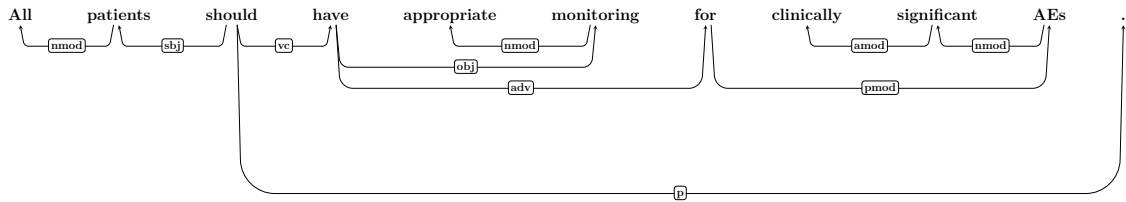


Figure 4.2: Dependency parse tree for the sentence: "All patients should have appropriate monitoring for clinically significant AEs."

#### 4.1.4 Semantic analysis of guideline sentences

Similarly to the syntactic analysis, we adapt the approach of Diamantopoulos et al. (2017) in our semantic analysis step. This step resembles the semantic role labelling pipeline defined by Björkelund, Hafdell et al. (2009), namely the *predicate identification*, *predicate disambiguation*, *argument identification*, and *argument classification*. In relation to our problem domain, each step in the pipeline deals with one particular task as follows:

1. identifying words that are either **Action** or **Object**, which corresponds to the *predicate identification*. The reasoning behind choosing these two concepts is because they define the relationships to others. For example, by knowing if a word is an **Action** or an **Object**, we can further find the rest of the concepts through the relationships *acts\_on*, *has\_actor*, and *modifies*.
2. classifying words identified in step 1 to their correct concept, similar to the *predicate disambiguation*. For every verb and noun that can be either an **Action** or an **Object**, this step classifies them into the actual concept, e.g., *have*/**Action**, *monitoring*/**Object**.
3. identifying words that are related to the instances in step 1, which corresponds to the *argument identification*. The instances that we are looking for in this step are the **Actor** of an **Action** and the **Modifier** related to any other concepts.



For example, this step will recognise *patients* as an Actor and *appropriate* as a Modifier.

4. classifying the relationship holds between a pair of instances from step 1 and step 3, which corresponds to the *argument classification*. This step will generate a pair of words and its corresponding relation such as  $\langle \textit{patients}, \textit{have} \rangle \rightarrow \langle \text{Actor}, \text{Action} \rangle$  and  $\langle \textit{appropriate}, \textit{monitoring} \rangle \rightarrow \langle \text{Modifier}, \text{Object} \rangle$ .

### 4.1.5 Features

To do the semantic analysis, we build one learning model for every task mentioned in the previous section. This means that we need to have a set of features for every learning model as it is more likely that one set of features for a task will not perform as well when used for different tasks. We based our feature sets on the intersection between the approach used by Diamantopoulos et al. (2017) and Gildea and Jurafsky (2002) for SRL.

Most of the basic features have been implemented with spaCy. Furthermore, our additional features can be derived from the ones that have been provided. These features are as follows:

1. *affected word form*, which is the original word, capitalised or not, in the sentence;
2. *affected word lemmata* taken from the lemmatisation step of the syntactic analysis;
3. *word part-of-speech* taken from the part-of-speech tagging of the syntactic analysis;
4. *relation to parent* taken from the relation of dependency parsing of the syntactic analysis;
5. *parent part-of-speech* derived from the dependency parsing of the syntactic analysis;
6. *child words*, similar to the *affected word form*, but for all children of the current word, are derived from the dependency parsing of the syntactic analysis;

7. *child part-of-speech*, similar to the *parent part-of-speech*, but applied to all children of current word;
8. *dependency between words*, i.e., the words in dependency relations between the action and its object, the action and its actor, or the modifier and its action/actor/object;
9. *position of affected words*, i.e., before or after the predicate;
10. *word vector representation*, denoting the word embedding or the numerical representation for every word.

As the learning process can only use numerical representation for the features, we need to transform every non-number feature into numbers. We use two different approaches to achieve this, namely:

1. *tf-idf* to vectorise features 1 and 2. *tf-idf*, short for *term frequency-inverse document frequency*, computes the importance of every word in that particular feature and rerank them accordingly. In other words, *tf-idf* will give higher ranks for rare words and put common words in lower ranks. In our corpus, every sentence serves as a document.
2. *one-hot encoding* to vectorise features 3 to 9. A one-hot representation will generate a 1 by  $N$  vector for every categorical value, where  $N$  is the size of the category, i.e., the number of unique values in the category. This encoding makes sure that there is only a single value 1 in a cell where that particular value appears and the remaining cells are all 0.

#### 4.1.6 Word embedding

Vectorising categorical features into numerical form, especially using one-hot encoding, will inevitably generate a very sparse matrix where most entries are zero. This matrix is also heavily affected by the size of the vocabulary, i.e., the bigger the vocabulary the sparser the matrix will be.

To compensate this phenomenon, we add a word embedding or a word vector representation (feature 10) which is a dense matrix that captures a semantic representation of the words in the text. This feature is also used to give more context generalisa-

tion since some features in our current task are very domain specific, e.g., the word lemmata feature.

A word embedding of dimension  $d$  can also capture the similarity of words such that words with closer meaning will have close or similar vector representations. The dimension  $d$  is normally set between 50 and 300, i.e., for a word  $w$ , there is a dense  $d$ -dimension vector representing its meaning. The specific word embedding dimensions that we use in our experiments are further explained in Section 5.2.2 (p. 70).

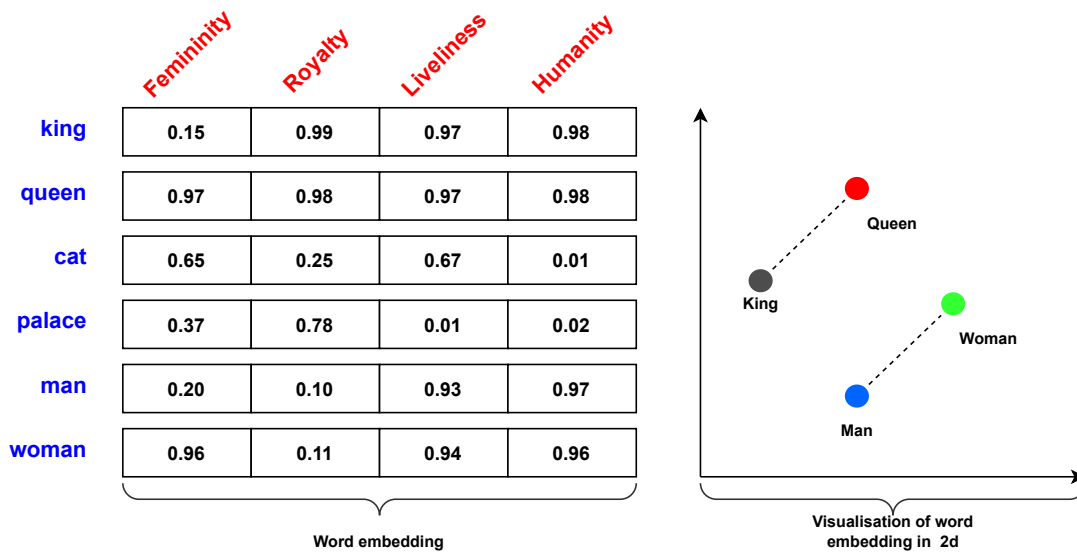


Figure 4.3: Word embedding capturing the semantic similarity of words

Figure 4.3 illustrates a 4-dimensional word embedding for several words. Note that in reality these dimensions do not have real meaning, i.e., they only represent latent relations between the words and are open to many possible interpretations. Here we illustrate them with some artificial semantic concepts (highlighted in red) so the similarity between words is comprehensible. Using the cosine similarity<sup>4</sup> of two vectors, we can infer that the word *king* is closer to *queen* than to *cat*. The diagram on the right side shows how words are related to each other in a 2 dimensional space. Using word embedding, we can do arithmetic operation to get that  $(king - man + woman) \approx queen$ .

Initially, we used the word2vec (Mikolov, Chen et al., 2013) to generate the word

<sup>4</sup>[https://en.wikipedia.org/wiki/Cosine\\_similarity](https://en.wikipedia.org/wiki/Cosine_similarity)

vector representation specific to our problem domain and ran several experiments. However, the result obtained was not promising. We suspect this is due to the small size of our corpus. In addition, we believe that using word2vec trained on our corpus will make the word embedding very specialised and less generic. The top results for the experiments using word2vec will be mentioned in Section 5.2.2 (p. 70) and Section 5.2.4 (p. 76).

For these reasons, we then explored the use of pre-trained word embeddings as done by Diamantopoulos et al. (2017). However, slightly different from their approach which adapting the word vector representation from curriculum learning coined by Bengio et al. (2009), we utilise pre-trained word embeddings from two different sources: Global Vector (GloVe) (Pennington et al., 2014)<sup>5</sup> and fastText (Mikolov, Grave et al., 2018)<sup>6</sup>.

For the GloVe word embedding, there are several available choices. They differ in terms of the matrix dimension as well as the data source that they are trained on:

- Wikipedia 6B uncased tokens: 50d, 100d, 200d and 300d vectors
- Common Crawl 42B uncased tokens : 300d vectors
- Common Crawl 840B cased tokens: 300d vectors
- Twitter 27B uncased tokens: 25d, 50d, 100d and 200d vectors

We did a set of preliminary experiments on all four tasks using all the possible choices and found that the 300 dimensions' performance is better than the 50 and 100 dimensions, while the 200 dimension is slightly worse. We also found that the result using Common Crawl and Twitter corpus is not better than using Wikipedia. We suspect this is because there are many non-standard words in the Common Crawl and Twitter corpus, which will affect the overall embedding values for all words.

Similar to GloVe, fastText also has several choices for its word embedding. They are as follows:

- Wikipedia news 16B tokens: 300d vectors

---

<sup>5</sup><https://nlp.stanford.edu/projects/glove/>

<sup>6</sup><https://fasttext.cc/>

- Wikipedia news 16B tokens with subword information: 300d vectors
- Common Crawl 600B tokens: 300d vectors
- Common Crawl 600B tokens with subword information: 300d vectors

In the same manner with GloVe, to see the effect of using fastText word embeddings, we conducted experiments on all four tasks. However, we found that the general performance is still lower than using GloVe. Furthermore, the subword information provided by fastText does not contribute to a better output. Here we suspect that fastText word embeddings are too generic that they miss to capture intrinsic representation for specific words used in clinical text. For this reason, we do not include fastText word embeddings in our final evaluation. The top results for the experiments using fastText are mentioned in Section 5.2.2 (p. 70) and Section 5.2.4 (p. 76).

Table 4.3 shows the features and their usage in each semantic analysis step. We only include here the combination of features that give the best performance.

	Action and Object		Related concepts	
	Identification	Classification	Identification	Classification
word form	✓	✓	✓	✓
word lemmata	✓	✗	✗	✗
word POS	✓	✗	✓	✓
dependency relation	✓	✗	✓	✓
parent POS	✓	✓	✗	✗
child words	✓	✗	✗	✗
child POS	✓	✗	✗	✗
dependency words	✗	✗	✓	✓
position	✗	✗	✓	✓
word embedding	✓	✓	✓	✓

Table 4.3: Feature sets and their usage for SRL task. The ✓ denotes the features used in a particular task while the ✗ otherwise.

### 4.1.7 Learning

To get the best learning model for our semantic role labelling process, we run our dataset against several classifiers. To achieve this, we annotate our guideline sentences following the concepts needed for the particular step. For example, in the first and second step, we only annotated words in the sentences as either **Action** or

**Object**. Then we give the label for those words as either **1** (for potential **Action** or **Object**) or **0** (for others). For the second step, the classifier will learn to distinguish the words recognised in step 1 as either **1** (for **Action**) or **0** (for **Object**).

After comparing several classifiers such as decision tree, random forest, logistic regression, naive Bayes and perceptron, we choose perceptron (Rosenblatt, 1957) as our learning algorithm as it gives us the best result. We use the free perceptron library from scikit-learn<sup>7</sup>. In perceptron, during the training step at time step  $t$ , for every input  $\mathbf{x}_j$  and expected output  $d_j$  in the training set, the algorithm will calculate the predicted output  $y_j(t)$  using the weight matrix  $\mathbf{w}(t)$  and activation function  $f$  as in Equation 4.1. At the end of every training iteration, the weight matrix for the next time step  $\mathbf{w}(t+1)$  is updated following Equation 4.2 where  $w_i$  is the weight for feature  $i$ ,  $x_{j,i}$  is the  $i$ th feature value of  $j$ th training data, and  $\eta$  is the learning rate.

$$y_j(t) = f[\mathbf{w}(t) \cdot \mathbf{x}_j] \quad (4.1)$$

$$w_i(t+1) = w_i(t) + \eta \cdot (d_j - y_j(t))x_{j,i} \quad (4.2)$$

The learning process will stop when it converges, i.e., the value of  $|d_j - y_j| \leq \epsilon$  where  $\epsilon$  is a very small threshold value. Otherwise, it will stop if it reaches the maximum number of learning iterations.

## 4.2 NER & Relation Classification

In addition to using a CNL shown in Chapter 3 (p. 23) and SRL (discussed in the previous section), we also investigated another approach to achieve our goal for tagging the key concepts in the sentences. In particular, the semantic or ontology concepts labelling can also be done using Named-Entity Recognition (NER). We adapt the relation classification technique to link two entities in a relationship. The models discussed in this section have some resemblance with the work of Grivas et al.

---

<sup>7</sup>[https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.Perceptron.html](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html)

(2020). However, as their aim is to mark *findings* (such as tumour), *modifiers* (such as recent, acute, cortical) and *negation*, we have different entity and relation labels sets.

### 4.2.1 NER in biomedical texts

NER is an NLP task for detecting the entity in the text that can be referred to with a proper name such as a person, a location, an organisation, or even a thing that is not a proper entity such as date, time, or price (Jurafsky and Martin, 2009).

For example, in the sentence: "[*Bill Gates*]<sub>PERSON</sub>, the co-founder of [*Microsoft*]<sub>ORG</sub>, lives in [*the USA*]<sub>GPE</sub>.", there are three named-entities, namely: **PERSON** for *Bill Gates*, **ORG** (stands for organisation) for *Microsoft*, and **GPE** (stands for geopolitical entity) for *the USA*. From this example, we can also see that a named-entity can span multiple words, as in *Bill Gates* and *the USA*.

There are several prominent language processing libraries that have provided NER as one of their functionalities, for example spaCy, Stanza<sup>8</sup> and Flair<sup>9</sup>. Some of these libraries have an additional tool to recognise named-entity in biomedical texts such as Stanza (Yuhao Zhang et al., 2020) and Flair (Akbik et al., 2018). These libraries have used several biomedical corpora for training their NER modules as shown in Table 4.4.

	Corpus	Entity Types
Stanza	AnatEM	Anatomy
	BC5CDR	Chemical, Disease
	Linnaeus	Species
	NCBI-Disease	Disease
	i2b2-2010	Problem, Test, Treatment
Flair	BioCreative II GM	Gene
	CHEBI	Chemical, Gene, Species
	JNLPBA	Cell line, Gene
	miRNA	Disease, Gene, Species
	S800	Species

Table 4.4: Some Stanza and Flair biomedical NER corpora

Despite the existence of tools to mark the named-entity in biomedical texts, we

<sup>8</sup><https://stanfordnlp.github.io/stanza/>

<sup>9</sup><https://github.com/flairNLP/flair>

cannot just use them as they have been trained for different purposes. As discussed in previous sections, the nature of our task is to mark the process/event in clinical guidelines which involve an actor, action, object and modifier of a process, which is different to finding a disease, chemical, gene or species in the text. We found that the closest NER tags to our problem are the i2b2-2010 corpus tags from Stanza. As the i2b2-2010 corpus is built for marking the concepts, assertions, and relations in clinical text, it has some similarities to our domain. Nevertheless, since the tags in i2b2-2010 are completely different from what we need for our problem (problem, test, and treatment vs. action, actor, object and modifier), we still need to train our own model to suit our particular problem.

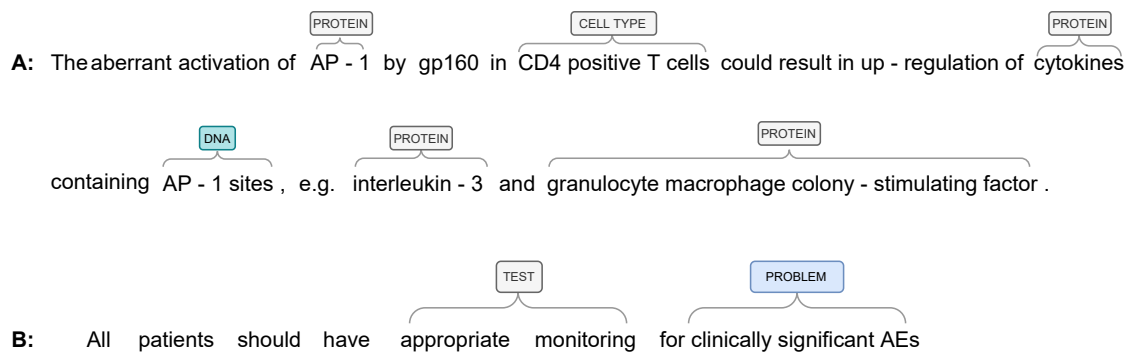


Figure 4.4: Two sentences marked with Stanza i2b2-2010 tags

Figure 4.4 shows two sentences, one taken from Stanza (sentence A) and another from our corpus (sentence B). It can be seen in sentence A that Stanza manages to recognise protein, cell type, and DNA in a word or group of words. As in our original sentence B, corpus 12b2-2010 helps Stanza to recognise what a test and a problem in a clinical sentence are.

### 4.2.2 Baseline model for NER

In NER, the named-entities are usually marked using the BIO format (short for beginning-inside-outside). The beginning of an entity type is marked with a B-*prefix* tag. I-*prefix* tag marks every token inside an entity type whereas an O tag is used for tokens that do not belong to any entity. For example, the prefix GPE tag *in New York* will be marked as *in<sub>O</sub> New<sub>B-GPE</sub> York<sub>I-GPE</sub>*. Using this format, the annotation model can learn to tag a single word as well as multiple words.



Figure 4.5 shows our earlier sentence marked in the BIO format. For this task, we use the concepts from the semantic role labelling as entities. As there are two tags for each entity, i.e., the *B-tag* and the *I-tag*, our label size becomes 9 (from  $2n + 1$ ) namely *B-action*, *I-action*, *B-actor*, *I-actor*, *B-object*, *I-object*, *B-modifier*, *I-modifier*, and *O*.

All patients should have appropriate monitoring for clinically significant AEs  
B-actor B-action B-modifier B-object

Figure 4.5: A sentence marked with the BIO format

To implement our NER, we build a neural network model using Long Short-Term Memory (LSTM; Hochreiter and Schmidhuber, 1997). LSTM is different from a typical Recurrent Neural Network (RNN): while the latter constitutes a class of neural networks to analyse sequence and time series data, the former can in addition capture long-term dependencies in the data. An LSTM unit/cell is made up of an input gate, an output gate, and a forget gate. This mechanism makes an LSTM cell able to learn an important input, keep it as long as it is deemed important, and extract it when it is required.

Figure 4.6 shows our NER model using LSTM. We use bidirectional LSTM to recognise the pattern in the sentence in both forward and backward directions. For every sentence, the model has two inputs: the part-of-speech of the words in the sentence as well as the word vector representation from GloVe. Each input is passed to a bidirectional LSTM and the outputs are concatenated into a three dense neural layer.

Although Figure 4.6 only shows that the input is represented by word embeddings and part-of-speech, we also ran many experiments using the combination of all possible feature sets as in Table 4.3 (p. 50). We also conducted experiments to see the effect of using different word embeddings such as word2vec and fastText. The experiments conducted and their results will be discussed further in Section 5.2.2 (p. 70).

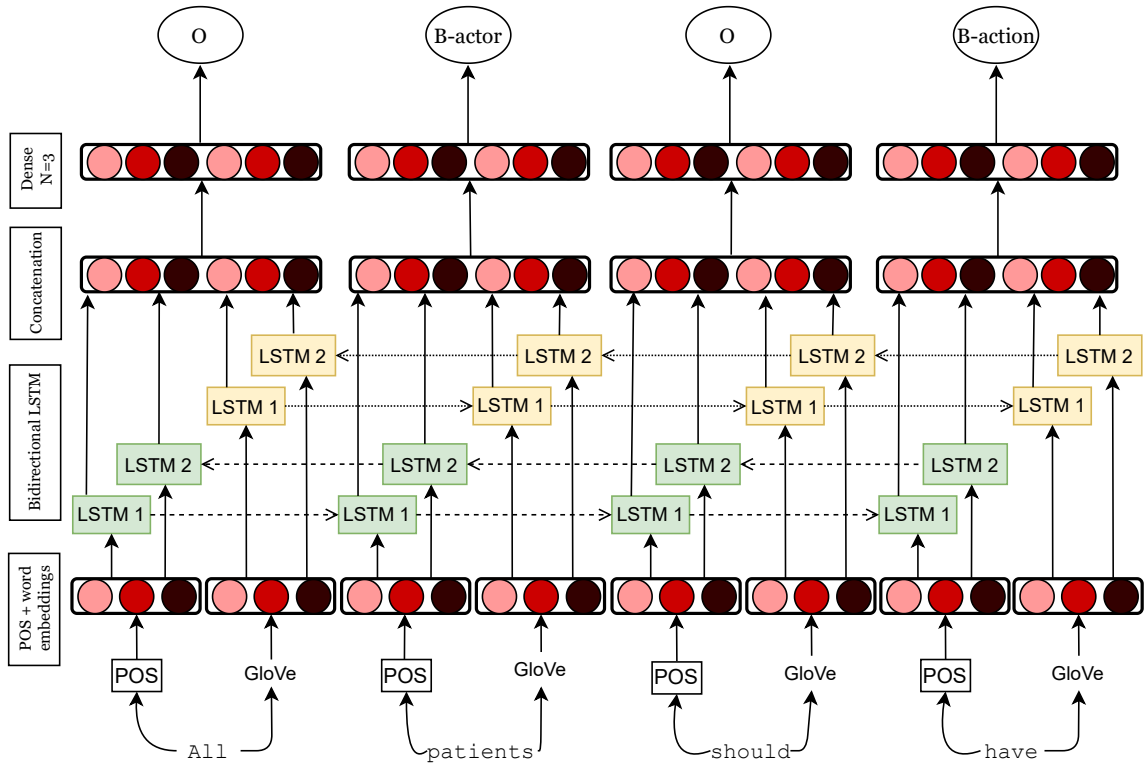


Figure 4.6: Named-entity recognition model implemented using BiLSTM

### 4.2.3 Baseline model for relation classification

After getting the labeled entities in the text, the next task is to connect them into a meaningful relation. This task is called *relation classification* and is essential for further text analysis, such as for information extraction, question answering and knowledge base population (Nguyen and Grishman, 2015).

Concretely, given two nominals in a sentence, relation classification will classify the semantic relation between them (Hendrickx et al., 2010). In other words, given a sentence  $S$  with the annotated pairs of nominals  $e_1$  and  $e_2$ , the aim is to identify the relation between  $e_1$  and  $e_2$ . For example, in the sentence: "*The cup contained tea from dried ginseng*", the relation between *tea* and *ginseng* is ENTITY-ORIGIN.

Despite the original setting being only between two nominals, we adapt this approach to allow the identification of the relationship that we have from previous approaches as shown in Table 2.2 (p. 15). For our domain, we have four bidirectional relations: **acts-on**, **modifier**, **owner**, and **receiver** and four unidirectional relations: **actor-of**, **choice**, **joint**, and **other**. In total, we have 12 labels for our model to learn.

Table 4.5 (p. 57) shows these relationships and their example. For example, in the sentence: "*For people who cannot <e1> tolerate </e1> <e2> aminosalicylates </e2>, consider a time-limited course of a topical or an oral corticosteroid.*", *tolerate* is the first entity whereas *aminosalicylates* is the second entity in the relationship *acts-on*.

Our relation classification model is adapted from the work of Zeng et al. (2014) that used a Convolutional Neural Network (CNN) to learn the features at a sentence level. These features are combined with lexical level features in order to predict the relationship between two marked entities. Instead of using the 50 dimensions Google News pre-trained word embedding as in (Zeng et al., 2014), we use word embedding from GloVe and fastText to see the effect of the learning process as discussed in Section 4.1.6. Furthermore, we do not add the WordNet features into our model.

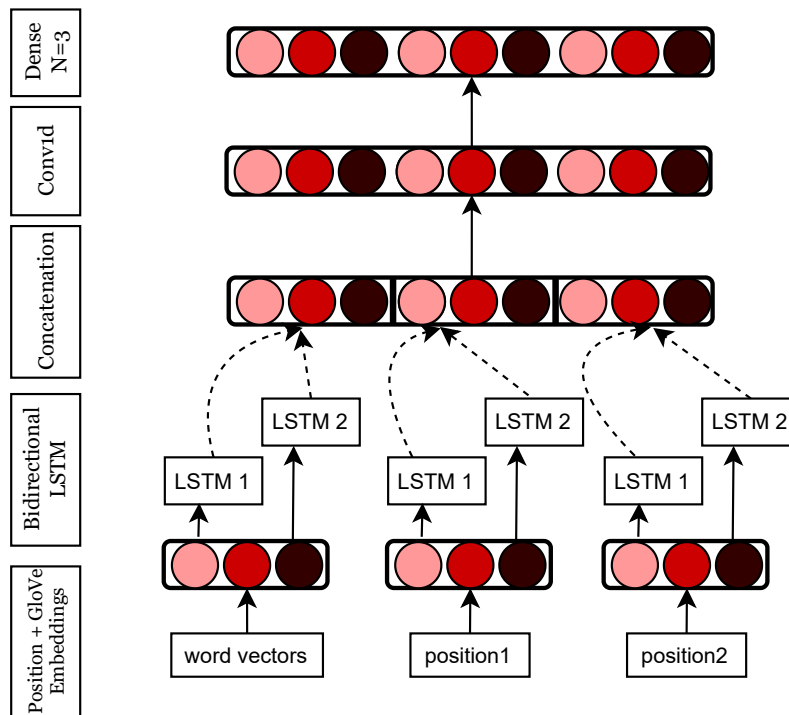


Figure 4.7: Relation classification model implemented using BiLSTM

Figure 4.7 (p. 56) shows our relation classification model for a single word in a sentence. As the inputs, a word in a sentence is represented by its word embedding from GloVe or fastText and its relative positions to the first and the second entities. These positions mark the distance of this particular word to the entities for which the model is trying to identify a relationship. The position value 0 indicates that this particular word is one of these two entities.

Relationship	Sentence
acts-on(e1,e2)	For people who cannot tolerate aminosalicylates, consider a time-limited course of a topical or an oral corticosteroid.
acts-on(e2,e1)	Vaccination in patients with AIIRD should ideally be administered during stable disease.
modifier(e1,e2)	For people who cannot tolerate aminosalicylates, consider a time-limited course of a topical or an oral corticosteroid.
modifier(e2,e1)	Request the oestrogen receptor, progesterone receptor and human epidermal growth receptor 2 status of all invasive breast cancers simultaneously at the time of initial histopathological diagnosis.
owner(e1,e2)	Maintain blood pressure below 130/80 mmHg for people with advanced pancreatic cancer.
owner(e2,e1)	If dipstick screening is positive, use albumin:creatinine ratio or protein:creatinine ratio to quantify proteinuria in pregnant women.
receiver(e1,e2)	For people who cannot tolerate aminosalicylates, consider a time-limited course of a topical or an oral corticosteroid.
receiver(e2,e1)	Maintain blood pressure below 130/80 mmHg for people with advanced pancreatic cancer.
actor-of(e1,e2)	For people who cannot tolerate aminosalicylates, consider a time-limited course of a topical or an oral corticosteroid.
choice	For people who cannot tolerate aminosalicylates, consider a time-limited course of a topical or an oral corticosteroid.
joint	Request the oestrogen receptor, progesterone receptor and human epidermal growth receptor 2 status of all invasive breast cancers simultaneously at the time of initial histopathological diagnosis.
other	For people who cannot tolerate aminosalicylates, consider a time-limited course of a topical or an oral corticosteroid.

Table 4.5: Relationships and their example. To declutter the table and for an aesthetic reason, we replace the  $\langle e1 \rangle \langle /e1 \rangle$  and  $\langle e2 \rangle \langle /e2 \rangle$  notations to mark the first and the second entities by highlighting them in yellow and green respectively.

word	All	...	have	...	monitoring	...
word vectors	[..., ...]	...	[..., ...]	...	[..., ...]	...
position1	-3	...	0	...	2	...
position2	-5	...	-2	...	0	...

Table 4.6: Input example for relation classification model

Table 4.6 shows an example of inputs for recognising the relationship between *have* and *monitoring* in the sentence: "*All patients should <e1> have </e1> appropriate <e2> monitoring </e2> for clinically significant AEs.*".

The model learns from these inputs using bidirectional LSTM and combines the outputs to be used in the subsequent layers. The convolutional layer will learn local features in the sentences. The last three layers are dense layers where the final one is responsible for identifying the probability of the 12 labels.

## 4.2.4 Fine-tuning state-of-the-art models

In addition to creating our neural baseline model, we also conducted several experiments using some existing pre-trained models. Since good generic NER models that have been trained with huge datasets are available, it is intuitive to build our own model on top of them. To make these models more relevant to our own domain specific task, we adjust the model's weights by training them using our own dataset. This approach is called *fine-tuning the models*.

### 4.2.4.1 Fine-tuning spaCy model for NER

There are two notable models that we will discuss, namely spaCy and BERT. As we briefly explained in Section 4.2.1 (p. 52), spaCy provides NER among many other functionalities in its library. spaCy allows us to train its model so it can be more specific to our problem. For this task, we need to slightly change the input format as spaCy uses the Begin-Inside-Last-Unit-Outside (BILUO) format instead of the BIO format. The BILUO format differentiates between the last entity of a tag in compound words (using the *L-tag*) as well as a single word tag (using the *U-tag*). Unlike our baseline model whose inputs are the word embedding and the part-of-speech, fine-tuning the spaCy model only uses the word tokens as spaCy has its

internal modules such as POS tagger and dependency parser. Figure 4.8 illustrates the difference between the BIO and the BILUO formats.

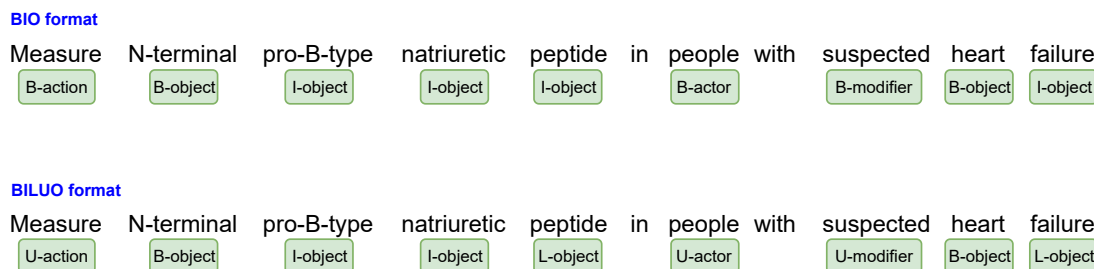


Figure 4.8: A sentence marked with the BIO and the BILUO formats

For fine-tuning spaCy, we used the large spaCy model `en_core_web_lg` whose NER model has been pretrained using OntoNotes<sup>10</sup> as its dataset and GloVe word vectors trained on Common Crawl.

#### 4.2.4.2 Fine-tuning BERT model

**The limitation of pre-trained word embeddings** Despite the benefit of using pre-trained word embeddings such as GloVe or fastText to capture the latent semantic of words, it also has some drawbacks as pre-trained word embeddings are static and context independent. This means that they do not have different representations for different meanings of the same word. In other words, they cannot disambiguate the word sense based on the context it is used in. For example, GloVe has only one entry for the word *right* even though its meaning is totally unrelated in "*She is right*" and "*Turn right*". Therefore, dynamic and context sensitive word embeddings are more favourable to capture higher semantics phenomena and to overcome the oversimplification in the representation.

**BERT as a dynamic and context sensitive word embedding** BERT, short for Bidirectional Encoder Representations from Transformers (Devlin et al., 2018), is a pre-trained neural model based on the Transformer architecture (Vaswani et al., 2017), both developed by Google. Similar to RNN and LSTM, Transformer is also used to handle sequential data like texts. However, instead of using a recurrent network, Transformer uses a self attention mechanism to obtain the context of the

<sup>10</sup><https://catalog.ldc.upenn.edu/LDC2013T19>

words in a text, i.e., how one word in the text is related to every other word in the sentence *including itself*, and to capture these long-term dependencies. The bidirectional nature added in BERT allows it to capture the context of a word in forward and backward directions which gives performance improvement.

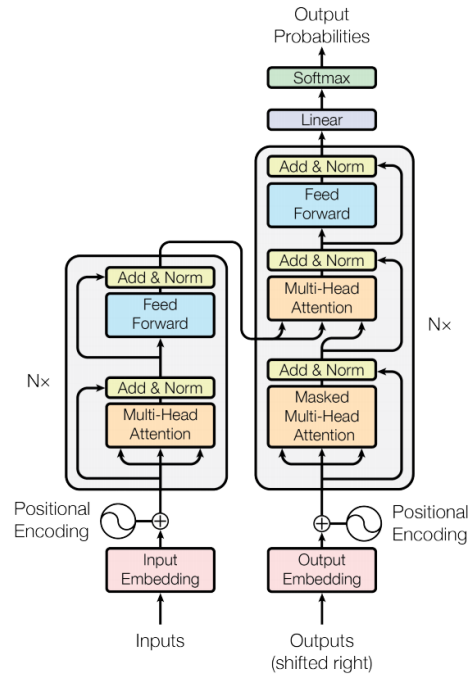


Figure 4.9: The Transformer architecture (Vaswani et al., 2017)

**Encoder, decoder and Transformer** Figure 4.9 shows the architecture of Transformer. An input sentence is fed as a sequence of tokens to the encoder (the left part) and the output from the encoder is fed to the decoder (the right part). The original architecture has six layers of encoders ( $N \times = 6$ ) and six layers of decoders. The encoder consists of two layers: self multi-head attention and feed forward. The crux of the transformer is self-attention: a mechanism for an encoder to attend to the context of a specific token while reading the input sequence. For every token, the output of this layer will be a matrix showing its relationship to each token in the text, including itself. In the sentence: "*I have a cat. It sat on a mat."*, the token *It* will have a higher value in its relationship with the token *cat* than the token *mat*.

The decoder layer is similar to the encoder with additional masked self multi-head attention layer to perform a cross-attention mechanism over the output of the encoder. It is worth noting that Transformer only uses the standard feed forward

neural network. Nevertheless, with a self-attention mechanism, it still outperforms RNN and LSTM due to its ability to understand the context of a token even to the most distant part of the sentence, hence reducing the locality bias.

Another visible difference between RNN/LSTM and Transformer is in the way they process the input. Unlike RNN and LSTM, which process the input text sequentially token by token so the network understands the whole sentence, Transformer processes in parallel all tokens in the text to its network. This parallelisation can help decrease the training time as well as the time required for learning the long-term dependency in the text. However, the tokens' position in the text is a crucial factor to understand their meanings. To keep this information, Transformer uses positional encoding to mark the token order in the text. This positional encoding is combined with the input embedding in the encoder and output embedding in the decoder.

Although Transformer based architecture like BERT has become the state-of-the-art model for capturing and extracting language representation, it is also worth to note that this is not always the case. In some scenarios, traditional neural model such as LSTM can even outperform BERT. For example, experiments conducted by Makarenkov and Rokach (2020) show that for the proper word choice task<sup>11</sup> and binary classification for political perspective in news articles, LSTM, BiLSTM and logistic regression can yield better performance than BERT. Another experiment carried out by Ezen-Can (2020) shows that BiLSTM achieves higher results than a BERT model for intent classification task using a chatbot corpus. The similarity between these two findings are they both have small datasets. Furthermore, training BERT model needs powerful computing machine such as Graphics Processing Unit (GPU) for the training to run faster. We discuss this further by comparing the running time of our experiments using Central Processing Unit (CPU) and GPU in Section 5.2.2 (p. 70) and Section 5.2.4 (p. 76).

**BERT variants** BERT uses the same architecture as Transformer minus the decoder layer but with bidirectional functionality to the self-attention mechanism. Since it is an encoder representation, it generates the embedding for each word in the

---

<sup>11</sup>The proper word choice task aims to replace a *target* word in a sentence *s* based on the context of the *target* in *s* (Makarenkov and Rokach, 2020).



text based on its respective context. In the last couple of years, BERT has shown better performance compared to other models in many natural language understanding (NLU) tasks, such as natural language inference (NLI), sentiment analysis, question answering, and NER.

Since its inception, BERT has many variants that are developed either due to the difference in the size of the architecture, the domain that they are trained on, or the kind of task that needs to be addressed. For our fine-tuning BERT model, we consider several BERT variants to compare their performance. Table 4.7 shows the original Transformer architecture detail and a list of BERT models that we fine-tune. For the relation classification task, we only fine-tune the top two models that give best performance in NER task to save time and resources.

Model name	Detail
<b>Transformer</b>	L=6, E=512, H=512, A=8, Parameters=3M
<b>BERT<sub>BASE</sub></b>	L=12, E=768, H=768, A=12, Parameters=110M
<b>ALBERT<sub>BASE</sub></b>	L=12, E=128, H=768, A=12, Parameters=11M
<b>RoBERTa<sub>BASE</sub></b>	L=12, E=768, H=768, A=12, Parameters=110M
<b>Bio_ClinicalBERT</b>	L=12, E=768, H=768, A=12, Parameters=110M

Table 4.7: BERT models for fine-tuning (including the original Transformer). L is the number of layers, E is the size of the input word embeddings, H is the dimension of the hidden unit size, and A denotes the number of attention heads.

**BERT<sub>BASE</sub>** uses the encoder architecture from Transformer with different size in input embedding, layers, hidden units and heads. For this task, we use the basic `bert-based-cased` that has been train on cased English texts. We chose this model, rather than `bert-based-uncased`, as we consider the occurrence of terms in uppercase or lowercase matters in clinical texts as it is common in NER. Besides this, there are also other BERT models which differ in the size of the layers, the uncased/lower case treatment, multilingual BERT, etc.

**ALBERT<sub>BASE</sub>** (A Lite BERT) reduces two parameters in BERT to increase the training speed and to lower the memory consumption of BERT (Lan et al., 2019). The first one is the reduction of the input embedding to the magnitude of 6 (Table 4.7). Another difference is the cross-layer parameter sharing, i.e., instead of having a separ-

ate parameters for each encoder layers, ALBERT uses only one encoder’s parameter weights that is shared to all 12 layers.

**RoBERTa<sub>BASE</sub>** improves BERT’s training procedure by modifying some hyper-parameters (Y. Liu et al., 2019). It also uses much larger batch-training sizes, learning rates, and training corpus for a total of 160 GB of text. Due to this improvement, RoBERTa outperforms BERT’s accuracy in many NLP tasks.

**BIO\_ClinicalBERT** was created by Alsentzer et al. (2019) based on BERT and BioBERT (Lee et al., 2020). BioBERT itself is built from BERT trained on biomedical domain corpora, namely PubMed abstracts<sup>12</sup> and PMC full-text articles<sup>13</sup>. BIO\_ClinicalBERT added more training data from 2 million notes in the MIMIC-III Clinical Database<sup>14</sup>. This makes BIO\_ClinicalBERT very good at NLP tasks for the clinical domain.

### 4.3 Summary

In this chapter, we discussed our foundations for applying machine learning techniques to capture the key information in clinical guidelines in contrast to the approach we discussed in Chapter 3 (p. 23). We explained the semantic role labelling approach and its feature set. We also discussed the named-entity recognition approach and the relation classification between two entities using neural models of our own or fine-tuned some state-of-the-art models.

Details on the experiments we conducted are discussed separately in the next chapter. This includes all experiments’ setup and results for the machine learning approaches that we have discussed in this chapter.

---

<sup>12</sup><https://pubmed.ncbi.nlm.nih.gov/>

<sup>13</sup><https://www.ncbi.nlm.nih.gov/pmc/>

<sup>14</sup><https://physionet.org/content/mimiciii/1.4/>

*Les grandes personnes aiment les chiffres.*

Antoine de Saint-Exupéry, *Le Petit Prince*

# 5

## Experiments and Evaluation

In this chapter, we describe the experiment and the performance of our machine learning model to label and to link the key components in clinical guideline sentences. First, we describe how we set up our experiment environment by discussing the nature of our dataset. We then explain the performance of running several experiments and analyse lessons learnt. Finally, we will discuss how we may improve the learning models for our purposes. Part of the work presented in this chapter has been published in Rahman and J. Bowles (2021a) and Rahman and J. Bowles (2021b).

### 5.1 Dataset Analysis

Our dataset has a total of 379 sentences all written in English: 216 of them are taken from The National Institute for Health and Care Excellence (NICE)<sup>1</sup>, 98 are from The Scottish Intercollegiate Guidelines Network (SIGN)<sup>2</sup>, and 65 are from The Annals of the Rheumatic Disease (ARD)<sup>3</sup>. These sentences are gathered from guidelines for various diseases to capture the nature of the sentences in a clinical guideline setting.

---

<sup>1</sup><https://www.nice.org.uk/>

<sup>2</sup><https://www.sign.ac.uk/>

<sup>3</sup><https://ard.bmj.com/>

Table 5.1 shows some of the diseases and their sources used in this study. In general, we found that the structure of the sentences from Scottish Intercollegiate Guidelines Network (SIGN) is mostly in passive voice when compared to the ones from Scottish Intercollegiate Guidelines Network (SIGN) and Scottish Intercollegiate Guidelines Network (SIGN). As the aim of this study is to formally check actions in clinical guidelines, we only picked sentences that have explicit activity and ignored narrative sentences.

There are 7967 tokens and 1414 types (unique tokens) in our dataset distribution. The shortest sentence has 9 tokens whereas the longest has 66. Figure 5.1 shows the sentence distribution over token counts. The histogram shows that the data distribution is slightly right skewed. The token counts distribution has a mean of 21.02, median of 19, and standard deviation of 8.26.

Source	Code	Disease type
NICE	NG28	Type 2 diabetes
	NG129	Chron's disease
	NG118	Renal and ureteric stones
	NG122	Lung cancer
	NG128	Stroke and transient ischaemic
	CG182	Chronic kidney disease
SIGN	SIGN152	Cardiac arrhythmias
	SIGN158	Asthma
	SIGN135	Ovarian cancer
	SIGN133	Hepatitis C
	SIGN134	Breast cancer
ARD	76/1/17	Cardiovascular disease with rheumatoid
	75/3/499	Psoriatic arthritis
	74/7/1327	Spondyloarthritis
	72/12/1905	Glucocorticoid therapy in rheumatic diseases
	70/3/414.short	Inflammatory rheumatic diseases

Table 5.1: Data source for our corpus.

The annotation of the dataset was performed by the author to mark the concepts and relations as shown in Table 5.3. Some difficulties became evident when dealing with an **Actor** or an **Object** as well as a **Modifier**. For example, the annotator might at times mix up tagging a word as an **Actor** in a passive sentence instead of an **Object**, and vice versa. Determining if there is a **Modifier** in a phrase can also be challenging. For example, in the phrase: "*adjuvant therapy*", the annotator initially

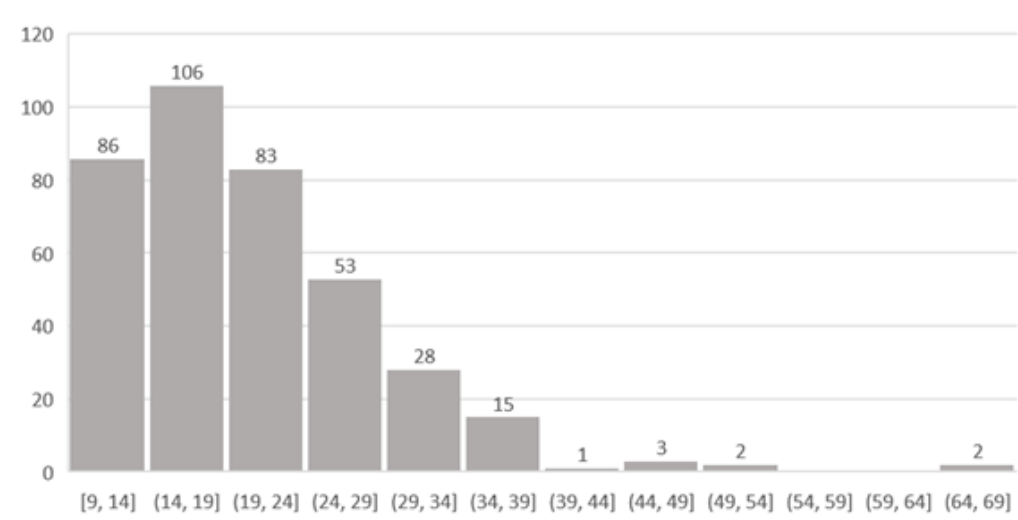


Figure 5.1: Sentence distribution over token counts. X axis denotes the interval of token size in a sentence. Y axis denotes the number of sentences having a particular token size.

marked both words as an *Object* phrase. On further examination, it was then revised so that *adjuvant* is the *Modifier* of the *Object therapy*, i.e., *adjuvant* is the adjective of *therapy*. We believe that it will also be more difficult if we want to have a more fine-grained concept in our annotations. This will also affect the performance of the learning model as currently there is not enough data to support many label instances, and hence the model will not be able to properly learn the representation for each label. Although all considerations have been taken into account, it is still possible that there are inconsistencies and/or ambiguities left in our final dataset (Rahman and J. Bowles, 2021b).

To check the quality of the annotation, we take 10% random subset from the training set to be annotated by a second annotator following the annotation scheme. We chose the second annotator who has some background in medical environment<sup>4</sup>. This process will show the clarity to follow the annotation scheme as well as to get the metrics of the annotation’s quality. Table 5.2 shows the Inter-Annotator Agreement (IAA) for the concepts and relationships. With Cohen’s Kappa and F1-score above 80%, it shows that there is a strong agreement between the two annotators (McHugh, 2012). This also shows that the annotation scheme is adequate to give a clear instruction for others to follow.

<sup>4</sup>At the time of this annotation was conducted, the second annotator had just finished their PhD from the School of Medicine, University of St Andrews.

	Concept	Relationship
Cohen’s Kappa	85.0	88.1
F1-score	83.7	80.9

Table 5.2: Inter-Annotator Agreement (IAA) for the concept and relationship instances (in %).

		Description
Concept	<b>Action</b>	an operation performed on some <b>Object</b> by an <b>Actor</b>
	<b>Actor</b>	the performer of an <b>Action</b> or the owner of an <b>Object</b>
	<b>Object</b>	the entity that an <b>Action</b> is performed on
	<b>Modifier</b>	all modifiers of an <b>Action</b> , an <b>Actor</b> , or an <b>Object</b>
Relationship	<b>actor-of</b>	the <b>Actor</b> of an <b>Action</b>
	<b>acts-on</b>	the <b>Object</b> that an <b>Action</b> is performed on
	<b>choice</b>	for two entities connected by <i>or</i>
	<b>joint</b>	for two entities connected by <i>and</i>
	<b>modifier</b>	an entity that modifies another entity
	<b>owner</b>	the <b>Actor</b> who owns an <b>Object</b>
	<b>receiver</b>	for the <b>Actor</b> who receives an <b>Action</b>
<b>other</b>	for other kind of relationships	

Table 5.3: Concepts and relationships in clinical guideline

The final concepts and relationships with their definition are explained in Table 5.3. As we discussed in Section 4.2.3 (p. 55), we introduced several new relationships that were absent in Table 4.2 (p. 43). We have the original **actor-of**, **acts-on** and **modifier**. We also have several additional relationships between two entities, namely **choice**, **joint**, **owner**, **receiver**, and **other**.

Table 5.4 shows the counts of named-entity instances in three different formats: the basic, BIO and BILUO. The basic format is used in the Semantic Role Labelling (SRL) model, whereas the BILUO format is used for fine-tuning the spaCy model. We use the BIO format for the baseline neural model and the fine-tuned Bidirectional Encoder Representations from Transformers (BERT) models as discussed in Section 4.2.2 (p. 53). This table also shows that for the **Object** and **Modifier** entities in our dataset, they can occur as a single word or as a phrase (marked by the existence of BILUO tags), whereas for the **Actor** and **Action** entities, they only happen as a single-token entity (marked by the only existence of an U tag).

We can see that the dataset is imbalanced; in the basic format, **Action**, **Object** and **Modifier** concepts appear frequently whereas the **Actor** concept is underrepresented,

Concept	#Instances	(%)	BIO tag	#Instances	(%)	BILUO tag	#Instances	(%)
						U-object	915	11.5
						B-object	593	7.4
						I-object	175	2.2
						L-object	593	7.4
			B-object	1507	18.9	U-modifier	794	10
			I-object	769	9.7	B-modifier	134	1.7
			B-modifier	928	11.6	I-modifier	91	1.1
Action	630	26.2	I-modifier	225	2.8	L-modifier	134	1.7
Actor	261	10.8	B-action	472	5.9	U-action	472	5.9
Object	691	28.7	B-actor	281	3.5	U-actor	281	3.5
Modifier	825	34.3	0	3785	47.5	0	3785	47.5
Total	<b>2407</b>			<b>7697</b>			<b>7697</b>	

Table 5.4: Counts of named-entity instances in the original, BIO, and BILUO format.

that is, between 2.4 to 3.2 times rarer. The specialisations into the BIO and BILUO formats could not alleviate this problem as the ratios between the most common and the rarest entities are now 6.8 and 10.5, excluding the 0 tag. This phenomenon of class imbalance can affect the learning performance as the models may be biased towards the majority classes or may disregard the minority classes completely in extreme cases (Johnson and Khoshgoftaar, 2019).

The counts of relationship instances are shown in Table 5.5. The contrast in the number of instances between `actor-of` and `acts-on` is due to the fact that there are many actions without explicit actors in our dataset. Furthermore, some actors are not involved in any action, i.e., they just have some properties to modify them.

The relationships are further specialised for the positions of their arguments. In the phrase: "`<e1> give </e1> vitamins to young <e2> people </e2>`", the relationship between `give` and `people` is `receiver(e1,e2)` as the receiver of the giving comes after the action of giving. Whereas in the sentence: "`For <e1> people </e1> with diabetes, <e2> give </e2> insulin.`", it is `receiver(e2,e1)` as the receiver comes before the giving action. We use this specialised format for the baseline and the fine-tuned BERT models, whereas the basic one is used for the SRL model.

Although technically `actor-of`, `acts-on`, `modifier`, `owner`, and `receiver` can have two different order of arguments, in our dataset, the actor of an action always comes before the action in all `actor-of` instances. Furthermore, it is also clear that the

Relationship	#Instances	(%)	Relationship	#Instances	(%)
			actor-of(e1,e2)	176	6.7
			acts-on(e1,e2)	493	18.8
			acts-on(e2,e1)	75	2.9
			choice	116	4.4
actor-of	176	6.7	joint	150	5.7
acts-on	568	21.7	modifier(e1,e2)	576	22
choice	116	4.4	modifier(e2,e1)	216	8.3
joint	150	5.7	other	351	13.4
modifier	792	30.3	owner(e1,e2)	225	8.6
other	351	13.4	owner(e2,e1)	6	0.2
owner	231	8.8	receiver(e1,e2)	87	3.3
receiver	234	8.9	receiver(e2,e1)	147	5.6
<b>Total</b>	<b>2618</b>			<b>2618</b>	

Table 5.5: Counts of relationship instances

distribution of instances is imbalanced, as the ratios between the most common and the rarest relationship instances is 6.8 for the basic format and 110 for the specialised one. Similarly to Named-Entity Recognition (NER), this imbalance may undoubtedly affect the performance of the learning models.

## 5.2 Experiments

After finalising the dataset, we ran several experiments to evaluate the performance of our SRL approach, NER, and relation classification. We use the common evaluation metrics precision, recall and F1-score. Precision is defined as the percentage of predicted instances that are correct, whereas recall is defined as the percentage of correct instances that are predicted by the model. The F1-score is computed as the harmonic mean of precision and recall.

### 5.2.1 Experiments on Semantic Role Labelling (SRL)

Table 5.6 shows the performance of several classifiers for the SRL approach. For each classifier, we perform evaluation using tenfold cross-validation setting, i.e., in every fold, there will be ten equal portions of data where one portion out of ten will be used as testing. We can see variations of trend for each performance metric. Random forest has the best performance (79.4%) for correctly predicting the concepts and



relations in the sentences, i.e., 4 out of 5 annotations are correct. Meanwhile, perceptron is the best for predicting all correct concepts and relations (68.3%), roughly 7 out of 10 correct annotations can be predicted. Using the F1-score, the best one is achieved by perceptron (64.0%) (highlighted in gray).

Classifier	Precision	Recall	F1-score
Decision tree	74.9	50.7	60.4
Random forest	<b>79.4</b>	47.4	59.3
Perceptron	60.3	<b>68.3</b>	<b>64.0</b>

Table 5.6: Semantic role labelling evaluation scores for several classifiers (in %).

## 5.2.2 Experiments on Named-Entity Recognition (NER)

For the NER approach using neural network models, the performances can be seen in Table 5.7. Here, we only show five experiments using Global Vector (GloVe) embeddings and two top results using word2vec and fastText. We ran experiments using every possible combination of features from Table 4.3 (p. 50). We found the best F1-score of 86.7% using the combination of POS feature and GloVe embeddings of size 300. For word2vec, the best result is achieved when the embedding dimension equals to 100 with the combination of POS and parent POS as its features. Additionally, when using fastText, the top score is attained when using Wikipedia dataset and combined with POS as the features. All five GloVe models shown in the table use Wikipedia as the dataset.

Input features	Precision	Recall	F1-score
GloVe 300 + POS	85.4	88.1	<b>86.7</b>
GloVe 200 + POS & Lemma	85.5	87.9	86.7
GloVe 300 + POS & Parent	84.6	<b>88.8</b>	86.6
GloVe 300 + POS & Dependency	85.3	87.6	86.4
GloVe 300 + POS & Parent & Dependency	<b>85.6</b>	87.2	86.4
fastText 300 + POS	82.9	83.1	83.0
word2vec 100 + POS & Parent	79.3	81.7	80.5

Table 5.7: Named-entity recognition evaluation scores for several word embeddings and input features (in %)

From the results shown in Table 5.6 and Table 5.7, we can say that the performance of the SRL approach is always under the NER model using LSTM. For further

experiments and comparisons, we will only use the NER model with 300 dimensions GloVe embeddings and POS feature as our baseline (highlighted in gray).

### 5.2.3 Fine-tuning spaCy and BERT models for NER

In addition to building our baseline model, we also fine-tuned some state-of-the-art architectures such as spaCy (p.58) and BERT (p.59) for the NER task as discussed in Section 4.2.4 (p. 58). Concretely, we fine-tuned BERT and some of its variants namely ALBERT (p.62), RoBERTa (p.63), and BIO\_ClinicalBERT (p.63). At the time of this experiments were conducted, we used the large spaCy model `en_core_web_lg` that has been pre-trained using the transition-based chunking model (Lample et al., 2016). In its current development, spaCy uses RoBERTa model as its backend<sup>5</sup>.

The performance of our baseline NER model Table 5.7 compared to fine-tuning spaCy and BERT variants is shown in Table 5.8. We can see that the baseline model performs better than the fine-tuned spaCy model by a margin of  $\approx 5\%$ . For the BERT model, the performances range from almost equal for ALBERT to better by a margin of  $\approx 3\%$  for BIO\_ClinicalBERT. Since BIO\_ClinicalBERT has been pre-trained on clinical texts and our domain is clinical guideline texts, we suspect this to be the reason why it outperforms other BERT variants for the NER task.

Model	Precision	Recall	F1-score
Baseline	85.4	88.1	86.7
spaCy	80.8	82.5	81.6
BERT	87.7	<b>90.4</b>	89.0
ALBERT	86.9	87.9	87.4
RoBERTa	87.0	89.6	88.3
BIO_ClinicalBERT	<b>88.4</b>	90.2	<b>89.3</b>

Table 5.8: Evaluation F1-score values for NER model (in %)

For multiclass classification, there are several average metrics that can be used when calculating the precision, recall, and F1-score, such as<sup>6</sup>:

<sup>5</sup><https://spacy.io/usage/v3#features-transformers>

<sup>6</sup>[https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1\\_score.htm](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html)

- micro: Calculate metrics globally by counting the total true positives, false negatives and false positives.
- macro: Calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account.
- weighted: Calculate metrics for each label, and find their average weighted by support (the number of true instances for each label). This alters 'macro' to account for label imbalance; it can result in an F-score that is not between precision and recall.

As we have seen in the dataset analysis, the concept classes in our corpus are mostly imbalance. Therefore, when calculating the performance for each instance, we will use the weighted average as it takes into account the number of class instance.

Table 5.9 shows the performance detail of the baseline model, fine-tuned BERT and BIO\_ClinicalBERT per NER class instance. In two cases, namely `Action` and `Object`, BERT is slightly better than BIO\_ClinicalBERT. However, for recognising `Actor` and `Modifier`, BIO\_ClinicalBERT is better than BERT.

Entity type	Baseline			BERT			BIO_ClinicalBERT		
	P	R	F1	P	R	F1	P	R	F1
<code>Action</code>	92.5	99.0	95.6	94.2	99.0	96.6	95.1	98.0	96.5
<code>Actor</code>	97.6	80.4	88.2	90.6	94.1	92.3	90.7	96.1	93.3
<code>Modifier</code>	81.5	79.8	80.6	86.1	83.4	84.7	87.8	86.0	86.9
<code>Object</code>	80.6	88.0	84.1	85.9	91.4	88.6	86.2	89.4	87.8
Weighted	85.4	88.1	<b>86.7</b>	87.7	90.4	<b>89.0</b>	88.4	90.2	<b>89.3</b>

Table 5.9: Instance score for NER (in %)

The scores in Table 5.9 are calculated following the evaluation metrics defined by Tjong Kim Sang and De Meulder (2003):

"Precision is the percentage of named entities found by the learning system that are correct. Recall is the percentage of named entities present in the corpus that are found by the system. A named entity is correct only if it is an exact match of the corresponding entity in the data file."

It is obvious that the definition above only takes into account strict cases where either (1) the predicted entity is the same with the gold-standard, (2) the model predicts

entity that is not in the gold-standard, or (3) the model misses predicting an entity in the gold-standard. These three cases and one neutral case where the model and the gold-standard agree that there is no entity are illustrated in Table 5.10 rows 1, 2, 3 and 4. However, as we can see, this definition discards partial matches due to the wrong boundary (rows 6 and 7) or the cases where the model predicts the wrong entity (row 5). These last three phenomena, rows 5 - 7, are called a labelling error (le), a boundary error (be), and a label-boundary error (lbe) by Chris Manning<sup>7</sup>. He argues that using only F1-score can be misleading when it is used to measure the performance of a NER model.

Case		Example
1	The gold-standard and the model prediction are the same	GS: O B-object I-object O MP: O B-object I-object O
2	The model predicts an entity not in the gold-standard	GS: O O O O MP: O B-object I-object O
3	The model misses an entity	GS: O B-object I-object O MP: O O O O
4	The model and the gold-standard agree that there is no entity	GS: O O O O MP: O O O O
5	The model predicts the wrong entity	GS: O B-object I-object O MP: O B-actor I-actor O
6	The predicted entity is correct but the boundary is not	GS: O B-object I-object O MP: B-object I-object I-object O
7	The predicted entity and the boundary are wrong	GS: O B-object I-object O MP: B-actor I-actor I-actor O

Table 5.10: Cases for measuring a NER model. GS stands for "gold standard" whereas MP stands for "model prediction". For example, in the case number 2, the actual labels are "O O O O" but the second and the third labels are wrongly predicted as "B-object" and "I-object" by the model.

To take into account the cases where the model can predict the wrong entities or the wrong boundaries, a different evaluation is proposed based on the drug-drug interactions extraction in biomedical texts (I. Segura-Bedmar, Martinez and Herrero-Zazo, 2013). The method calculates the performance based on strict evaluation, exact and partial boundary matching (regardless to the entity type), as well as the entity type matching. This alternative evaluation uses the scoring categories proposed by the Message Understanding Conference (MUC; Chinchor and Sundheim, 1993), namely:

- Correct (COR): the gold-standard and the model prediction match;

<sup>7</sup><https://nlpers.blogspot.com/2006/08/doing-named-entity-recognition-dont.html>

- Incorrect (INC): the gold-standard and the model prediction do not match;
- Partial (PAR): the gold-standard and the model prediction have some overlap;
- Missing (MIS): the model misses an entity in the gold-standard;
- Spurious (SPU): the model predicts an entity not in the gold-standard;

Furthermore, there are two additional metrics for the measurement of annotations of the gold-standard and the model predictions:

1. The number of annotations in the gold-standard

$$POSSIBLE(POS) = COR + INC + PAR + MIS = TP + FN \quad (5.1)$$

2. The number of annotations produced by the system

$$ACTUAL(ACT) = COR + INC + PAR + SPU = TP + FP \quad (5.2)$$

Now the calculation of precision and recall can be defined based on the new evaluation categories as defined in the formulae below. Note that although the strict and exact match fall into the same formula (the same with partial and type match), the individual category (such as the number of COR, INC, PAR, MIS or SPU) may give different value.

- Strict and exact match

$$Precision = \frac{COR}{ACT} = \frac{TP}{TP + FP} \quad (5.3)$$

$$Recall = \frac{COR}{POS} = \frac{TP}{TP + FN} \quad (5.4)$$

- Partial and type match

$$Precision = \frac{COR + 0.5 * PAR}{ACT} \quad (5.5)$$

$$Recall = \frac{COR + 0.5 * PAR}{POS} \quad (5.6)$$

Following the new strategies, we calculate the F1-score of the baseline and the fine-tuned BERT and BIO\_ClinicalBERT using the strict, exact and partial evaluations

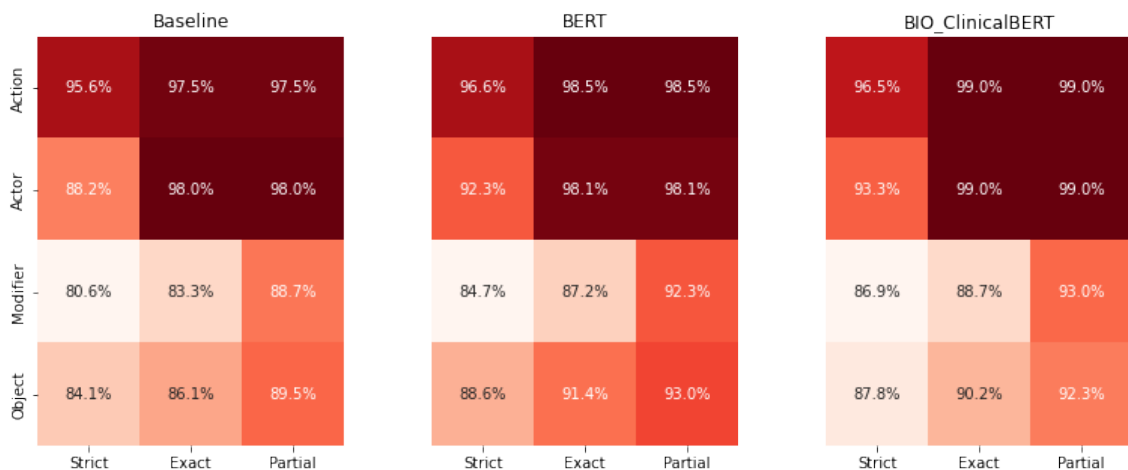


Figure 5.2: Heatmaps of the difference between the baseline, BERT and BIO\_ClinicalBERT NER confusion matrix based on the strictness of entity classification. The X-axis shows the evaluation strategy, and the Y-axis shows the named-entities. The numbers are showing F1-scores (in %)

incorporating the `nervaluate`<sup>8</sup> library. The result is shown as heatmaps in Figure 5.2.

For the `Actor` entity, compared to BERT and BIO\_ClinicalBERT, the baseline model has the lowest performance following the strict evaluation, but almost the same for the exact and partial ones. This means that the mistakes for the `Actor` entity prediction in the baseline model are due to the boundaries or the prediction of nonexistent entities (row 2 in Table 5.10).

Moreover, the baseline model cannot perform well to predict the `Object` entity, no matter what evaluation is used, as the score between strict, extract and partial evaluation cannot reach 90%. This phenomenon does not happen for BERT and BIO\_ClinicalBERT. This trend is also similar for the `Modifier` entity that suffers the lowest score in the baseline model.

By contrast, all models perform really well when predicting the `Action` entity. We suspect this is because the `Action` entity is almost always a verb in the sentence. However, not all verbs in the sentence are `Action` entities. The examples below illustrate this phenomenon.

The patients *have* undergone surgery. Dynamic *have*

<sup>8</sup><https://github.com/ivyleavedtoadflax/nervaluate>

The patients *have* diabetes. *Stative have*

In the dynamic *have*, the verb *have* serves as the marker for the real **Action** entity which is *undergone*, while it serves as the marker for the owning/owner relationship in the stative *have*. As all models have F1-score greater than 95% for the strict, exact and partial evaluations, we can say that all models have learnt good representations for the **Action** entity.

Finally, although the performance of the baseline model is lower than the other models, in terms of training time, BERT and BIO\_ClinicalBERT took considerably longer time to train. For 50 epochs, it took  $\approx 10$  minutes to train the baseline, whereas BERT and BIO\_ClinicalBERT took  $\approx 50$  minutes. Another disadvantage is that BERT and BIO\_ClinicalBERT need more computation resources, such as big memory and Graphics Processing Unit (GPU), which are not always available in most computing environments. The need of GPU also affects the prediction time as running the prediction on Central Processing Unit (CPU) is noticeably slower compared to the baseline model. This factor can be used as consideration when integrating the model into a bigger system: lower performance but faster process or vice versa.

#### 5.2.4 Experiments on relation classification

	Precision	Recall	F1-score
word2vec 100	82.9	82.5	82.7
fastText 300 Wikipedia	83.1	83.3	83.2
GloVe 300 Common Crawl 840B tokens	85.8	84.0	84.5
GloVe 300 Wikipedia	85.6	85.5	85.3
GloVe 200 Wikipedia	88.0	87.8	87.8
GloVe 300 Common Crawl 42B tokens	87.9	<b>88.6</b>	87.8
GloVe 100 Wikipedia	<b>89.0</b>	88.2	<b>88.4</b>

Table 5.11: Evaluation of F1-score (in %) on relation classification for several GloVe word embeddings with varying dimension size.

The result for relation classification is shown in Table 5.11. To calculate the value for each metric, we compared the model prediction against the gold standard. We do not calculate directly from the output of the NER model hence we do not propagate the NER errors. Here, we have the top seven models built under the architecture

described in Figure 4.7 (p. 56), varying on the word embeddings sources and dimensions. The highest F1-score is achieved by using the 100 dimensions GloVe embeddings trained on Wikipedia data. We can see the trend that the performance is better if the word embeddings dimension is lower. None of the top seven models achieves a score greater than 90%. Similar to the results shown in Table 5.7 (p. 70), our experiments using word2vec and fastText as word embeddings are not better than using GloVe. For further experiments and comparisons, we will use the model using 100 dimensions Wikipedia GloVe embeddings as the baseline (highlighted in gray). The performance detail, with the fine-tuned models, is further explained in Table 5.13 (p. 78)

### 5.2.5 Fine-tuning BERT models for relation classification

Table 5.12 shows the comparison of the baseline model with two BERT variants. Different from NER, we did not run experiments using ALBERT and RoBERTa for this task as we only take the top 2 BERT models from NER. Further consideration factors include the limited amount of memory and GPU resources that we have.

Model	Precision	Recall	F1-score
Baseline	89.0	88.2	88.4
<b>BERT</b>	<b>92.7</b>	<b>91.6</b>	<b>91.8</b>
BIO_ClinicalBERT	91.2	90.1	90.3

Table 5.12: Evaluation F1-score values for NER model (in %)

Similarly to NER, BERT and BIO\_ClinicalBERT have better performance than the baseline model. However, surprisingly, BERT performs better here than BIO\_ClinicalBERT not only in terms of the F1-score, but also in the precision and in the recall. Our explanation for this is that the relation tags are quite generic and do not really need clinical knowledge. Therefore, for this reason BERT can perhaps be better at generalising its learning process, leading to a better performance.

The performance detail for relation classification is shown in Table 5.13. In three relationships, namely `acts-on(e2,e1)`, `modifier(e2,e1)` and `receiver(e1,e2)`, the baseline model performs better than BERT and BIO\_ClinicalBERT in terms of the F1-score. Especially for `receiver(e1,e2)`, even though its instances are few, the baseline can capture its representation better than BERT. This phenomenon is



Relations	Baseline			BERT			BIO_ClinicalBERT		
	P	R	F1	P	R	F1	P	R	F1
actor-of(e1,e2)	88.9	80.0	84.2	95.2	100	97.6	87.0	100.0	93.0
acts-on(e1,e2)	92.5	90.7	91.6	98.1	96.3	97.2	98.1	96.3	97.2
acts-on(e2,e1)	50.0	50.0	50.0	27.3	75.0	40.0	27.3	75.0	40.0
choice	70.0	87.5	77.8	85.7	75.0	80.0	85.7	75.0	80.0
joint	100.0	87.5	93.3	88.9	100	94.1	80.0	100.0	88.9
modifier(e1,e2)	96.3	91.2	93.7	100	96.5	98.2	96.5	96.5	96.5
modifier(e2,e1)	59.3	80.0	68.1	73.3	55.0	62.9	73.3	55.0	62.9
other	84.9	87.5	86.2	90.9	93.8	92.3	89.7	81.3	85.3
owner(e1,e2)	100.0	96.2	98.0	100	96.2	98.0	96.2	96.2	96.2
owner(e2,e1)	0	0	0	0	0	0	0	0	0
receiver(e1,e2)	90.0	90.0	90.0	80.0	80.0	80.0	88.9	80.0	84.2
receiver(e2,e1)	95.2	90.9	93.0	95.7	100	97.8	100	100	100
Weighted	89.0	88.2	<b>88.4</b>	92.7	91.6	<b>91.8</b>	91.2	90.1	<b>90.3</b>

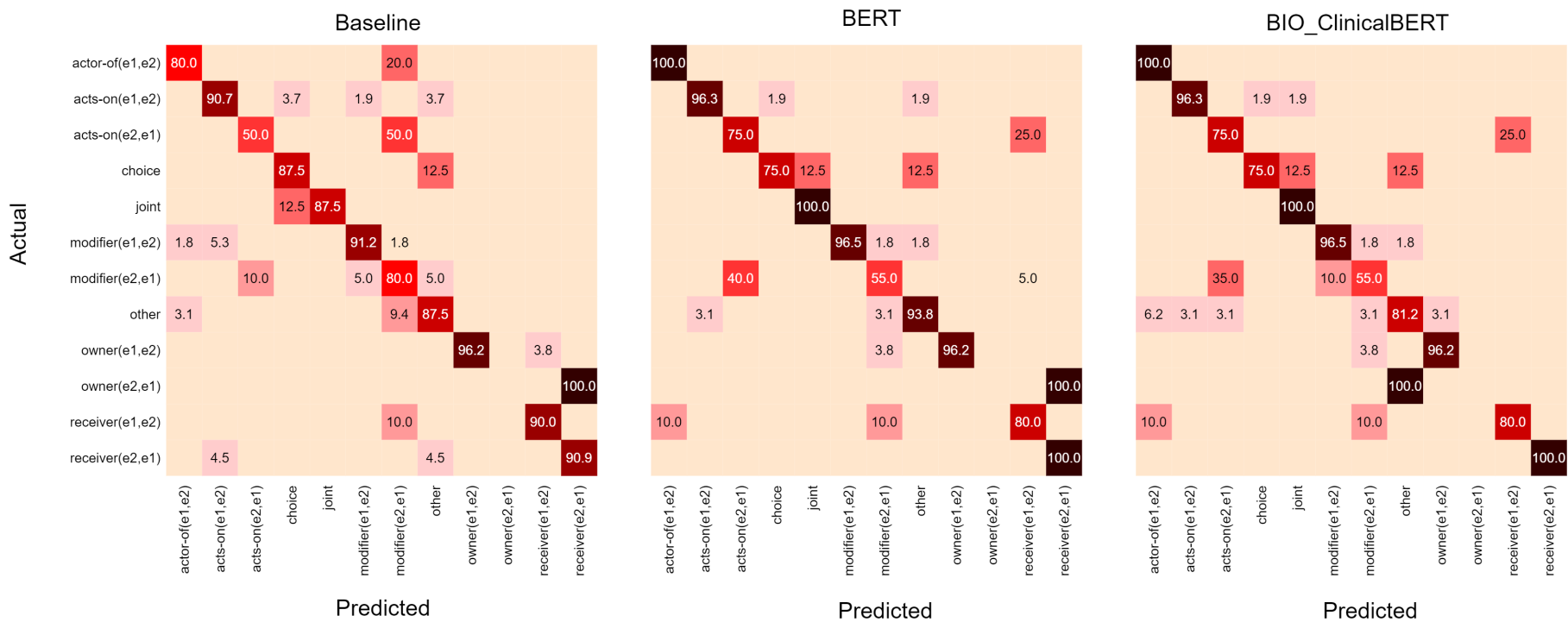
Table 5.13: Evaluation score for relation classification (in %)

not always true. Another case worth noting is the `acts-on(e2,e1)`: both BERT and BIO\_ClinicalBERT perform equally, where the models can predict more instances of `acts-on(e2,e1)` in the test set (the recall score). However, for the baseline, the model is equal when correctly predicting the instances (the precision score) as well as when predicting the correct instances (the recall score). And lastly, for `owner(e2,e1)`, no model can predict it correctly. This is unsurprising as there are only 6 such instances in our dataset which explains why the models cannot learn this representation properly.

Figure 5.3 shows the heatmaps of the relation classification’s confusion matrix for the three models. We can see that in general, different models make similar mistakes. For `owner(e2,e1)`, both the baseline and BERT model tend to assign `receiver(e2,e1)` while BIO\_ClinicalBERT assigns `other`. Using the notation introduced in Table 4.5 (p. 57), the sentence below is an instance for this phenomenon where *proteinurea* is the first element and *women* is the second element. The baseline and BERT somehow recognise that the word *women* is the receiver of the *proteinurea* action.

If dipstick screening is positive, use albumin:creatinine ratio or protein:creatinine ratio to quantify <e1> proteinuria </e1> in pregnant <e2> women </e2>.

Figure 5.3: Heatmaps of the difference between the baseline, BERT and BIO\_ClinicalBERT relation classification confusion matrix. The numbers are showing F1-scores (in %)



For `acts-on(e2,e1)`, both BERT and BIO\_ClinicalBERT have the tendency to mistakenly assign `receiver(e2,e1)`, whereas the baseline assigns `modifier(e2,e1)`. This case happens mostly in passive voice where the verb can act as a participle or an adjective. For the former case, BERT and BIO\_ClinicalBERT consider the second element as an action and the first element is the receiver of the action. In contrast, the baseline model falls to the latter case where the second element is an adjective for the first element. The sentence below is an instance of this phenomenon where *osteoporosis* is the first element and *assessed* is the second element.

In patients with axial SpA without syndesmophytes in the lumbar spine on conventional radiography, `<e1>osteoporosis </e1>` should be `<e2>assessed </e2>` by hip DXA and AP-spine DXA.

In contrast to the previous case, all models tend to label `modifier(e2,e1)` as `acts-on(e2,e1)`. However, with further investigation, this is not a mistake in the model but an incorrect annotation. As a verb can function as a participle or an adjective in passive voice, we suspect the annotator made a mistake when annotating the gold-standard. This phenomenon shows that the models can learn very well to differentiate between the two cases. The sentence below illustrates the phenomenon.

Where appropriate, functional tests, including the exercise tolerance `<e1>test </e1>`, should be `<e2>considered </e2>` to aid in the risk stratification of patients with known CAD.

In terms of running time, the difference between the baseline model and BERT is even greater for this task compared to NER. For 50 epochs, our model took  $\approx$  30 minutes whereas BERT and BIO\_ClinicalBERT took  $\approx$  3.5 hours and  $\approx$  5.5 hours, respectively. Due to the limitation of our GPU memory, we also lowered our training batch size to 8 for BERT and BIO\_ClinicalBERT which affected the overall training time. Likewise with NER, this factor can also be taken into consideration when integrating the model into a bigger system, as the prediction time is much longer for BERT and BIO\_ClinicalBERT than the baseline model.

## 5.3 Improving the Models

After running the experiments, there are still a few options if we want to improve our models further:

**Investigate the data** The misclassified samples can be used further to see if there is a consistent labelling error. If there are overlapping cases, we can create new concepts/relationships or merge old ones where applicable.

**Cleaning up the data** We see that the dataset itself still contains label noise, i.e., the annotator may have used the wrong label. Refining the dataset could improve the quality of the data as well as the performance of the models. Furthermore, we also believe that having more annotators can increase the quality of the data.

Another way to handle label noise is label smoothing. A study on the CIFAR-10 image data has shown that smoothing exhibits denoising effects by making the models less overconfident when drawing the decision boundaries (Lukasik et al., 2020).

**Resampling the dataset** One way to make the dataset more balanced is by resampling during training. There are several ways of doing this, such as over-sampling (or replicating) the minority classes and/or undersampling (or eliminating) the majority classes (Batista et al., 2004), or bootstrapping (Reed et al., 2015).

**Apply larger models** As we had some limitations with the computing resources available, we could not apply larger models when training our data. In combination to having larger datasets, applying larger models may improve the models' performance. However, we have to make a trade-off between gaining better performance and reducing the carbon footprint when training large models (Hao, 2019; Dhar, 2020)

## 5.4 Summary

In this chapter, we have discussed the experiments conducted over our machine learning models. We found that fine-tuning state-of-the-art architectures give us the best performance. We also found that, in addition to the size of our dataset, we need

to consider the effect of label noise and imbalanced datasets. We discussed different solutions to address the identified problems.

In the next chapter, we will discuss how our annotated clinical guideline sentences can be transformed further and be explored for formal verification.

*Toute leur vie était régie non par des lois, des statuts ou des règles,  
mais selon leur volonté et leur libre arbitre.*

François Rabelais, *Gargantua*

# 6

## Formal Model Generation and Verification

In this chapter, we describe how we transform guideline texts into formal specification. This transformation process starts after we extract the concepts either by parsing the guideline texts written in the Controlled Natural Language (CNL) or using Machine Learning (ML) as discussed in Chapter 3 and Chapter 4 respectively.

We begin by explaining different logic forms that can be used for reasoning in our framework, i.e., Propositional Logic (PL), First-Order Logic (FOL), Linear Temporal Logic (LTL), and Computation Tree Logic (CTL). We will then discuss the task of model checking and the use of model checking tools such as UPPAAL and PRISM, as well as constraint solver such as Z3 to verify the transformed guideline text.

Figure 6.1 shows the pipeline of a clinical guideline verification. Once the formal specification is generated (either for a model checker or a constraint solver), it can be verified against some properties. In addition, further information may be extracted from available health records and added to the specification to help the verification process.

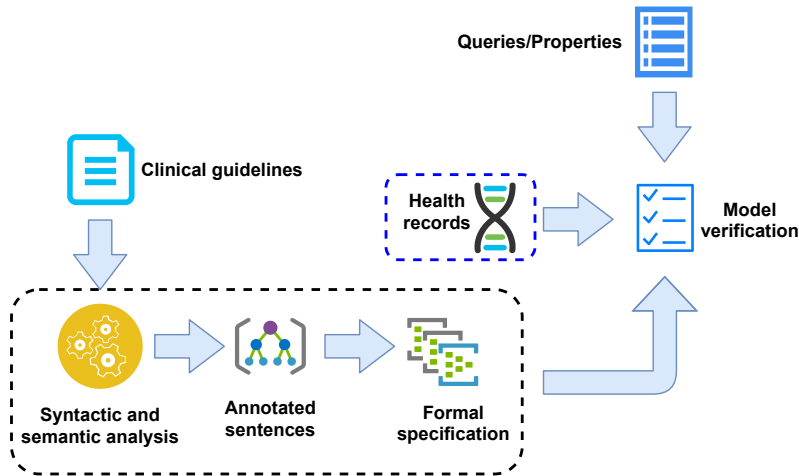


Figure 6.1: Pipeline for verifying a clinical guideline

The use of a model checker or a constraint solver are quite different. Temporal logics such as LTL and CTL (and further variants) are used in model checking to capture properties we want to check against a model. In the context of medical guidelines, we can use model checking to verify, for instance, treatment options across a guideline where the guideline represents the model. Some guidelines, however, do not lend themselves to be represented as a model. In such cases, a constraint solver approach may be more useful as we will see in this chapter. Part of the work presented in this chapter has been published in Rahman and J. K. F. Bowles (2017).

## 6.1 Logical Form

To verify the correctness of the clinical guideline texts, they need to be transformed into some formal specification. Further, we can specify some propositions or properties in logical forms to check if they are entailed by the specification. In other words, the specification will be the model to prove if the given properties are correct or not. This section will briefly explain some logics that will be used to define properties of the model checker or constraint solver mentioned in Section 6.2 (p. 89). For example, we specify the properties written in timed computation tree logic, a variation of computation tree logic, when checking against UPPAAL and PRISM. For Z3 constraint solver, the properties are expressed in a language family of first-order logic.

### 6.1.1 Propositional Logic

Propositional logic can be used to formally represent statements through the use of propositions and certain logical connectors. Each proposition has its truth value and they can be combined using the logical connectors to form compound propositions. A stand alone proposition that is not connected to the others is called an atomic proposition. The connectors in propositional logic are negation ( $\neg$ ), conjunction ( $\wedge$ ), disjunction ( $\vee$ ), implication ( $\rightarrow$ ) and equivalence ( $\leftrightarrow$ ).

Propositional logic can represent and be used to make inference about the world. For example, if  $r$  denotes the sentence: "*it rains*",  $w$  denotes the sentence: "*the street is wet*", example  $A1$ ,  $A2$ , and  $A3$  are valid propositional sentences.

( $A1$ ) "*if it rains, the street is wet*" is represented by  $r \rightarrow w$

( $A2$ ) "*it rains and the street is wet*" is represented by  $r \wedge w$

( $A3$ ) "*it does not rain or the street is not wet*" is represented by  $\neg r \vee \neg w$

The logical connectors define the truth value of the propositions. Table 6.1 shows the truth value of two propositions  $\mathbf{P}$  and  $\mathbf{Q}$  with the logical connectors where  $T$  denotes **true** and  $F$  denotes **false**.

$\mathbf{P}$	$\mathbf{Q}$	$\neg\mathbf{P}$	$\mathbf{P} \wedge \mathbf{Q}$	$\mathbf{P} \vee \mathbf{Q}$	$\mathbf{P} \rightarrow \mathbf{Q}$	$\mathbf{P} \leftrightarrow \mathbf{Q}$
$T$	$T$	$F$	$T$	$T$	$T$	$T$
$T$	$F$	$F$	$F$	$T$	$F$	$F$
$F$	$T$	$T$	$F$	$T$	$T$	$F$
$F$	$F$	$T$	$F$	$F$	$T$	$T$

Table 6.1: Propositional logic truth table

### 6.1.2 First-order Logic

Although propositional logic is useful to model facts about things in general, it still has some limitations. For example, propositional logic can be used to model the sentences: "*Socrates is a human*" and "*Plato is a human*" as  $p$  and  $q$  respectively. But it lacks the ability to say that both Plato and Socrates belong to the human domain.

On the other hand, FOL has the ability to express that an individual of some sort



of domain is being considered, as well as to express the equality of two propositional symbols. Together with the connectors from propositional logic, this makes first-order logic more powerful in expressing statements about the world.

Using the previous example of Plato and Socrates, these statements below are valid first-order logic sentences.

(B1)  $Human(socrates)$

(B2)  $Mortal(socrates)$

(B3)  $\forall x Human(x) \rightarrow Mortal(x)$

(B4)  $\exists x Human(x) \wedge Mortal(x)$

(B1) and (B2) show two predicates to model the *Human* and the *Mortal* domain and *socrates* is a member of these domains. Two new quantifier symbols present in (B3) and (B4). The symbol  $\forall$  in (B3) is the universal quantifier and used to say "for all" or "for every". In other words, it tells that for all  $x$ , if  $x$  is a member of the *Human* domain, then  $x$  is also a member of the *Mortal* domain, i.e., every human is mortal. Another quantifier is the existential quantifier, marked as  $\exists$  in (B4), to say "there exists" or "there is". Statement (B4) can be read as there is  $x$  that is both *Human* and *Mortal*.

### 6.1.3 Temporal Logic and Linear Temporal Logic

Linear Temporal Logic (LTL) extends propositional logic and has the ability to model and reason about situations that involve an ordering of events. As the name suggests, it assumes a linear interpretation of time (one possible future), consider discrete time (as opposed to continuous time) which can be seen as an abstraction. Temporal logic only considers the relative ordering of events and assumes discrete time (Baier and Katoen, 2008). For example, LTL can be used to express: "*the elevator door opens after the open button is pressed*" even though there is no precise timing of when the elevator door will exactly open after the button is pressed. Other more expressive logics with additional characteristics (e.g., bounded temporal operators) exist and can be used when it is important to reason about time, etc. LTL is the simplest temporal logic that we can define, and hence our starting point.

Operator symbol	Alternative symbol	Explanation
$\diamond\psi$	<b>F</b> $\psi$	$\psi$ is eventually true in the future
$\square\psi$	<b>G</b> $\psi$	$\psi$ is globally/always true
$\bigcirc\psi$	<b>X</b> $\psi$	$\psi$ is true in the next state
$\psi_1 U \psi_2$		$\psi_1$ is true until $\psi_2$ becomes true

Table 6.2: Temporal operators in LTL.  $\psi$  denotes a formula.

Temporal logics such as LTL are interpreted over models known as (labelled) transition systems. These models are state-based representations of a system. Table 6.2 shows operators present in LTL to express modality that are absent in propositional logic. Besides these modal operators, LTL also inherits the connectors from propositional logic.

**Definition 6.1 (LTL formulae)** *LTL formulae  $\psi$  under atomic proposition  $\mathbf{a}$  are defined in the following grammar:*

$$\psi ::= a \mid true \mid \psi_1 \wedge \psi_2 \mid \neg\psi \mid \bigcirc\psi \mid \psi_1 U \psi_2$$

Following the Definition 6.1, in LTL, for each state at a moment of time, there is only one possible successor state. Therefore, each time moment has a unique possible future (Baier and Katoen, 2008). Implicitly, a state satisfies an LTL formula if *all paths* from the given state satisfy it, analogous to having a universal quantifier  $\forall$  (Huth and Ryan, 2004). Figure 6.2 illustrates a path for an LTL formula  $\square\diamond\mathbf{a} \rightarrow \square\diamond\mathbf{b}$  from an initial state  $s_1$ .

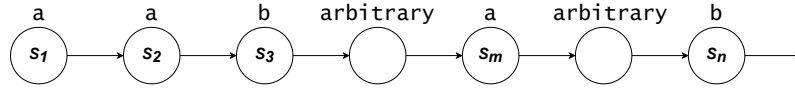


Figure 6.2: An LTL path for  $\square\diamond\mathbf{a} \rightarrow \square\diamond\mathbf{b}$  formula

In Figure 6.2 above,  $\mathbf{a}$  holds in the initial state  $s_1$ .  $\diamond\mathbf{b}$  holds in state  $s_1$  because there is a state in the future of  $s_1$  (for instance  $s_3$ ) where  $\mathbf{b}$  holds.  $\square\diamond\mathbf{b}$  holds in  $s_1$ , because at all future states of  $s_1$ ,  $\diamond\mathbf{b}$  holds. This effectively means that  $\mathbf{b}$  will occur infinitely often. A more complex formula such as  $\square\diamond\mathbf{a} \rightarrow \square\diamond\mathbf{b}$  holds at  $s_1$ , iff  $\square\diamond\mathbf{a}$  holds at  $s_1$  then  $\square\diamond\mathbf{b}$  at  $s_1$ . Following the operators mentioned in Table 6.2 (p. 87),  $\bigcirc\mathbf{a}$  holds in  $s_1$  since  $\mathbf{a}$  still holds in the next state  $s_2$ . And lastly,  $\mathbf{a} U \mathbf{b}$  also holds in  $s_1$  as  $\mathbf{a}$  holds in  $s_1$  and  $s_2$  and not anymore in  $s_3$  where  $\mathbf{b}$  holds. Satisfaction

of arbitrary formula  $\psi$  is determined by checking the individual atomic proposition that make up  $\psi$ .

The examples below show two sentences written as LTL formula.

$$(C1) \quad \Box \Diamond \text{open\_button\_pressed} \rightarrow \Box \Diamond \text{door\_opened}$$

$$(C2) \quad \Box (\neg \text{crit1} \vee \neg \text{crit2})$$

Example (C1) shows how temporal operators can be used in combination. The formula means that if the open button is pressed infinitely often, then the door will open infinitely often. Figure 6.2 illustrates an LTL path for this formula where  $a$  and  $b$  denote `open_button_pressed` and `door_opened` respectively. The second example (C2) shows the property that is expected from any system, that it is always true that no two processes are accessing their critical section at the same time. In our context, LTL can be used, for instance, to express an expected outcome of a medication: "*if a patient takes paracetamol, then the fever will eventually subside.*"

### 6.1.4 Computation Tree Logic

Another logic that uses temporal logic as its base is Computation Tree Logic (CTL). CTL can express an existence of a path that satisfies a formula, in contrast to LTL that implicitly uses the universal quantifier to all its states. In other words, in CTL we do not care only about one possible future for a state  $s_1$ , but all possible future states that can be reached from  $s_1$ , and we can write properties to that effect.

**Definition 6.2 (CTL state formulae)** *CTL state formulae over atomic proposition  $a$  and a path formula  $\psi$  are defined in the following grammar:*

$$\phi ::= \text{true} \mid a \mid \phi_1 \wedge \phi_2 \mid \neg \phi \mid \exists \psi \mid \forall \psi$$

**Definition 6.3 (CTL path formulae)** *CTL path formulae over atomic proposition  $a$  and a state formula  $\phi$  are defined in the following grammar:*

$$\psi ::= \bigcirc \phi \mid \phi_1 U \phi_2$$

In CTL, the formula is expressed as state and path formula. The former is used to assert states and their branching structure, while the latter is used to assert the

temporal properties of paths (Baier and Katoen, 2008). The examples below show the two LTL formulae  $(C1)$  and  $(C2)$  written in CTL.

$$(D1) \forall \square \forall \diamond \text{open\_button\_pressed} \rightarrow \forall \square \forall \diamond \text{door\_opened}$$

$$(D2) \forall \square (\neg \text{crit1} \vee \neg \text{crit2})$$

The formula in example  $(D1)$  means that when open button is pressed infinitely often in all path, the door will also open infinitely often. Figure 6.3 illustrates this formula where  $a$  and  $b$  denote `open_button_pressed` and `door_opened` respectively. Example  $(D2)$  tells that for every state in all path, there is only one process accessing the critical section at a time.

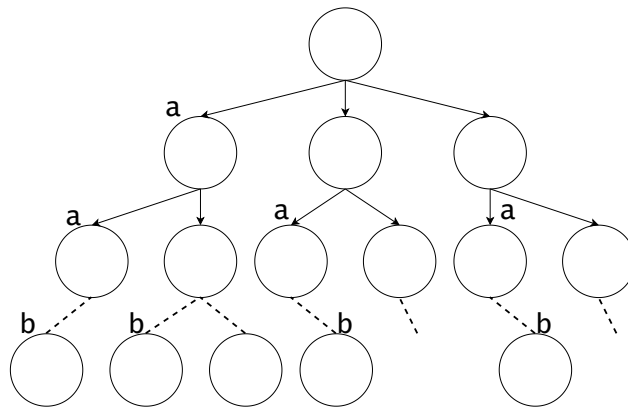


Figure 6.3: A CTL tree for  $\forall \square \forall \diamond a \rightarrow \forall \square \forall \diamond b$  formula

## 6.2 Constraint Solver and Model Checker

Logics such as LTL, CTL, and many others, are used in automated verification with model checkers whereas FOL is useful in the context of finding solutions that satisfy sets of constraints. Model checkers require a finite representation of a model and we can check whether certain properties hold against the model or not. We explore which approach can be more useful in the context of clinical guidelines.

Once a clinical guideline has been transformed into a formal specification, it can be used for reasoning and for instance to check the validity of some properties. Here properties can be anything, such as if some treatments would conflict with others given for another disease, or if a guideline has some missing gaps of information that need to be addressed. To achieve this task, either a constraint solver or a

model checker can be used. The choice will make the transformation into formal specification differ in accordance to the target specific tools. Here we consider the use of the Z3 constraint solver (Moura and Bjørner, 2008) as well as UPPAAL (Behrmann et al., 2004) and PRISM (Kwiatkowska et al., 2011) model checkers.

### 6.2.1 Transforming and verifying guidelines in UPPAAL

UPPAAL<sup>1</sup> is an integrated tool environment, created by a longstanding collaboration between the universities of Uppsala and Aalborg, for modelling, validation and verification of real-time systems seen as networks of timed automata (Behrmann et al., 2004). Timed automata (TA) add the notion of time to standard automata (based on a finite set of states and labelled transitions between them) through a set of variables called *clocks* (Alur and Dill, 1994). Clocks are special variables which can be inspected or reset but not assigned a value (they can be seen as stopwatches). A time unit represents a second, minute or month, depending on what is a sensible unit for the model (note that there is no time in our diabetes therapy algorithm snippet). A constraint can be placed on *locations* (the term used for states in a TA) to denote a *location invariant* (to indicate for instance how long the automaton can remain in the location) and on transitions where it acts as a guard.

In UPPAAL, we can create several instances of processes with the same behaviour. The behaviour is captured in a so-called template (a TA), and one or more instances of that template can be declared for runtime verification. In the example of the type 2 diabetes therapy algorithm shown in Figure 3.1 (p. 25), we need a template for the behaviour captured in the algorithm for treating diabetes, and an additional template to simulate a (random) change of the value of *HbA1c* in the blood for a given patient. Further templates can be added as needed, for instance to capture other disease parameters, etc. These combined templates can be used to represent treatments and the state of individual parameters and tools such as UPPAAL can be used to simulate what may happen to a patient with these conditions over time, as well as infer longterm consequences of treatments.

---

<sup>1</sup><https://uppaal.org/>

### 6.2.1.1 Guideline transformation

The process of generating templates from the T2D therapy algorithm is done automatically by traversing the case frames generated by Algorithm 1 (p. 36) in Section 3.1.2.2. This process is illustrated by the `generate_template` function in Algorithm 2 following the roles definition from Section 3.1.2.2 (p. 33). As UPPAAL specifications are stored in Extensible Markup Language (XML) format, this function, besides writing the output to an XML file, also generates an output of an XML tree object. The implementation of these algorithms has been done in Python programming language.

---

**Algorithm 2:** Algorithm for generating UPPAAL template

---

```
1 def generate_template(case_frames):
2     xml_tree = ElementTree()
3     var_set = set()
4
5     for f in case_frames do
6         num = f.num
7         cpt, cac, cmd, ctv = f.cpt, f.cac, f.cmd, f.ctv
8         agt, act, pat, tov, cact = f.agt, f.act, f.pat, f.tov, f.cact
9
10        get_variables (var_set, cpt, pat)
11        set_values (var_set, ctv, tov)
12        add_invariant (xml_tree, num, cac, cmd, var_set)
13        add_location (xml_tree, num, agt, act, cact, var_set)
14
15        if act == 'consider' then
16            cond = make_branching_condition (cpt, ctv)
17            make_branching (xml_tree, num, cond, var_set)
18
19    build_hba1c_template ()
20    write_to_file (xml_tree)
```

---

Looking back to the example of case frames in Table 3.2 (p. 37) generated from the sentence: "*1 if HbA1c level rises to 48 mmol/mol (6.5%) on lifestyle interventions, the doctor shall: offer standard-release metformin, support to aim for an HbA1c level of 48 mmol/mol (6.5%).*" (highlighted in green in Figure 6.4), line 10 and 11 in Algorithm 2 will identify all variables from CPT and PAT roles and their respective values from CTV and TOV. If the value contains a number, the type of the variable is **int**,

otherwise it is **bool**. For example, finding *HbA1c* can have value of 48, 53, and 58 made it an **int** variable. To mark if a drug is taken or not, for example in the phrase: "*standard-release metformin*", we set its type as **bool**.

Next, an invariant is generated for every condition found in the case frame (line 12). With respect to the invariants, a location is also generated for every action found in the case frame (line 13). The locations are built sequentially from the first sentence to the last following the sentence order information in NUM role. Whenever the verb *rises* is found in CAC role, its CTV value is used as the upper limit for the current location's invariant and as the lower limit for the transition's guard to the next location. Updates in transitions are done by setting the boolean variable from PAT to **true**.

Whenever the action *consider* is found in ACT role, some branches to the next location are generated as this keyword is used to express several drugs that can be given to a patient. Figure 6.4 below, taken from the T2D therapy algorithm, illustrates this phenomenon (highlighted in yellow):

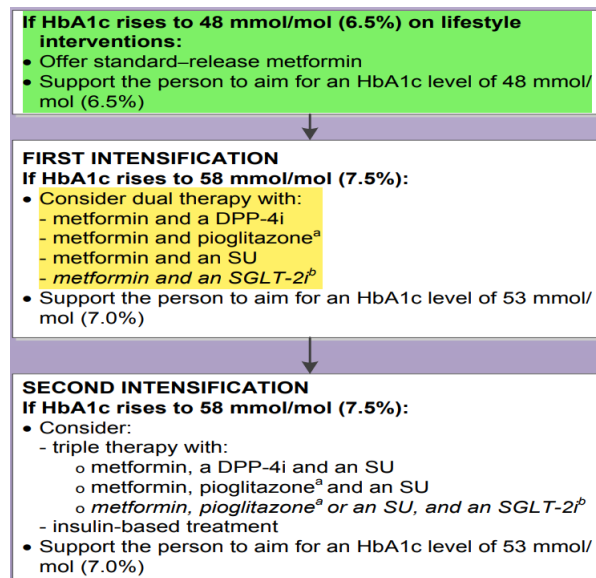


Figure 6.4: A text snippet from the T2D therapy algorithm

To model how *HbA1c* can change its value, a simple template that always increments the value of *HbA1c* is also created. Here, we assume  $40 \leq HbA1c \leq 60$  to capture the progression from being healthy to taking the first treatment (i.e.,  $48 \leq HbA1c$ ), and having the first intensification and the second intensification (i.e.,  $58 \leq HbA1c$ ).

**Transforming the T2D therapy algorithm** Figure 6.5 (p. 94) shows the original model generated automatically. In this model, we only show the therapy path where standard-release metformin is tolerated (marked by the first bullet point in Figure 6.4), the first intensification (the middle section in Figure 6.4) and the second intensification (the bottom section in Figure 6.4). The locations have a location invariant on the value of *HbA1c* to indicate that if *HbA1c* rises above a certain value (for instance above 48 in the location **Normal**) then the location must be left and a treatment given. The different transitions between intensifications show the different options available and are dependant on the first treatment received: the taken medications are captured by a boolean variable with the same name and the value is set as true.

When investigating the generated model, we noticed that there is no transition that takes the system (in this case a potential patient) back to the initial location **Normal**. As it may be more realistic to assume that other factors (such as changing life style habits and diet) can have an effect and recovery is possible, we want to refine the model to take this into account. This means that the model should cover a situation where the patient's condition turns back to normal after a time under treatment.

On the other hand, this may not be a very frequent outcome and ideally we would like to quantify the likelihood of such a recovery to happen as opposed to a deterioration of the condition as given in the model of Figure 6.5 (p. 94). This can be done with Probabilistic Timed Automata which extend TA with probabilistic transitions (Norman et al., 2013). The quantification of such transitions (probability values) can be done by analysing large datasets of records for patients with diabetes.



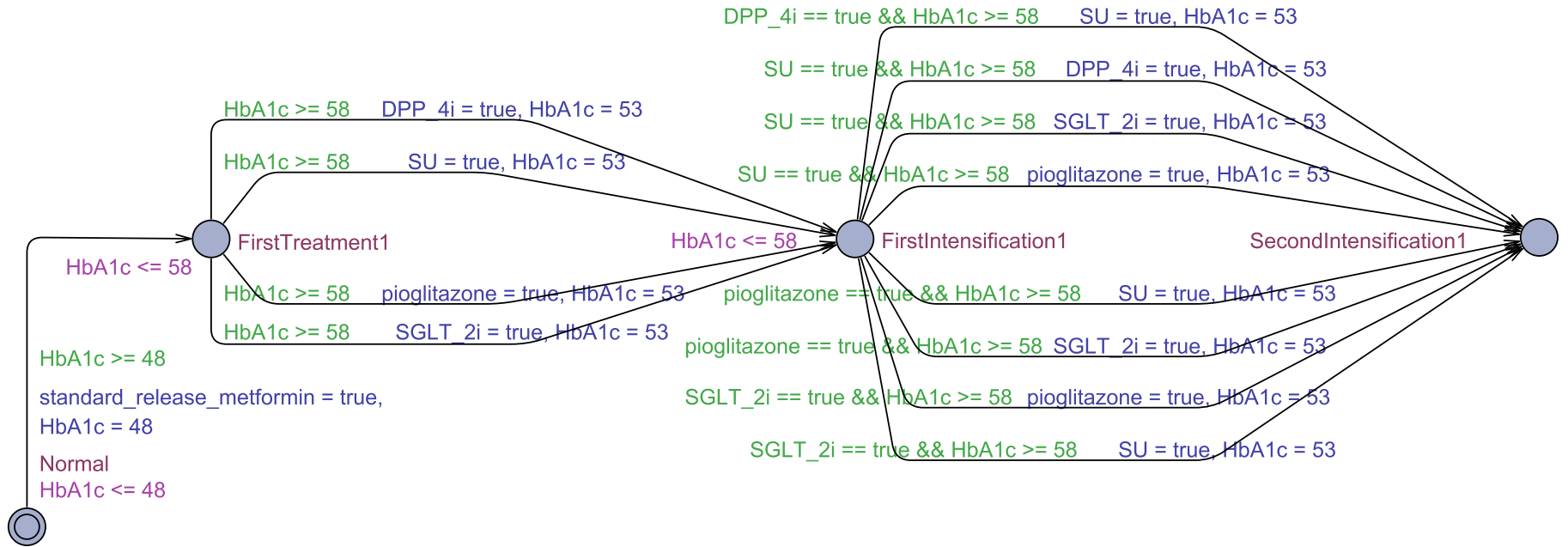


Figure 6.5: Generated model for adults with T2D that tolerate metformin

**Refining the T2D therapy algorithm model** We manually modify the model generation process to create some branch points after a treatment is taken. A branch is created whenever we find the verb *aim* denoting the situation when the doctor tries to stabilise the *HbA1c* level after giving a treatment. At present, we give a value of 20 and 80 as the weight to go back to a normal state and to the next (deteriorated) state respectively. We choose these two values by considering the likelihood that for a person with type 2 diabetes to get better is smaller in general than to get worse. In reality, these values can vary between each branching points, i.e., patients will have lower chance to get better in the second intensification stage than the first intensification.

Figure 6.6 (p. 96) shows the graphical model with branch points. To keep the model more readable, we currently only show going back to a normal state, but further possible transition branches include returning to any other previous treatment state with different weights. The real weights can be obtained by running statistical analysis on registry records of people who are admitted for having T2D. One way to do this is by checking the progression of taking one, two or three combination of drugs that marked the first and second intensification. We also add the first alternative path where modified-release metformin is used. This will create a mirror of locations and transitions which only differs on the guard for the first transition.

The actual model generation is done by generating an XML file which can then be visualised in the UPPAAL tool as in Figure 6.6 (p. 96) (after minor adjustments to take into account the visual placement of the elements). The XML file contains all the information of locations, location invariants, transitions and variables, but can also be given directly on the command line to the model checker for verification.

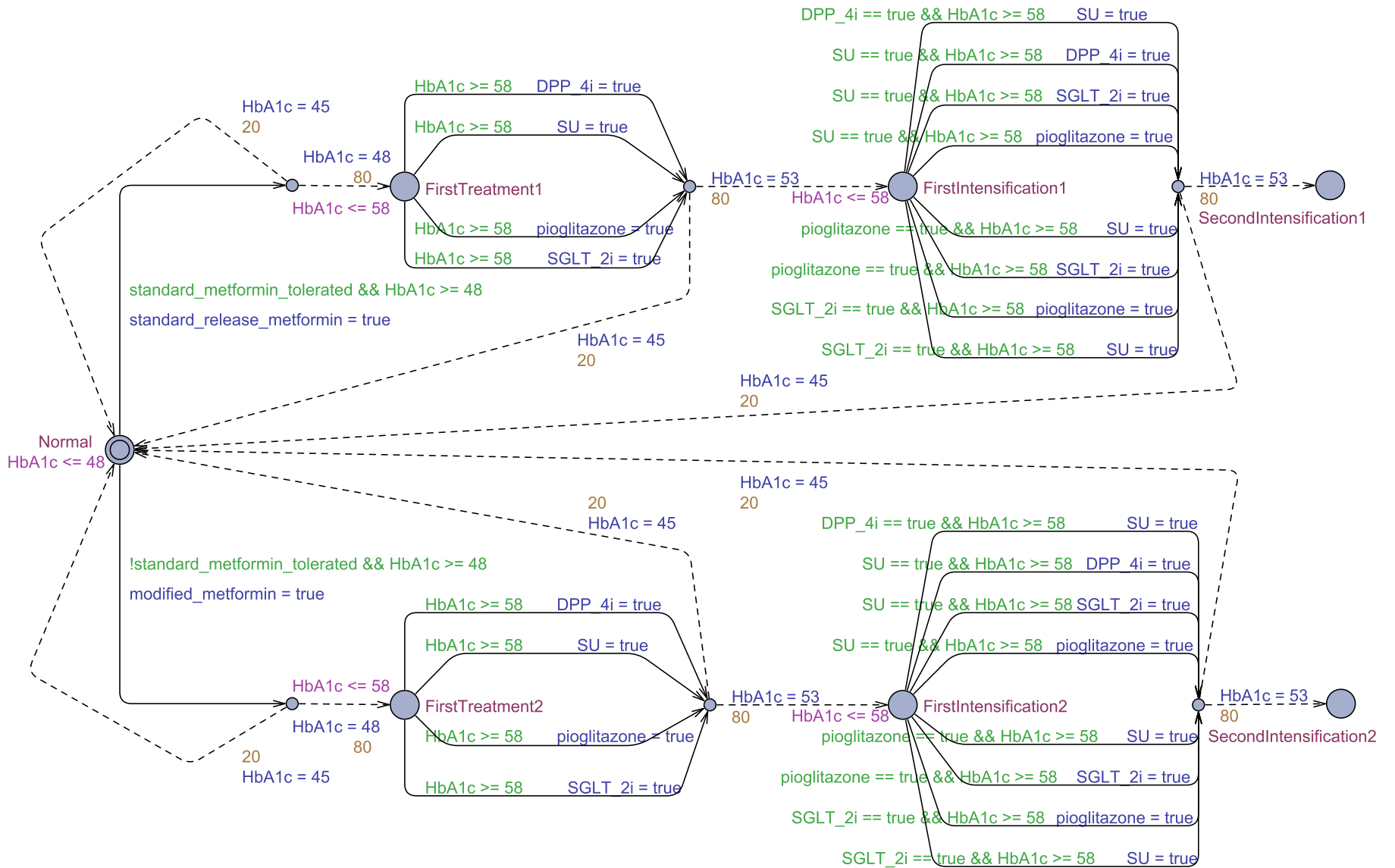


Figure 6.6: Model for adults with T2D that tolerate metformin modified with branching points.

### 6.2.1.2 Verification

We verify our model against some properties as shown in Table 6.3 (p. 98). The properties selected are used primarily to evaluate the approach. UPPAAL takes properties written in a restricted form of the timed CTL (TCTL; Henzinger et al., 1994). The properties were verified against the model in Figure 6.6 after an initial analysis of the original model and its refinement to include recovery.

The property

```
A[] !diab.SecondIntensification1 && !diab.SecondIntensification2
```

illustrates the situation where a second intensification of the therapy has been reached, either by taking standard release metformin or modified metformin, denoted by `diab.SecondIntensification1` and `diab.SecondIntensification2` respectively. The property itself only holds if and only if second intensification is never reached, and the fact that it is not satisfied will result in a trace that shows a second intensification of the therapy being reached.

Another similar property,  $A[] \text{!deadlock}^2$ , is not satisfied because the model at present and as generated from the therapy algorithm contains no further steps after second intensification and hence deadlocks at that point. This suggests that further discussions with clinicians are required to understand available options from that point and what is realistic. A self-loop on the find location can be added to avoid a deadlock scenario.

From the model in Figure 6.6, we can also see that there is a possibility of never reaching a first intensification or second intensification shown by verifying the formulae:

```
E<> !diab.FirstIntensification1 && !diab.FirstIntensification2
```

```
E<> !diab.SecondIntensification1 && !diab.SecondIntensification2
```

The model finds that these two properties are satisfiable. Again, these situations

---

<sup>2</sup>`deadlock` is a keyword in UPPAAL which can be used to verify any model to check for the absence of deadlocks.













UPPAAL Queries	Expected	Verification Result	Remark
$A[] \text{!deadlock}$			There is a path that leads to a deadlock.
$E \text{!diab.FirstIntensification1} \ \&\& \ \text{!diab.FirstIntensification2}$			There exists a path where first intensification is never reached.
$E \text{!diab.SecondIntensification1} \ \&\& \ \text{!diab.SecondIntensification2}$			There exists a path where second intensification is never reached.
$A[] \text{!diab.SecondIntensification1} \ \&\& \ \text{!diab.SecondIntensification2}$			There is a path that reaches a second intensification.
$\text{diab.FirstIntensification2} \rightarrow \text{diab.SecondIntensification2}$			There is a path in which a second intensification will never be reached from a first intensification.

Table 6.3: UPPAAL model verification result.  and  stand for *satisfied* and *not satisfied* respectively.

became possible because we added a scenario where the patient’s condition improves after getting some treatment.

Finally, the last property which is written in temporal implication notation

$$\text{diab.FirstIntensification2} \rightarrow \text{diab.SecondIntensification2}$$

also shows the possibility of never evolving to a second intensification after reaching the first intensification. The model finds the implication of this property to be unsatisfiable as we added branches for a patient to get better. In other words, there are possible paths in the model where even though a patient may at some point be at a stage of `FirstIntensification2`, the patient never progresses in their diabetes to reach the stage for a `SecondIntensification2`.

## 6.2.2 Transforming and verifying guidelines in PRISM

As we have seen in the previous section, when modelling clinical guidelines, we may also need to capture their probabilistic behaviour. For example, in Figure 6.6 (p. 96), we set 20-80 weights to model the probability of a patient getting better/worse after taking a drug prescription. Similarly to UPPAAL, PRISM is a probabilistic model checker for systems which can exhibit stochastic behaviour<sup>3</sup> (Kwiatkowska et al.,

<sup>3</sup><https://www.prismmodelchecker.org/>

2011). It supports discrete and continuous time models as well as deterministic or nondeterministic behaviour. Furthermore, PRISM can also be used for simulation and statistical model checking. For these reasons, and also for providing multimodel output from our framework, we also transform the T2D therapy algorithm into a PRISM specification.

To generate a PRISM model, we also traverse the case frame in a similar manner as described in the previous section. However, as PRISM requires probabilities when defining updates in a command, we use the statistical probability taken from the trends in diabetes medication in Scotland reported in (Greiver et al., 2021). Besides capturing the trends in drug use in the Scottish population between 2012 and 2017, it also shows the double and triple therapy drugs combination in Figure 6.7.

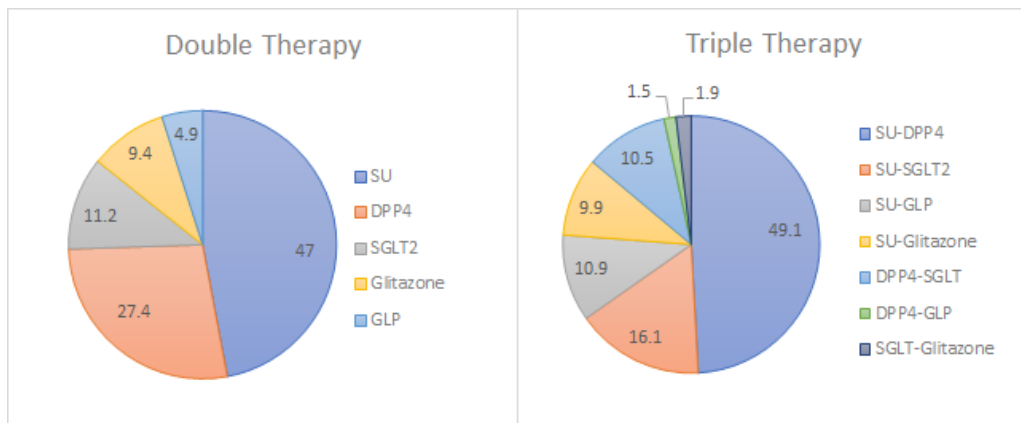


Figure 6.7: Percentage of drugs combination for double and triple therapy (Greiver et al., 2021)

### 6.2.2.1 Guidelines transformation

For T2D, we will use the probabilistic timed automata (PTA) model provided by PRISM, amongst many other models such as discrete-time Markov chains (DTMCs), Markov decision processes (MDPs), probabilistic automata (PAs), continuous-time Markov chains (CTMCs), and priced probabilistic timed automata (PPTAs) (Hartmanns and Hermanns, 2015). In PRISM, a model is composed of *modules* interacting with each other. Each module has its own *variables* whose values represent the state of the module. The state of all modules will determine the state of the whole model. A set of *commands* in the module is defined to describe its behaviour in the following form:

[action] guard  $\rightarrow$  prob\_1: update\_1 + ... + prob\_n: update\_n;

Each command comprises a guard to express the enabling condition and several updates with their respective probabilities. An *action* can be provided to a command for labelling purpose or for synchronisation with other modules.

The process for generating a PRISM model is illustrated by the `generate_model` function in Algorithm 3. This function will generate two modules, one for the drugs therapy and another one for modelling the HbA1c progress. Similar to UPPAAL, we define all variables that do not have numeric value as **bool** type, i.e., all drugs mentioned. However, to simplify the process as well as to make it clearer, we manually set the lower bound and the upper bound of *HbA1c* to the minimum and the maximum values it can have in accordance with the text. Similarly to the UPPAAL conversion, we also implement this algorithm in Python.

---

**Algorithm 3:** Algorithm for generating PRISM model

---

```
1 def generate_model(case_frames):
2     var_set = set()
3     treatment_branching = list()
4     actions = list()
5     conditions = list()
6     commands = list()
7     location_index = [0..]
8
9     for f in case_frames do
10        num = f.num
11        cpt, cac, cmd, ctv = f.cpt, f.cac, f.cmd, f.ctv
12        agt, act, pat, tov, cact = f.agt, f.act, f.pat, f.tov, f.cact
13
14        get_variables (var_set, cpt, pat)
15        set_values (var_set, ctv, tov)
16        make_commands (var_set, commands)
17        if act == 'consider' then
18            make_branching_condition (cpt, ctv, conditions)
19            make_treatment_branching (location_index, conditions, var_set,
20                                     treatment_branching)
21
22        therapy_module = build_therapy_module (var_set, actions, commands)
23        hba1c_module = build_hba1c_module (var_set, actions, conditions)
24        write_to_file (therapy_module, hba1c_module)
```

---

A further difference with UPPAAL's model is that we use an integer number to mark the location of the state. Whenever there is an update, such as taking a drug, the model updates its state by moving to the next location. The new location can have the same value as the current one (to mark continuation) or a greater one (to mark progression). Moreover, we add a new drug that does not exist in the original text, namely GLP, but is commonly prescribed according to the study (Greiver et al., 2021).

As PRISM specifications are textual, it is easier to edit PRISM models, as opposed to updating UPPAAL models which are given in XML. In addition to manually adding the new drug GLP, we also update the action name in command statements since the therapy module should synchronise with the HbA1c progression module. Initially, actions are named with integer denoting the location name. To make it easier for the reader to understand, we update it to reflect the situation when the command is executed, e.g., `first_treatment`, `first_intensification`, etc. Finally, we add some state labels in our model for facilitating the verification process. For example, to mark the states where a combination of two or three drugs is taken (in first and second intensification), we add labels `doubleMetSU`, `doubleMetDPP4`, or `tripleMetSUDPP4`. Our final PRISM model can be seen in Listing 6.1 (p. 102).



Listing 6.1: PRISM specification

```

1  pta
2
3  module therapy
4    o : [0..5];
5    x : clock;
6    metformin : bool init false;
7    su : bool init false;
8    dpp4 : bool init false;
9    pioglitazone : bool init false;
10   sgl2 : bool init false;
11   glp : bool init false;
12
13   [first_treatment] o=0 ->
14     (o'=1) & (metformin'=true);
15   [first_treatment_branching] o=1 ->
16     0.83:(o'=1) + 0.17:(o'=2);
17   [first_intensification] o=2 ->
18     0.42:(o'=3) & (su'=true) +
19     0.29:(o'=3) & (dpp4'=true) +
20     0.13:(o'=3) & (sglt2'=true) +
21     0.10:(o'=3) & (pioglitazone'=true) +
22     0.06:(o'=3) & (glp'=true);
23   [second_treatment_branching] o=3 ->
24     0.68:(o'=3) + 0.32:(o'=4);
25   [second_intensification] o=4 & su=true ->
26     0.51:(o'=5) & (dpp4'=true) +
27     0.23:(o'=5) & (sglt2'=true) +
28     0.14:(o'=5) & (glp'=true) +
29     0.12:(o'=5) & (pioglitazone'=true);
30   [second_intensification] o=4 & dpp4=true ->
31     0.7:(o'=5) & (su'=true) +
32     0.22:(o'=5) & (sglt2'=true) +
33     0.08:(o'=5) & (glp'=true);
34   [second_intensification] o=4 & sgl2=true ->
35     0.59:(o'=5) & (su'=true) +
36     0.34:(o'=5) & (dpp4'=true) +
37     0.07:(o'=5) & (pioglitazone'=true);
38   [second_intensification] o=4 & pioglitazone=true ->
39     0.83:(o'=5) & (su'=true) +
40     0.17:(o'=5) & (sglt2'=true);
41   [second_intensification] o=4 & glp=true ->
42     0.82:(o'=5) & (su'= true) +
43     0.18:(o'=5) & (dpp4'=true);
44  endmodule
45
46  module hba1c_unit
47    p : [0..5];
48    y : clock;
49    hba1c : [45..65];
50
51  invariant
52    (p=0 => hba1c<=48) &

```

```

53     (p=1 => hba1c<=58) &
54     (p=2 => hba1c<=58)
55     endinvariant
56
57     [first_treatment] p=0 -> (p'=1) & (hba1c'=48);
58     [first_intensification] p=1 -> (p'=2) & (hba1c'=53);
59     [second_intensification] p=2 -> (p'=3) & (hba1c'=53);
60     endmodule
61
62     label "normal" = o=0;
63     label "firstTreatment" = o=2;
64     label "firstIntensification" = o=3;
65     label "secondTreatment" = o=4;
66     label "secondIntensification" = o=5;
67     label "doubleMetSU" = o=3 & su=true;
68     label "doubleMetDPP4" = o=3 & dpp4=true;
69     label "doubleMetSGLT2" = o=3 & sgl2=true;
70     label "doubleMetPioglitazone" = o=3 & pioglitazone=true;
71     label "doubleMetGLP" = o=3 & glp=true;
72     label "tripleMetSUDPP4" = o=5 & su=true & dpp4=true;
73     label "tripleMetSUSGLT2" = o=5 & su=true & sgl2=true;
74     label "tripleMetSUGLP" = o=5 & su=true & glp=true;
75     label "tripleMetSUPioglitazone" = o=5 & su=true &
76         pioglitazone=true;
77     label "tripleMetDPP4SGLT2" = o=5 & dpp4=true & sgl2=true;
78     label "tripleMetDPP4GLP" = o=5 & dpp4=true & glp=true;
79     label "tripleMetSGLT2Pioglitazone" = o=5 & sgl2=true &
80         pioglitazone=true;
81

```

---

### 6.2.2.2 Verification

Similarly to verifying the UPPAAL model, we also verify the PRISM model against several properties in Table 6.4. These properties are also selected primarily for the purpose of checking our approach. It is worth noting that we also add the probability of staying in a state, e.g., the probability of not having first/second intensification.

Different from UPPAAL, this time we can ask the model for a specific probability of an event. For example, the property

$$P_{\max}=? \quad [F \text{"tripleMetSUDPP4"}]$$

asks the probability of having triple therapy Metformin, SU and DPP4. The probability given by PRISM is very close to the cumulative probability shown in Figure 6.7 (p. 99).

We have verified the PRISM model against all possible drugs combinations. Overall, we get the same values with our cumulative probability. There are several cases where the calculated probabilities do not match the cumulative probabilities. For example, the triple therapy Metformin-SU-Pioglitazone has 13% probability, Metformin-DPP4i-SGLT2i has 11% probability, and Metformin-DPP4i-GLP has 3% probability in contrast to 11%, 13%, and 5% respectively. We believe that this difference is a result from a rounding error when we put the probabilities in our model taken from Figure 6.7 (p. 99).

Properties	Result	Remark
Pmax=? [F "doubleMetSU"]	0.4199	The probability of double therapy Metformin-SU
Pmax=? [F "doubleMetDPP4"]	0.2899	The probability of double therapy Metformin-DPP4i
Pmax=? [F "tripleMetSUDPP4"]	0.4171	The probability of triple therapy Metformin-SU-DPP4i
Pmax=? [F "tripleMetSUSGLT2"]	0.1732	The probability of triple therapy Metformin-SU-SGLT2i
Pmax=? [F "tripleMetDPP4SGLT2"]	0.1079	The probability of triple therapy Metformin-DPP4i-SGLT2i

Table 6.4: PRISM model verification result

### 6.2.3 Transforming and verifying guidelines in Z3

Z3 is a Satisfiability Modulo Theory (SMT) solver created by Microsoft and which can be used for deciding if a formula is satisfiable under first-order theories<sup>4</sup> (Moura and Bjørner, 2008). Besides Z3, there are other constraint solver such as Conjure (Frisch, Jefferson et al., 2005; Akgun et al., 2011), MiniZinc (Nethercote et al., 2007), Essence (Frisch, Harvey et al., 2008), CVC4 (Barrett et al., 2011) and Yices (Dutertre, 2014). They differ mainly in terms of the built-in first-order theories that they support, how straightforward it is to specify the constraint problems as well as their performance when solving problems. The decision to use Z3 is mainly because we are quite familiar with it compared to other solvers.

<sup>4</sup><http://research.microsoft.com/projects/z3>

A formula is satisfiable if there are assignments of values for its variables that make it true. An SMT solver is used in broad applications such as software tests using bounded model checking (Clarke et al., 2004), directed automated random testing (DART; Godefroid et al., 2005), or automatic exploit generation (AEG; Avgerinos et al., 2014); as network security tests for finding protocol manipulation attacks (Kothari et al., 2011), checking access control for web applications (Ghotbi and Fischer, 2012), or checking data flow security (Son et al., 2013); and as resource scheduling and optimisation (Erkinger, 2013; Nieuwenhuis and Oliveras, 2006).

Despite Z3's main applications being static checking, test case generation, and predicate abstraction, we can also use it to prove any model that can be supported in first-order logic. As it gains more users, Z3 now provides bindings to another programming language, therefore it can be programmed through scripts. Some supported languages are .NET, C, C++, Java, OCaml and Python. As the previous cases, we use Python in our implementation.

We use Z3 for modelling guidelines that do not have clear ordering, in other words, where we do not have a clear process or model to describe treatment options for a disease. This is the case when following the recommendations for chronic kidney disease. We transform every concept into Z3 sorts (or variables) and add properties as assertions to check the satisfiability of the whole specification.

$$\begin{aligned}
 & \text{○} + \text{○} = 10 \\
 & \text{○} \times \text{□} + \text{□} = 12 \\
 & \text{○} \times \text{□} - \text{△} \times \text{○} = \text{○} \\
 & \text{△} = ?
 \end{aligned}$$

Figure 6.8: A simple system equation

Listing 6.2: Z3 example

---

```

1(declare-fun circle () Int)
2(declare-fun square () Int)
3(declare-fun triangle () Int)
4(assert (= (* 2 circle) 10))
5(assert (= (+ (* circle square) square) 12))
6(assert (= (- (* circle square) (* triangle circle)) circle))
7(check-sat)
8(get-model)

```

---

Figure 6.8 shows an example of a simple system of equations involving three variables<sup>5</sup>. Listing 6.2 shows how the equations are captured as formulas in Z3. Running the Z3 code in Listing 6.2, we check that the problem is satisfiable (line 7) and generate a solution (line 8). The output is shown in Listing 6.3. Here, the obtained solution (model) finds the assignment of 1, 2 and 5 for triangle, square, and circle, respectively. When trying to find valid assignments for the given variables, an SMT solver generates one solution (if it exists), though more than one solution may be available.

---

Listing 6.3: Z3 output

---

```
1 sat
2 (model
3  (define-fun triangle () Int 1)
4  (define-fun square () Int 2)
5  (define-fun circle () Int 5)
6 )
```

---

### 6.2.3.1 Guideline transformation

To illustrate the transformation process, we take these sentences from the Chronic Kidney Disease (CKD) guideline<sup>6</sup>:

1. "Offer a low-cost renin-angiotensin-aldosterone system antagonist to people with CKD and diabetes and an ACR of 3 mg/mmol or more (ACR category A2 or A3)."
2. "Offer a low-cost renin-angiotensin-aldosterone system antagonist to people with CKD and hypertension and an ACR of 30 mg/mmol or more (ACR category A3)."
3. "Offer a low-cost renin-angiotensin-aldosterone system antagonist to people with CKD and an ACR of 70 mg/mmol or more (irrespective of hypertension or cardiovascular disease)."

We pass these sentences into the NER and relation classification models to get the entities and their relationships. From the list of entities and relationships in every

---

<sup>5</sup>[https://sat-smt.codes/SAT\\_SMT\\_by\\_example.pdf](https://sat-smt.codes/SAT_SMT_by_example.pdf)

<sup>6</sup><https://www.nice.org.uk/guidance/cg182/resources/algorithms-pdf-498987181>

sentence, we create triples in the form  $[\text{entity}_1, \text{relationship}, \text{entity}_2]$ . The triples below are formed from the first sentence above:

```

["Offer", "acts-on(e1,e2)", "system antagonist"]

["low-cost renin-angiotensin-aldosterone", "modifier(e1,e2)", "system antagonist"]

["Offer", "receiver(e2,e1)", "people"]

["people", "owner(e1,e2)", "CKD"]

["people", "owner(e1,e2)", "diabetes"]

["CKD", "joint", "diabetes"]

["people", "owner(e1,e2)", "ACR"]

["diabetes", "joint", "ACR"]

["ACR", "other", "3 mg/mmol"]

["3 mg/mmol", "modifier(e2,e1)", "more"]

```

From these triples, we apply the function `generate_specification` in Algorithm 4 for transforming the triples into a Z3 specification. We use the `z3-solver`<sup>7</sup> library to help run the specification directly from Python.

By passing the list of triples in a sentence, we construct components such as the premise, conclusions, and all sorts (or variables) to create the specification. We do this until all sentences have been processed. The last step is to add an implication construct in Z3 by passing the premise and conclusion. Listing 6.4 shows the output for processing the first in the CKD guideline.

Listing 6.4: Z3 specification for sentence 1 in the CKD guideline

---

```

1(declare-fun low-cost_renin-angiotensin-aldosterone_system_antagonist () Bool)
2(declare-fun diabetes () Bool)
3(declare-fun ckd () Bool)
4(declare-fun acr () Bool)
5(assert (=> (and acr ckd diabetes) low-cost_renin-angiotensin-
  aldosterone_system_antagonist))

```

---

<sup>7</sup><https://github.com/Z3Prover/z3>

---

**Algorithm 4:** Algorithm for generating Z3 model

---

```
1 def construct_components(triples):
2     owner = get_owner(triples)
3     premise = get_premise(triples, owner)
4     choice = get_compound(triples, 'or')
5     joint = get_compound(triples, 'and')
6     all_sort = get_sort(triples, premise, choice, joint)
7     conclusion = set(all_sort) - set(premise)
8
9     return premise, conclusion, all_sort
10
11 def generate_specification(sentences):
12     premise, conclusion, all_sort = list(), list(), list()
13     solv = Solver()
14
15     for s in sentences do
16         premise_, conclusion_, all_sort_ = construct_components(s.triples)
17         premise.append(premise_)
18         conclusion.append(conclusion_)
19         all_sort.append(all_sort_)
20
21     p_clause = create_z3_clause(premise)
22     c_clause = create_z3_clause(conclusion)
23     solv.add(Implies(p_clause, c_clause))
24     solv.check()
25     solv.model()
26     solv.sexpr()
```

---

There are some points worth to note from Listing 6.4. First, we lower case all words and replace the space between compound words with a dash. For example, "*low cost*" and "*renin angiotensin aldosterone*" will become "*low-cost*" and "*renin-angiotensin-aldosterone*". Next, a modifier and a modifiee are connected into a single sort by underscore. Thus, the modifier "*low-cost renin-angiotensin-aldosterone*" and the modifiee "*system antagonist*" in ["*low-cost renin-angiotensin-aldosterone*", "modifier(e1,e2)", "*system antagonist*"] will produce a sort named "*low-cost\_renin-angiotensin-aldosterone\_system\_antagonist*".

Next, we cannot capture negation in the relationship. Our learning model can capture the ownership between *people* and *CKD*, *diabetes* or *ACR*. However, it is still

the case even if the preposition *with* changes to *without*. Knowing the absence of a sort, we could have added the assertion as `false` for that sort in the Z3 specification.

Lastly, we can see that the transformation assigns *ACR* as a boolean sort instead of integer and does not assert its value to be greater than or equal to 3 (mg/mmol). This is due to currently we only have the `other` relationship to capture the relationship between *ACR* and its value. As can be seen from the list of triples above, the important triples for *ACR* in the sentence are ["*ACR*", "`other`", "3 mg/mmol"] and ["3 mg/mmol", "`modifier(e2,e1)`", "`more`"]. Due to the relationships' genericness, none of it can be used to infer the sort of *ACR*. To get a better result, we need to have more representative relationships than those shown in Table 5.5 (p. 69), with the caveat that the number of instances should be significant for the model to learn. Listing 6.5 shows the final specification for sentence 1 after manual revision.

Listing 6.5: Z3 specification for sentence 1 in the CKD guideline (revised)

---

```

1(declare-fun low-cost_renin-angiotensin-aldosterone_system_antagonist () Bool)
2(declare-fun diabetes () Bool)
3(declare-fun ckd () Bool)
4(declare-fun acr () Int)
5(assert (=> (and (>= acr 3) ckd diabetes) low-cost_renin-angiotensin-
  aldosterone_system_antagonist))

```

---

### 6.2.3.2 Verification

For the verification of Z3 specification taken from the CKD guideline, we do it incrementally to see the output of the SMT solver adding one sentence at a time.

Listing 6.6: Z3 output for sentence 1 in the CKD guideline

---

```

1sat
2(model
3  (define-fun low-cost_renin-angiotensin-aldosterone_system_antagonist () Bool
4    false)
5  (define-fun acr () Int
6    3)
7  (define-fun diabetes () Bool
8    true)
9  (define-fun ckd () Bool
10   false)
11)

```

---

Listing 6.6 shows the output for checking the satisfiability of Listing 6.5 and printing the model. For this particular specification, one values assignment that Z3 found to



make the whole formula satisfiable is 3 for *ACR*, true for *Diabetes*, false for *CKD* and false for "*low-cost renin angiotensin aldosterone system antagonist*". Obviously, there are many other assignments where the formula is still satisfiable and we can ask Z3 to output them one by one.

Listing 6.7: Z3 specification for sentence 1, 2 and 3 in the CKD guideline

---

```

1 (declare-fun low-cost_renin-angiotensin-aldosterone_system_antagonist () Bool)
2 (declare-fun acr () Int)
3 (declare-fun diabetes () Bool)
4 (declare-fun ckd () Bool)
5 (declare-fun hypertension () Bool)
6 (assert (=> (and ckd diabetes (>= acr 3)) low-cost_renin-angiotensin-
  aldosterone_system_antagonist))
7 (assert (=> (and ckd hypertension (>= acr 30)) low-cost_renin-angiotensin-
  aldosterone_system_antagonist))
8 (assert (=> (and ckd (>= acr 70)) low-cost_renin-angiotensin-
  aldosterone_system_antagonist))

```

---

Finally, Listing 6.7 combines all specifications from the three sentences in the CKD guideline. When we ask Z3 to find if satisfiability and the model (if it exists), one of them is shown in Listing 6.8.

Listing 6.8: Z3 output for sentence 1, 2 and 3 in the CKD guideline

---

```

1 sat
2 (model
3 (define-fun low-cost_renin-angiotensin-aldosterone_system_antagonist () Bool
4   false)
5 (define-fun acr () Int
6   0)
7 (define-fun hypertension () Bool
8   true)
9 (define-fun diabetes () Bool
10  true)
11 (define-fun ckd () Bool
12  true)
13)

```

---

Using Z3, we can keep adding more statements in the form of assertions and ask Z3 to provide the model configuration where everything holds. This is suitable in a situation where we want to combine sentences from different guidelines in addition to particular information from a patient's condition.

## 6.3 Summary

In this chapter, we described the process of generating formal specifications and verifying them against some properties. We briefly explained the logical forms that we used to specify the properties. We then described the process to convert guidelines into formal specifications accepted by tools such as UPPAAL, PRISM and Z3. The choice of the verification tool will depend on the guideline text itself and how much information is available.

The transformation still runs in semi-automatic way as we need to include more realistic situations in UPPAAL, add probabilities from extracted patient data in PRISM, and fix problems with variable types in Z3 due to the limitation of the annotation tags. Finally, we run some verification checks as a proof of concept that the models we generate are correct for our current task at hand.

*In other words, apart from the known and the unknown, what else is there?*

Harold Pinter

# 7

## Conclusions

Clinical guidelines are often written in human natural language therefore they are also often ambiguous and/or incomplete. For example, a guideline on blood glucose lowering for people with type 2 diabetes might have missing recommendations for when patients improve, or might suggest the use of a drug or a combination of drugs without precise dosage information. This thesis looked into this problem by contributing with an approach that allows us to consider clinical guidelines and patient information in natural language, and how to convert that into machine readable information for automated reasoning. To do so, we address two big challenges: (1) how to extract the main concepts from guideline texts and (2) how to convert guidelines into formal specifications.

One of our main goals is to automatically get the key concepts contained in clinical guidelines. We performed our study from various domains, such as automatic test case generation in software engineering using Controlled Natural Language (CNL) and Semantic Role Labelling (SRL) techniques, as well as medical entities annotation in biomedical corpus using dictionary lookup, rule-based or machine learning.

To address the first challenge, we followed two approaches. The first approach to annotate the key concepts for clinical process uses CNL. We showed that by writing

clinical guidelines in accordance to our handcrafted rules, it is easier to get the main information from the text. Defining a domain-specific CNL makes the guideline representation as close as possible as human natural language although it adds more restrictions on the sentence structure. This approach is also suitable when the dataset size is not very large. However, we note that a CNL is not easy to maintain as it is difficult to have a grammar that can accommodate all possible sentence structures.

A second alternative approach looks into the possibility of annotating the key concepts in clinical guidelines without restricting the sentence structure. This was done by creating machine learning models on annotated clinical guideline corpus using SRL, Named-Entity Recognition (NER) and relation classification with bidirectional Long Short-Term Memory (LSTM). We also discussed fine-tuning state-of-the-art neural networks models for the NER and relation classification tasks to compare the results with our baseline models. Using the approach that we introduced here, we minimise the amount of restriction imposed to writing guideline text and give more possibilities to deal with much larger corpus containing sentences with various structures.

We conducted several experiments and evaluations for our machine learning models. We found that, in general, fine-tuned models using Bidirectional Encoder Representations from Transformers (BERT) and its variants perform better than the baselines. However, the trends also show that all models tend to make mistakes on similar issues. We discussed several factors that might explain this phenomenon: (1) the small amount of sentences in our dataset, (2) the class imbalance in our dataset, and (3) possible mistakes in the annotation. Nevertheless, we also found evidence where the models can learn better to mitigate such annotation mistakes. We also explored several methods to improve the performance of an ML-based annotation approach. Comparing the result of the baseline model with the fine-tuned ones, we recognise a trade-off between time and precision level. Indeed, faster approach may be less precise when compared with slower alternatives.

To address our second challenge, we showed how to transform guidelines into several formal specifications for formal verification. This transformation is done from the marked concepts using CNL and machine learning models. Although we can generate

the formal models and verify some properties, we also found that our current machine learning annotations are quite generic and are partially incomplete therefore some manual edits are still needed prior to formalisation. For example, we still cannot capture the comparative case in the text, e.g., *ACR of 3 mg/mmol or more*. Our current machine learning models also lack the ability to identify negation or an absence of an event, e.g., *for patient without diabetes*" vs. *for patient with diabetes*". We believe that having a bigger dataset to enable more annotation concepts might address these issues with the caveat that the dataset should have balance classes between positive and negative instances. Our current dataset has very few instances of negation, hence the trained model will be biased to mark most concepts as positive.

## 7.1 Key Contributions and Findings

There are several contributions and findings in this thesis:

- \* We have learnt to annotate clinical guidelines with process labels. Having the annotations, we can use them further to formally verify the guidelines.
- ➔ The approach to address the annotation can range from creating grammar rules to learning from data. The former one is suitable when the text structure is less varied and the dataset size is small. For a more scalable strategy, with the support of adequate dataset size, the latter approach is preferable. With advances in Natural Language Processing (NLP), machine learning and computation power, many options can be used, including creating simple models manually, or fine-tuning some existing state-of-the-art models.
- ➔ As humans are more dependent of computer software/hardware, tools and techniques for verification have increasingly become more developed in recent years. In the context of verifying clinical guidelines, we have the options to either use a model checker or a Satisfiability Modulo Theory (SMT) solver. The decision largely depends on the representation of the system that we want to check. For example, for dealing with a state-based system possibly with stochastic and/or timed information, we may be able

to explore the use of model checkers such as UPPAAL or PRISM. For the case where a model is not available and the system can be specified as a set of constraints that need to be fulfilled, SMT solvers such as Z3 may be a better option. We note that many other model checkers and constraint solvers exist and could have been used instead.

\* The whole approach in this thesis is still semi-automatic, from representing the guidelines sentences to generating the formal specifications and properties. To achieve a (near) fully automatic process, we suggest to have a bigger dataset (as it is the case in machine learning) since we can have more refined and less generic annotations, which help the transformation into formal specifications more smoothly.

➔ Although this thesis domain is clinical guidelines, the approach that we presented can also be applied in other domains. To do so, a set of new lexicons (and maybe grammar rules) needs to be defined following the CNL approach. In terms of a machine learning approach, new models need to be created or fine-tuned as our current models only learnt from clinical guideline texts, hence they cannot associate the tags to words in different domain (e.g., tagging *user profile* as `Object` in a software requirement domain). From here to the conversion into formal specification, the steps would also need small adjustments to handle new keywords for the new domain. The same is also true for the machine learning outputs: new tags will add new triplet rules, hence some adjustments are needed as well.

\* We have created a small clinical guidelines dataset annotated with action process and relationships. This dataset can be used further as a comparative study for developing new and better techniques<sup>1</sup>.

## 7.2 Future Work

There are many possible directions for future work. Firstly, we have developed some models to annotate action process in clinical guidelines. These annotations can

---

<sup>1</sup>The dataset can be found on the repository in this url: <https://github.com/rah-man/ClinicalGuideline>

still be further developed by adding more complex entities and relationships. For example, our current entities still lack the notion of time and quantity as presented by Miksch et al. (1997) to name but a few. In terms of relationships, the concept of negation and modality can also be introduced. For the former, a rule-based approach such as NegEx (W. Chapman et al., 2001; Mehrabi et al., 2015) or NegBio (Peng et al., 2018) or machine learning based method such as NegTool (Enger et al., 2017) can be considered. We can also consider to do multitask-learning, i.e., training several tasks together in a single model, to do NER and finding negation as is done by Grivas et al. (2020). In terms of modality extraction, some research has been done on the use of deontic modalities in the field of legal text (Dragoni et al., 2016; Camilleri and Schneider, 2017).

When verifying the guidelines in formal specification, it is also useful if the properties to be checked can be given in natural language. Some notable work and study to achieve this for example translating natural language into temporal logic (Dzifcak et al., 2009; Brunello et al., 2019) or by using combinatory categorial grammar (CCG; Zettlemoyer and Collins, 2009; Kwiatkowski et al., 2010; Matuszek et al., 2013). In the opposite direction, the output from a model checker or an SMT solver like Z3 would also be more useful if translated to human language. One example of the work towards the interpretability towards medical explainable artificial intelligence (Medical XAI; Tjoa and Guan, 2019) is generating treatment plans from Z3 output by Shaheen et al. (2020) and Shaheen et al. (2021).

In terms of the dataset used, the work in this thesis still requires more data to be more comparable and useful overall. Having more annotators and a clear annotation guideline would be one step to improve our current data quality that still has potential noise. A bigger dataset also means more refined annotations that can be made to help the guidelines conversion into formal specification.

Our current triples format can also be transformed into Resource Description Framework (RDF) format by adding more concepts to link the current ones. For example, having the `Drug` concept, we can create an association to say that a metformin `Ob-`

ject is also a type of Drug. This can be converted into a knowledge graph<sup>2</sup> that allows us to make information extraction from the data.

Lastly, we also need to present this work to practitioners to get their invaluable feedback. This also opens the possibility to integrate our work into a clinical decision support system that will allow for the formulation of a variety of guidelines for treatment of (chronic) condition as patient-specific information. Ideally, this should be done through collaboration with clinicians and domain experts.

---

<sup>2</sup><https://www.ontotext.com/knowledgehub/fundamentals/what-is-a-knowledge-graph/>



# Bibliography

- Akbik, Alan, Duncan Blythe and Roland Vollgraf (2018). ‘Contextual String Embeddings for Sequence Labeling’. In: *COLING 2018, 27th International Conference on Computational Linguistics*, pp. 1638–1649 (cit. on p. 52).
- Akgun, Ozgur, Ian Miguel, Chris Jefferson, Alan M Frisch and Brahim Hnich (2011). ‘Extensible automated constraint modelling’. In: *Twenty-Fifth AAAI Conference on Artificial Intelligence* (cit. on p. 104).
- Alex, Beatrice, Claire Grover, Richard Tobin, Cathie Sudlow, Grant Mair and William Whiteley (2019). ‘Text mining brain imaging reports’. In: *Journal of biomedical semantics* 10.1, pp. 1–11 (cit. on p. 19).
- Alsentzer, Emily, John R Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann and Matthew McDermott (June 2019). ‘Publicly available clinical BERT embeddings’. In: pp. 72–78. DOI: [10.18653/v1/W19-1909](https://doi.org/10.18653/v1/W19-1909). URL: <https://www.aclweb.org/anthology/W19-1909> (cit. on p. 63).
- Alur, Rajeev and David L Dill (1994). ‘A theory of timed automata’. In: *Theoretical computer science* 126.2, pp. 183–235 (cit. on p. 90).
- Aronson, Alan R (2006). ‘Metamap: Mapping text to the umls metathesaurus’. In: *Bethesda, MD: NLM, NIH, DHHS* 1, p. 26 (cit. on pp. 18, 19).
- Avgerinos, Thanassis, Sang Kil Cha, Alexandre Rebert, Edward J Schwartz, Maverick Woo and David Brumley (2014). ‘Automatic exploit generation’. In: *Communications of the ACM* 57.2, pp. 74–84 (cit. on p. 105).
- Baier, Christel and Joost-Pieter Katoen (2008). *Principles of Model Checking (Representation and Mind Series)*. The MIT Press. ISBN: 026202649X (cit. on pp. 86, 87, 89).
- Barrett, Clark W., Christopher L. Conway, Morgan Deters, Liana Hadarean, Dejan Jovanovic, Tim King, Andrew Reynolds and Cesare Tinelli (2011). ‘CVC4’. In: *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*. Ed. by Ganesh Gopalakrishnan and Shaz Qadeer. Vol. 6806. Lecture Notes in Computer Science. Springer, pp. 171–177. DOI: [10.1007/978-3-642-22110-1\\_14](https://doi.org/10.1007/978-3-642-22110-1_14). URL: [https://doi.org/10.1007/978-3-642-22110-1\\_14](https://doi.org/10.1007/978-3-642-22110-1_14) (cit. on p. 104).

- Batista, Gustavo E. A. P. A., Ronaldo C. Prati and Maria Carolina Monard (June 2004). ‘A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data’. In: *SIGKDD Explorations Newsletter* 6.1, pp. 20–29. ISSN: 1931-0145. DOI: [10.1145/1007730.1007735](https://doi.org/10.1145/1007730.1007735). URL: <https://doi.org/10.1145/1007730.1007735> (cit. on p. 81).
- Bäumler, Simon, Michael Balsler, Andriy Dunets, Wolfgang Reif and Jonathan Schmitt (2006). ‘Verification of Medical Guidelines by Model Checking – A Case Study’. In: *Model Checking Software*. Ed. by Antti Valmari. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 219–233. ISBN: 978-3-540-33103-2 (cit. on pp. 17, 37).
- Beck, Kent (1999). ‘Embracing change with extreme programming’. In: *Computer* 32.10, pp. 70–77 (cit. on p. 7).
- Becker, Matthias and Britta Böckmann (2017a). ‘Personalized Guideline-Based Treatment Recommendations Using Natural Language Processing Techniques’. In: *Studies in Health Technology and Informatics* 235, pp. 271–275. ISSN: 1879-8365. DOI: [10.3233/978-1-61499-753-5-271](https://pubmed.ncbi.nlm.nih.gov/28423796). URL: <https://pubmed.ncbi.nlm.nih.gov/28423796> (cit. on pp. 19, 20).
- (2017b). ‘Semi-Automatic Mark-Up and UMLS Annotation of Clinical Guidelines’. In: *Studies in Health Technology and Informatics* 245, pp. 294–297. ISSN: 1879-8365. URL: <https://pubmed.ncbi.nlm.nih.gov/29295102> (cit. on pp. 19, 20).
- Becker, Matthias, Britta Böckmann, Karl-Heinz Jöckel, Martin Stuschke, Andreas Paul, Stefan Kasper and Isabel Virchow (Mar. 2020). ‘Mapping Patient Data to Colorectal Cancer Clinical Algorithms for Personalized Guideline-Based Treatment’. In: *Applied Clinical Informatics* 11.2, pp. 200–209. ISSN: 1869-0327. DOI: [10.1055/s-0040-1705105](https://doi.org/10.1055/s-0040-1705105) (cit. on pp. 19, 20).
- Behrmann, Gerd, Alexandre David and Kim G Larsen (2004). ‘A tutorial on uppaal’. In: *Formal methods for the design of real-time systems*, pp. 200–236 (cit. on pp. 5, 90).
- Bengio, Yoshua, Jérôme Louradour, Ronan Collobert and Jason Weston (2009). ‘Curriculum Learning’. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML ’09. Montreal, Quebec, Canada: Association for Computing Machinery, pp. 41–48. ISBN: 9781605585161. DOI: [10.1145/1553374.1553380](https://doi.org/10.1145/1553374.1553380) (cit. on p. 49).
- Beyer, Dirk (2020). ‘Advances in Automatic Software Verification: SV-COMP 2020’. In: *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by Armin Biere and David Parker. Springer International Publishing, pp. 347–367. ISBN: 978-3-030-45237-7 (cit. on p. 8).
- Björkelund, Anders, Bernd Bohnet, Love Hafdel and Pierre Nugues (2010). ‘A High-Performance Syntactic and Semantic Dependency Parser’. In: *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*

- tions. COLING '10. Beijing, China: Association for Computational Linguistics, pp. 33–36. URL: <https://www.aclweb.org/anthology/C10-3009.pdf> (cit. on p. 15).
- Björkelund, Anders, Love Hafdel and Pierre Nugues (June 2009). ‘Multilingual Semantic Role Labeling’. In: *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*. Boulder, Colorado: Association for Computational Linguistics, pp. 43–48. URL: <https://www.aclweb.org/anthology/W09-1206> (cit. on p. 45).
- Booch, Grady, James Rumbaugh and Ivar Jacobson (2005). *Unified Modeling Language User Guide, The (2nd Edition) (Addison-Wesley Object Technology Series)*. Addison-Wesley Professional. ISBN: 0321267974 (cit. on p. 7).
- Borchert, Florian, Christina Lohr, Luise Modersohn, Thomas Langer, Markus Follmann, Jan-Philipp Sachs, Udo Hahn and Matthieu-P. Schapranow (Nov. 2020). ‘GGPONC: A Corpus of German Medical Text with Rich Metadata Based on Clinical Practice Guidelines’. In: *Proceedings of the 11th International Workshop on Health Text Mining and Information Analysis*. Association for Computational Linguistics, pp. 38–48. DOI: [10.18653/v1/2020.louhi-1.5](https://doi.org/10.18653/v1/2020.louhi-1.5). URL: <https://aclanthology.org/2020.louhi-1.5> (cit. on p. 20).
- Boyce, Lee (2012). *Can I force NatWest or RBS to cover late payment penalties or extra costs I get hit with because of its banking meltdown?* URL: <https://web.archive.org/web/20200618232317/https://www.thisismoney.co.uk/money/saving/article-2163238/Do-NatWest--RBS-cover-late-payment-penalties-extra-costs-caused-banking-meltdown.html> (cit. on p. 8).
- Brady, Fiorenza (2013). *Cambridge University Study States Software Bugs Cost Economy \$312 Bill*. URL: <https://web.archive.org/web/20200618221438/https://www.prweb.com/releases/2013/1/prweb10298185.htm> (cit. on p. 8).
- Brechner, Eric (2015). *Agile Project Management with Kanban*. Best practices. Microsoft Press. ISBN: 9780735698956 (cit. on p. 7).
- Brunello, Andrea, Angelo Montanari and Mark Reynolds (2019). ‘Synthesis of LTL formulas from natural language texts: State of the art and research directions’. In: *26th International Symposium on Temporal Representation and Reasoning (TIME 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (cit. on p. 116).
- Bui, Quoc-Chinh, Sophia Katrenko and Peter M. A. Slood (Nov. 2010). ‘A hybrid approach to extract protein–protein interactions’. In: *Bioinformatics* 27.2, pp. 259–265. DOI: [10.1093/bioinformatics/btq620](https://doi.org/10.1093/bioinformatics/btq620). URL: <https://doi.org/10.1093/bioinformatics/btq620> (cit. on p. 19).
- Camilleri, John J and Gerardo Schneider (2017). ‘Modelling and analysis of normative documents’. In: *Journal of logical and algebraic methods in programming* 91, pp. 33–59 (cit. on p. 116).

- Carretta Zamborlini, Veruska, Rinke Hoekstra, Marcos Da Silveira, Cedric Pruski, A.C.M. ten Teije and F.A.H. van Harmelen (2016). ‘Inferring recommendation interactions in clinical guidelines’. In: *Semantic Web 7.4*, pp. 421–446. ISSN: 1570-0844. DOI: [10.3233/SW-150212](https://doi.org/10.3233/SW-150212) (cit. on pp. 20, 21).
- Carvalho, Gustavo Henrique Porto de (May 2016). ‘NAT2TEST: Generating Test Cases from Natural Language Requirements based on CSP’. PhD thesis. Universidade Federal de Pernambuco (cit. on pp. 9, 11–13, 22, 23, 27, 34, 37).
- Chapman, Wendy, Will Bridewell, Paul Hanbury, Gregory F Cooper and Bruce G Buchanan (2001). ‘A simple algorithm for identifying negated findings and diseases in discharge summaries’. In: *Journal of biomedical informatics* 34.5, pp. 301–310 (cit. on p. 116).
- Chinchor, Nancy and Beth Sundheim (1993). ‘MUC-5 Evaluation Metrics’. In: *Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore, Maryland, August 25-27, 1993*. URL: <https://www.aclweb.org/anthology/M93-1007> (cit. on p. 73).
- Clarke, Edmund, Daniel Kroening and Flavio Lerda (2004). ‘A Tool for Checking ANSI-C Programs’. In: *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by Kurt Jensen and Andreas Podelski. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 168–176. ISBN: 978-3-540-24730-2 (cit. on p. 105).
- De Nicola, Rocco and Frits Vaandrager (1990). ‘Action versus state based logics for transition systems’. In: *LITP Spring School on Theoretical Computer Science*. Springer, pp. 407–419 (cit. on p. 17).
- Dehbonei, Babak and Fernando Mejia (1970). ‘Formal development of software in railways safety critical systems’. In: *WIT Transactions on The Built Environment* 7 (cit. on p. 8).
- Demner-Fushman, Dina, Wendy W. Chapman and Clement J. McDonald (2009). ‘What can natural language processing do for clinical decision support?’ In: *Journal of Biomedical Informatics* 42.5. Biomedical Natural Language Processing, pp. 760–772. DOI: [10.1016/j.jbi.2009.08.007](https://doi.org/10.1016/j.jbi.2009.08.007) (cit. on p. 16).
- Demner-Fushman, Dina, Willie J. Rogers and Alan R. Aronson (2017). ‘MetaMap Lite: an evaluation of a new Java implementation of MetaMap’. In: *Journal of the American Medical Informatics Association* 24.4, pp. 841–844. DOI: [10.1093/jamia/ocw177](https://doi.org/10.1093/jamia/ocw177). URL: <https://doi.org/10.1093/jamia/ocw177> (cit. on pp. 18, 19).
- Devlin, Jacob, Ming-Wei Chang, Kenton Lee and Kristina Toutanova (2018). ‘BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding’. In: *CoRR* abs/1810.04805. arXiv: [1810.04805](https://arxiv.org/abs/1810.04805). URL: <http://arxiv.org/abs/1810.04805> (cit. on p. 59).
- Dhar, Payal (2020). ‘The carbon impact of artificial intelligence’. In: *Nat Mach Intell* 2, pp. 423–5. DOI: <https://doi.org/10.1038/s42256-020-0219-9> (cit. on p. 81).

- Diamantopoulos, Themistoklis, Michael Roth, Andreas Symeonidis and Ewan Klein (June 2017). ‘Software Requirements as an Application Domain for Natural Language Processing’. In: *Lang. Resour. Eval.* 51.2, pp. 495–524. ISSN: 1574-020X. URL: [10.1007/s10579-017-9381-z](https://doi.org/10.1007/s10579-017-9381-z) (cit. on pp. 12–15, 22, 41–43, 45, 46, 49).
- Dragoni, Mauro, Serena Villata, Williams Rizzi and Guido Governatori (2016). ‘Combining NLP approaches for rule extraction from legal documents’. In: *1st Workshop on Mining and Reasoning with Legal texts (MIREL 2016)* (cit. on p. 116).
- Dutertre, Bruno (2014). ‘Yices 2.2’. In: *Computer Aided Verification*. Ed. by Armin Biere and Roderick Bloem. Springer International Publishing, pp. 737–744. ISBN: 978-3-319-08867-9 (cit. on p. 104).
- Dzifcak, Juraj, Matthias Scheutz, Chitta Baral and Paul Schermerhorn (2009). ‘What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution’. In: *2009 IEEE International Conference on Robotics and Automation*. IEEE, pp. 4163–4168 (cit. on p. 116).
- Easterbrook, Steve, Robyn Lutz, Richard Covington, John Kelly, Yoko Ampo and David Hamilton (1998). ‘Experiences using lightweight formal methods for requirements modeling’. In: *IEEE Transactions on Software Engineering* 24.1, pp. 4–14 (cit. on p. 8).
- Elhadad, Noémie, Sameer Pradhan, Sharon Gorman, Suresh Manandhar, Wendy Chapman and Guergana Savova (2015). ‘SemEval-2015 task 14: Analysis of clinical text’. In: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 303–310 (cit. on p. 19).
- Enger, Martine, Erik Velldal and Lilja Øvrelid (Apr. 2017). ‘An open-source tool for negation detection: a maximum-margin approach’. In: *Proceedings of the Workshop Computational Semantics Beyond Events and Roles*. Valencia, Spain: Association for Computational Linguistics, pp. 64–69. DOI: [10.18653/v1/W17-1810](https://doi.org/10.18653/v1/W17-1810). URL: <https://www.aclweb.org/anthology/W17-1810> (cit. on p. 116).
- Erkinger, Christoph (2013). *Rotating workforce scheduling as satisfiability modulo theories* (cit. on p. 105).
- Ezen-Can, Aysu (2020). ‘A Comparison of LSTM and BERT for Small Corpus’. In: URL: <https://arxiv.org/abs/2009.05451> (cit. on p. 61).
- Fillmore, Charles J. (1968). ‘The Case for Case’. In: *Universals in Linguistic Theory*. Ed. by Emmon Bach and R. Harms. Holt, Rinehart, and Winston (cit. on pp. 11, 33).
- Friedman, Carol, Pauline Kra, Hong Yu, Michael Krauthammer and Andrey Rzhetsky (2001). ‘GENIES: a natural-language processing system for the extraction of molecular pathways from journal articles’. In: *ISMB (supplement of bioinformatics)*, pp. 74–82 (cit. on p. 19).

- Frisch, Alan M., Warwick Harvey, Chris Jefferson, Bernadette Martinez-Hernández and Ian Miguel (Sept. 2008). ‘Essence: A Constraint Language for Specifying Combinatorial Problems’. In: *Constraints* 13.3, pp. 268–306. ISSN: 1383-7133. DOI: [10.1007/s10601-008-9047-y](https://doi.org/10.1007/s10601-008-9047-y). URL: <https://doi.org/10.1007/s10601-008-9047-y> (cit. on p. 104).
- Frisch, Alan M., Christopher Jefferson, Bernadette Martinez Hernández and Ian Miguel (2005). ‘The rules of constraint modelling’. In: *International Joint Conferences on Artificial Intelligence*, pp. 109–116 (cit. on p. 104).
- Gaizauskas, Robert, George Demetriou, Peter J. Artymiuk and Peter Willett (2003). ‘Protein structures and information extraction from biological texts: the PASTA system’. In: *Bioinformatics* 19.1, pp. 135–143 (cit. on p. 19).
- Ghotbi, Seyed Hossein and Bernd Fischer (2012). ‘Fine-grained role-and attribute-based access control for web applications’. In: *International Conference on Software and Data Technologies*. Springer, pp. 171–187 (cit. on p. 105).
- Gildea, Daniel and Daniel Jurafsky (Sept. 2002). ‘Automatic Labeling of Semantic Roles’. In: *Computational Linguistics* 28.3, pp. 245–288. ISSN: 0891-2017. DOI: [10.1162/089120102760275983](https://doi.org/10.1162/089120102760275983). URL: <https://doi.org/10.1162/089120102760275983> (cit. on pp. 13, 46).
- Giordano, Laura, Paolo Terenziani, Alessio Bottrighi, Stefania Montani and Loredana Donzella (2006). ‘Model Checking for Clinical Guidelines: an Agent-based Approach’. In: *AMIA Annual Symposium*, pp. 289–93 (cit. on pp. 17, 37).
- Gleick, James (1996). *Little Bug, Big Bang*. URL: <https://web.archive.org/web/20200618225621/https://www.nytimes.com/1996/12/01/magazine/little-bug-big-bang.html> (cit. on p. 8).
- Godefroid, Patrice, Nils Klarlund and Koushik Sen (June 2005). ‘DART: Directed Automated Random Testing’. In: *SIGPLAN Not.* 40.6, pp. 213–223. ISSN: 0362-1340. DOI: [10.1145/1064978.1065036](https://doi.org/10.1145/1064978.1065036). URL: <https://doi.org/10.1145/1064978.1065036> (cit. on p. 105).
- Gorinski, Philip John, Honghan Wu, Claire Grover, Richard Tobin, Conn Talbot, Heather Whalley, Cathie Sudlow, William Whiteley and Beatrice Alex (2019). ‘Named Entity Recognition for Electronic Health Records: A Comparison of Rule-based and Machine Learning Approaches’. In: *CoRR* abs/1903.03985. URL: <http://arxiv.org/abs/1903.03985> (cit. on p. 19).
- Gorrell, Genevieve, Xingyi Song and Angus Roberts (2018). ‘Bio-YODIE: A Named Entity Linking System for Biomedical Text’. In: *CoRR* abs/1811.04860. URL: <http://arxiv.org/abs/1811.04860> (cit. on pp. 18, 19).
- Greiver, Michelle, Alys Havard, Juliana KF Bowles, Sumeet Kalia, Tao Chen, Babak Aliarzadeh, Rahim Moineddin, Julian Sherlock, William Hinton, Frank Sullivan, Braden O’Neill, Conrad Pow, Aashka Bhatt, Fahrurrozi Rahman, Bernardo Meza-Torres, Melisa Litchfield and Simon de Lusignan (2021). ‘Trends in diabetes medic-

- ation use in Australia, Canada, England, and Scotland: a repeated cross-sectional analysis in primary care’. In: *British Journal of General Practice* 71.704, e209–e218. ISSN: 0960-1643. DOI: [10.3399/bjgp20X714089](https://doi.org/10.3399/bjgp20X714089). eprint: <https://bjgp.org/content/71/704/e209.full.pdf>. URL: <https://bjgp.org/content/71/704/e209> (cit. on pp. 99, 101).
- Grivas, Andreas, Beatrice Alex, Claire Grover, Richard Tobin and William Whiteley (Nov. 2020). ‘Not a cute stroke: Analysis of Rule- and Neural Network-based Information Extraction Systems for Brain Radiology Reports’. In: *Proceedings of the 11th International Workshop on Health Text Mining and Information Analysis*. Online: Association for Computational Linguistics, pp. 24–37. DOI: [10.18653/v1/2020.louhi-1.4](https://doi.org/10.18653/v1/2020.louhi-1.4). URL: <https://aclanthology.org/2020.louhi-1.4> (cit. on pp. 21, 22, 51, 116).
- Hajic, Jan, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, M Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek et al. (2009). ‘The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages’. In: *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pp. 1–18 (cit. on p. 15).
- Hall, Anthony (1996). ‘Using formal methods to develop an ATC information system’. In: *IEEE Software* 13.2, pp. 66–76 (cit. on p. 8).
- Hao, Karen (2019). ‘Training a single AI model can emit as much carbon as five cars in their lifetimes’. In: *MIT Technology Review* (cit. on p. 81).
- Hartmanns, Arnd and Holger Hermanns (Nov. 2015). ‘In the Quantitative Automata Zoo’. In: *Sci. Comput. Program.* 112.P1, pp. 3–23. ISSN: 0167-6423. DOI: [10.1016/j.scico.2015.08.009](https://doi.org/10.1016/j.scico.2015.08.009). URL: <https://doi.org/10.1016/j.scico.2015.08.009> (cit. on p. 99).
- Heitmeyer, Constance, James Kirby, Bruce Labaw and Ramesh Bharadwaj (1998). ‘SCR: A toolset for specifying and analyzing software requirements’. In: *International Conference on Computer Aided Verification*. Springer, pp. 526–531 (cit. on p. 10).
- Hendrickx, Iris, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano and Stan Szpakowicz (July 2010). ‘SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations between Pairs of Nominals’. In: *Proceedings of the 5th International Workshop on Semantic Evaluation*. Uppsala, Sweden: Association for Computational Linguistics, pp. 33–38. URL: <https://www.aclweb.org/anthology/S10-1006> (cit. on p. 55).
- Henzinger, Thomas A, Xavier Nicollin, Joseph Sifakis and Sergio Yovine (1994). ‘Symbolic model checking for real-time systems’. In: *Information and computation* 111.2, pp. 193–244 (cit. on p. 97).

- Hochreiter, Sepp and Jürgen Schmidhuber (Dec. 1997). ‘Long Short-term Memory’. In: *Neural computation* 9, pp. 1735–1780. DOI: <https://doi.org/10.1162/neco.1997.9.8.1735> (cit. on p. 54).
- Holzmann, Gerard J. (2003). *The spin model checker: primer and reference manual*. Addison-Wesley (cit. on p. 17).
- Huth, Michael and Mark Ryan (2004). *Logic in Computer Science: Modelling and Reasoning about Systems*. 2nd ed. Cambridge University Press. DOI: [10.1017/CB09780511810275](https://doi.org/10.1017/CB09780511810275) (cit. on p. 87).
- Imler, Timothy D., Justin Morea, Charles J. Kahi, Jon Cardwell, Cynthia S. Johnson, Huiping Xu, Dennis J. Ahnen, Fadi Antaki, Christopher J. Ashley, György Baffy, Ilseung Cho, Jason A. Dornitz, Jason Hou, Mark A. Korsten, Anil Nagar, Kittichai Promrat, Douglas E. Robertson, Sameer D. Saini, Amandeep K. Shergill, Walter Smalley and Thomas F. Imperiale (2015). ‘Multi-Center Colonoscopy Quality Measurement Utilizing Natural Language Processing’. In: *The American Journal of Gastroenterology* 110, pp. 543–552. DOI: [10.1038/ajg.2015.51](https://doi.org/10.1038/ajg.2015.51) (cit. on p. 16).
- Institute of Medicine (1990). *Clinical Practice Guidelines: Directions for a New Program*. Ed. by Marilyn J. Field and Kathleen N. Lohr. Washington, DC: The National Academies Press. ISBN: 978-0-309-04346-5. DOI: [10.17226/1626](https://doi.org/10.17226/1626). URL: <https://www.nap.edu/catalog/1626/clinical-practice-guidelines-directions-for-a-new-program> (cit. on p. 1).
- Jacky, Jonathan (Feb. 1995). ‘Specifying a Safety-Critical Control System in Z’. In: *IEEE Transactions on Software Engineering* 21.2, pp. 99–106. ISSN: 0098-5589. DOI: [10.1109/32.345826](https://doi.org/10.1109/32.345826). URL: [10.1109/32.345826](https://doi.org/10.1109/32.345826) (cit. on p. 8).
- Johnson, Justin M and Taghi M Khoshgoftaar (2019). ‘Survey on deep learning with class imbalance’. In: *Journal of Big Data* 6.1, pp. 1–54 (cit. on p. 68).
- Jurafsky, Daniel and James H. Martin (2009). *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Upper Saddle River, N.J.: Pearson Prentice Hall. ISBN: 9780131873216 (cit. on p. 52).
- Kim, Gene, Patrick Debois, John Willis and Jez Humble (2016). *The DevOps Handbook: How to Create World-class Agility, Reliability, and Security in Technology Organizations*. ITpro collection. IT Revolution Press. ISBN: 9781942788003 (cit. on p. 7).
- Kogan, Yacov, Nigel Collier, Serguei V. S. Pakhomov and M. Krauthammer (2005). ‘Towards Semantic Role Labeling IE in the Medical Literature’. In: *AMIA Annual Symposium Proceedings*, p. 410 (cit. on p. 41).
- Kothari, Nupur, Ratul Mahajan, Todd Millstein, Ramesh Govindan and Madanlal Musuvathi (2011). ‘Finding protocol manipulation attacks’. In: *Proceedings of the ACM SIGCOMM 2011 Conference*, pp. 26–37 (cit. on p. 105).



- Kuhn, Tobias (2014). ‘A Survey and Classification of Controlled Natural Languages’. In: *Computational Linguistics* 40.1, pp. 121–170. URL: [10.1162/COLI\\_a\\_00168](https://doi.org/10.1162/COLI_a_00168) (cit. on p. 10).
- Kwiatkowski, Tom, Luke Zettlemoyer, Sharon Goldwater and Mark Steedman (2010). ‘Inducing probabilistic CCG grammars from logical form with higher-order unification’. In: *Proceedings of the 2010 conference on empirical methods in natural language processing*, pp. 1223–1233 (cit. on p. 116).
- Kwiatkowska, Marta, Gethin Norman and David Parker (2011). ‘PRISM 4.0: Verification of Probabilistic Real-Time Systems’. In: *Computer Aided Verification*. Ed. by Ganesh Gopalakrishnan and Shaz Qadeer. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 585–591. ISBN: 978-3-642-22110-1 (cit. on pp. 5, 90, 98).
- Lample, Guillaume, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami and Chris Dyer (June 2016). ‘Neural Architectures for Named Entity Recognition’. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, pp. 260–270. DOI: [10.18653/v1/N16-1030](https://doi.org/10.18653/v1/N16-1030). URL: <https://aclanthology.org/N16-1030> (cit. on p. 71).
- Lan, Zhenzhong, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma and Radu Soricut (2019). ‘ALBERT: A Lite BERT for Self-supervised Learning of Language Representations’. In: URL: <http://arxiv.org/abs/1909.11942> (cit. on p. 62).
- Lee, Jinhyuk, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So and Jaewoo Kang (2020). ‘BioBERT: a pre-trained biomedical language representation model for biomedical text mining’. In: *Bioinformatics* 36.4, pp. 1234–1240 (cit. on p. 63).
- Lindberg, Donald AB, Betsy L Humphreys and Alexa T McCray (1993). ‘The unified medical language system’. In: *Methods of information in medicine* 32.4, p. 281 (cit. on p. 18).
- Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer and Veselin Stoyanov (2019). ‘RoBERTa: A Robustly Optimized BERT Pretraining Approach’. In: URL: <http://arxiv.org/abs/1907.11692> (cit. on p. 63).
- Lukasik, Michal, Srinadh Bhojanapalli, Aditya Krishna Menon and Sanjiv Kumar (2020). ‘Does label smoothing mitigate label noise?’ In: *CoRR* abs/2003.02819. arXiv: [2003.02819](https://arxiv.org/abs/2003.02819). URL: <https://arxiv.org/abs/2003.02819> (cit. on p. 81).
- Makarek, Victor and Lior Rokach (2020). ‘Lessons Learned from Applying off-the-shelf BERT: There is no Silver Bullet’. In: URL: <https://arxiv.org/abs/2009.07238> (cit. on p. 61).

- Matuszek, Cynthia, Evan Herbst, Luke Zettlemoyer and Dieter Fox (2013). ‘Learning to parse natural language commands to a robot control system’. In: *Experimental robotics*. Springer, pp. 403–415 (cit. on p. 116).
- McCausland, Phil (2019). *Self-driving Uber car that hit and killed woman did not recognize that pedestrians jaywalk*. URL: <https://www.nbcnews.com/tech/tech-news/self-driving-uber-car-hit-killed-woman-did-not-recognize-n1079281> (cit. on p. 8).
- McHugh, Mary L. (Oct. 2012). ‘Interrater reliability: the kappa statistic’. In: *Biochemia Medica* 22.3, pp. 276–282. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3900052> (cit. on p. 66).
- McMillan, Kenneth L., Shaz Qadeer and James B. Saxe (2000). ‘Induction in Compositional Model Checking’. In: *Computer Aided Verification*. Ed. by E. Allen Emerson and Aravinda Prasad Sistla. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 312–327. ISBN: 978-3-540-45047-4 (cit. on p. 17).
- Medina-Moreira, José, Katty Lagos-Ortiz, Harry Luna-Aveiga, Óscar Apolinario-Arzuabe, María del Pilar Salas-Zárata and Rafael Valencia-García (2017). ‘Knowledge Acquisition Through Ontologies from Medical Natural Language Texts’. In: *Journal of Information Technology Research (JITR)* 10, pp. 56–69. DOI: [0.4018/JITR.2017100104](https://doi.org/10.4018/JITR.2017100104) (cit. on p. 41).
- Mehrabi, Saeed, Anand Krishnan, Sunghwan Sohn, Alexandra M Roch, Heidi Schmidt, Joe Kesterson, Chris Beesley, Paul Dexter, C Max Schmidt, Hongfang Liu et al. (2015). ‘DEEPEN: A negation detection system for clinical text incorporating dependency relation into NegEx’. In: *Journal of biomedical informatics* 54, pp. 213–219 (cit. on p. 116).
- Mikolov, Tomas, Kai Chen, Greg Corrado and Jeffrey Dean (2013). ‘Efficient Estimation of Word Representations in Vector Space’. In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: <http://arxiv.org/abs/1301.3781> (cit. on p. 48).
- Mikolov, Tomas, Edouard Grave, Piotr Bojanowski, Christian Puhersch and Armand Joulin (May 2018). ‘Advances in Pre-Training Distributed Word Representations’. In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA). URL: <https://www.aclweb.org/anthology/L18-1008> (cit. on p. 49).
- Miksch, Silvia, Yuval Shahar and Peter Johnson (1997). ‘Asbru: a task-specific, intention-based, and time-oriented language for representing skeletal plans’. In: *Proceedings of the 7th Workshop on Knowledge Engineering: Methods & Languages (KEML-97)*. Milton Keynes, UK, The Open University, Milton Keynes, UK, pp. 9–19 (cit. on pp. 17, 116).

- Moura, Leonardo de and Nikolaj Bjørner (2008). ‘Z3: An Efficient SMT Solver’. In: *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by C. R. Ramakrishnan and Jakob Rehof. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 337–340. ISBN: 978-3-540-78800-3 (cit. on pp. 5, 90, 104).
- Nethercote, Nicholas, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck and Guido Tack (Sept. 2007). ‘MiniZinc: Towards a Standard CP Modelling Language’. In: *Principles and Practice of Constraint Programming – CP 2007*. Berlin, Germany: Springer, pp. 529–543. ISBN: 978-3-540-74969-1. DOI: [10.1007/978-3-540-74970-7\\_38](https://doi.org/10.1007/978-3-540-74970-7_38) (cit. on p. 104).
- News, BBC (2012). *RBS computer problems kept man in prison*. URL: <https://web.archive.org/web/20200618231946/https://www.bbc.com/news/uk-18589280> (cit. on p. 8).
- Nguyen, Thien Huu and Ralph Grishman (June 2015). ‘Relation Extraction: Perspective from Convolutional Neural Networks’. In: *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. Denver, Colorado: Association for Computational Linguistics, pp. 39–48. DOI: [10.3115/v1/W15-1506](https://doi.org/10.3115/v1/W15-1506). URL: <https://www.aclweb.org/anthology/W15-1506> (cit. on p. 55).
- Nieuwenhuis, Robert and Albert Oliveras (2006). ‘On SAT modulo theories and optimization problems’. In: *International conference on theory and applications of satisfiability testing*. Springer, pp. 156–169 (cit. on p. 105).
- Norman, Gethin, David Parker and Jeremy Sproston (2013). ‘Model checking for probabilistic timed automata’. In: *Formal Methods in System Design* 43.2, pp. 164–190 (cit. on p. 93).
- Online, Today (2014). *Toyota recalls 466,000 vehicles globally for spare tire, braking issues*. URL: <https://www.todayonline.com/business/toyota-recalls-466000-vehicles-globally-spare-tire-braking-issues> (cit. on p. 8).
- Palmer, Martha, Daniel Gildea and Nianwen Xue (2010). ‘Semantic Role Labeling’. In: *Synthesis Lectures on Human Language Technologies* 3.1 (cit. on pp. 40, 41).
- Patel, Pinal, Disha Davey, Vishal Panchal and Parth Pathak (2018). ‘Annotation of a large clinical entity corpus’. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2033–2042 (cit. on p. 19).
- Peleska, Jan, Elena Vorobev, Florian Lapschies and Cornelia Zahlten (2011). ‘Automated model-based testing with RT-Tester’. In: *University of Bremen* (cit. on p. 10).
- Peng, Yifan, Xiaosong Wang, Le Lu, Mohammadhadi Bagheri, Ronald Summers and Zhiyong Lu (2018). ‘Negbio: a high-performance tool for negation and uncertainty detection in radiology reports’. In: *AMIA Summits on Translational Science Proceedings* 2018, p. 188 (cit. on p. 116).

- Pennington, Jeffrey, Richard Socher and Christopher D. Manning (2014). ‘GloVe: Global Vectors for Word Representation’. In: *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543. URL: <http://www.aclweb.org/anthology/D14-1162> (cit. on p. 49).
- Pérez, Beatriz and Ivan Porres (2010). ‘Authoring and verification of clinical guidelines: A model driven approach’. In: *Journal of Biomedical Informatics* 43.4, pp. 520–536. ISSN: 1532-0464. DOI: [10.1016/j.jbi.2010.02.009](https://doi.org/10.1016/j.jbi.2010.02.009) (cit. on pp. 17, 18, 37).
- Pnueli, Amir (1977). ‘The temporal logic of programs’. In: *18th Annual Symposium on Foundations of Computer Science*. IEEE, pp. 46–57 (cit. on p. 17).
- Pradhan, Sameer, Wendy Chapman, Suresh Man and Guergana Savova (2014). ‘Semeval-2014 task 7: Analysis of clinical text’. In: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)* (cit. on p. 19).
- Pressman, Roger S. and Bruce R. Maxim (2015). *Software engineering: a practitioner’s approach*. 8th ed. McGraw-Hill Education (cit. on p. 6).
- Rahman, Fahrurrozi and Juliana Bowles (2021a). ‘Semantic Annotations in Clinical Guidelines’. In: *From Data to Models and Back*. Ed. by Juliana Bowles, Giovanna Broccia and Mirco Nanni. Springer International Publishing, pp. 190–205. ISBN: 978-3-030-70650-0 (cit. on pp. 5, 39, 64).
- (2021b). ‘Semantic Labelling in Clinical Guidelines’. to appear in Healthcare Text Analytics Conference 2021 (cit. on pp. 5, 64, 66).
- Rahman, Fahrurrozi and Juliana Küster Filipe Bowles (2017). ‘Formal Verification of CNL Health Recommendations’. In: *Integrated Formal Methods*. Ed. by Nadia Polikarpova and Steve Schneider. Springer International Publishing, pp. 357–371. ISBN: 978-3-319-66845-1 (cit. on pp. 4, 5, 23, 84).
- Read, Jonathon, Erik Velldal, Marc Cavazza and Gersende Georg (May 2016). ‘A Corpus of Clinical Practice Guidelines Annotated with the Importance of Recommendations’. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*. Portorož, Slovenia: European Language Resources Association (ELRA), pp. 1724–1731. URL: <https://www.aclweb.org/anthology/L16-1272> (cit. on p. 19).
- Reed, Scott E., Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan and Andrew Rabinovich (2015). ‘Training Deep Neural Networks on Noisy Labels with Bootstrapping’. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: <http://arxiv.org/abs/1412.6596> (cit. on p. 81).
- Rosa, Jolyn (2018). *Ballistic missile warning sent in error by Hawaii authorities*. URL: <https://www.reuters.com/article/us-usa-missiles-falsealarm-idUSKBN1F20U1> (cit. on p. 8).

- Roscoe, A. W. (1997). *The Theory and Practice of Concurrency*. Prentice Hall PTR. ISBN: 0136744095 (cit. on p. 10).
- Rosenblatt, Frank (1957). *The Perceptron — a perceiving and recognizing automaton*. Tech. rep. 85-460-1. Cornell Aeronautical Laboratory (cit. on p. 51).
- Royce, Winston W. (Aug. 1970). ‘Managing the Development of Large Software Systems: Concepts and Techniques’. In: *Proceedings of IEEE WESCON*. Los Angeles, California, pp. 1–9 (cit. on p. 7).
- Savova, Guergana K., Steven Bethard, William F. Styler, James H. Martin, Martha Palmer, James J. Masanz and Wayne H. Ward (2009). ‘Towards Temporal Relation Discovery from the Clinical Narrative’. In: *AMIA Annual Symposium proceedings 2009*, pp. 568–72 (cit. on p. 18).
- Schwaber, Ken and Mike Beedle (2002). *Agile software development with Scrum*. Prentice-Hall (cit. on p. 7).
- Segura-Bedmar, Isabel, Paloma Martinez and Maria Herrero-Zazo (June 2013). ‘SemEval-2013 Task 9 : Extraction of Drug-Drug Interactions from Biomedical Texts (DDIExtraction 2013)’. In: *Second Joint Conference on Lexical and Computational Semantics (\*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Atlanta, Georgia, USA: Association for Computational Linguistics, pp. 341–350. URL: <https://www.aclweb.org/anthology/S13-2056> (cit. on p. 73).
- Segura-Bedmar, Isabel, Paloma Martinez and Maria Segura-Bedmar (2008). ‘Drug name recognition and classification in biomedical texts: a case study outlining approaches underpinning automated systems’. In: *Drug discovery today* 13.17-18, pp. 816–823 (cit. on p. 19).
- Settles, Burr (2004). ‘Biomedical named entity recognition using conditional random fields and rich feature sets’. In: *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications (NLPBA/BioNLP)*, pp. 107–110 (cit. on p. 19).
- Shah, Nigam H., Paea LePendou, Anna Bauer-Mehren, Yohannes T. Ghebremariam, Srinivasan V. Iyer, Jake Marcus, Kevin T. Nead, John P. Cooke and Nicholas J. Leeper (June 2015). ‘Proton Pump Inhibitor Usage and the Risk of Myocardial Infarction in the General Population’. In: *PLOS ONE* 10, pp. 1–16. DOI: [10.1371/journal.pone.0124653](https://doi.org/10.1371/journal.pone.0124653) (cit. on pp. 16, 19).
- Shahar, Yuval, Silvia Miksch and Peter Johnson (1998). ‘The Asgaard project: a task-specific framework for the application and critiquing of time-oriented clinical guidelines’. In: *Artificial Intelligence in Medicine* 14.1, pp. 29–51. ISSN: 0933-3657. DOI: [10.1016/S0933-3657\(98\)00015-3](https://doi.org/10.1016/S0933-3657(98)00015-3) (cit. on pp. 17, 37).
- Shaheen, Qurat-ul-ain, Alice Toniolo and Juliana K. F. Bowles (2020). ‘Dialogue Games for Explaining Medication Choices’. In: *Rules and Reasoning*. Ed. by Víctor Gutiérrez-Basulto, Tomáš Kliegr, Ahmet Soylu, Martin Giese and Dumitru Ro-

- man. Cham: Springer International Publishing, pp. 97–111. ISBN: 978-3-030-57977-7 (cit. on p. 116).
- (2021). ‘Argumentation-Based Explanations of Multimorbidity Treatment Plans’. In: *PRIMA 2020: Principles and Practice of Multi-Agent Systems*. Ed. by Takahiro Uchiya, Quan Bai and Iván Marsá Maestre. Cham: Springer International Publishing, pp. 394–402. ISBN: 978-3-030-69322-0 (cit. on p. 116).
- Sommerville, Ian (2016). *Software engineering*. 10th ed. Pearson Education. ISBN: 9781292096131 (cit. on p. 7).
- Son, Sooel, Seungwon Shin, Vinod Yegneswaran, Phillip Porras and Guofei Gu (2013). ‘Model checking invariant security properties in OpenFlow’. In: *2013 IEEE international conference on communications (ICC)*. IEEE, pp. 1974–1979 (cit. on p. 105).
- Taboada, M., M. Meizoso, D. Martínez, D. Riaño and A. Alonso (2013). ‘Combining open-source natural language processing tools to parse clinical practice guidelines’. In: *Expert Systems* 30.1, pp. 3–11. DOI: [10.1111/j.1468-0394.2010.00575.x](https://doi.org/10.1111/j.1468-0394.2010.00575.x). eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1468-0394.2010.00575.x>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1468-0394.2010.00575.x> (cit. on pp. 19, 20).
- Terenziani, Paolo, Gianpaolo Molino and Mauro Torchio (2001). ‘A modular approach for representing and executing clinical guidelines’. In: *Artificial Intelligence in Medicine* 23.3, pp. 249–276. ISSN: 0933-3657. DOI: [10.1016/S0933-3657\(01\)00087-2](https://doi.org/10.1016/S0933-3657(01)00087-2) (cit. on pp. 17, 37).
- Tjoa, Erico and Cuntai Guan (2019). ‘A Survey on Explainable Artificial Intelligence (XAI): Towards Medical XAI’. In: *CoRR* abs/1907.07374. URL: <http://arxiv.org/abs/1907.07374> (cit. on p. 116).
- Tjong Kim Sang, Erik F. and Fien De Meulder (2003). ‘Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition’. In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pp. 142–147. URL: <https://www.aclweb.org/anthology/W03-0419> (cit. on p. 72).
- ‘Chapter 1 - The challenge’ (2007). In: *Practical Model-Based Testing*. Ed. by Mark Utting and Bruno Legeard. San Francisco: Morgan Kaufmann, pp. 1–18. ISBN: 978-0-12-372501-1. DOI: <https://doi.org/10.1016/B978-012372501-1/50002-8>. URL: <https://www.sciencedirect.com/science/article/pii/B9780123725011500028> (cit. on p. 9).
- Valencia, Alfonso (2005). ‘Automatic annotation of protein function’. In: *Current opinion in structural biology* 15.3, pp. 267–274 (cit. on p. 19).
- Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser and Illia Polosukhin (2017). ‘Attention Is All You Need’.

- In: *CoRR* abs/1706.03762. arXiv: [1706.03762](https://arxiv.org/abs/1706.03762). URL: <http://arxiv.org/abs/1706.03762> (cit. on pp. 59, 60).
- Wang, Yefeng (2009). ‘Annotating and recognising named entities in clinical notes’. In: *Proceedings of the ACL-IJCNLP 2009 Student Research Workshop*, pp. 18–26 (cit. on p. 19).
- Wilk, Szymon, Martin Michalowski, Wojtek Michalowski, Daniela Rosu, Marc Carrier and Mounira Kezadri-Hamiaz (2017). ‘Comprehensive mitigation framework for concurrent application of multiple clinical practice guidelines’. In: *Journal of Biomedical Informatics* 66, pp. 52–71. ISSN: 1532-0464. DOI: [10.1016/j.jbi.2016.12.002](https://doi.org/10.1016/j.jbi.2016.12.002). URL: <https://www.sciencedirect.com/science/article/pii/S1532046416301770> (cit. on p. 21).
- Zeng, Daojian, Kang Liu, Siwei Lai, Guangyou Zhou and Jun Zhao (Aug. 2014). ‘Relation Classification via Convolutional Deep Neural Network’. In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, pp. 2335–2344. URL: <https://www.aclweb.org/anthology/C14-1220> (cit. on p. 56).
- Zettlemoyer, Luke and Michael Collins (Aug. 2009). ‘Learning Context-Dependent Mappings from Sentences to Logical Form’. In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore: Association for Computational Linguistics, pp. 976–984. URL: <https://www.aclweb.org/anthology/P09-1110> (cit. on p. 116).
- Zhang, Yaoyun, Buzhou Tang, Min Jiang, Jingqi Wang, Yonghui Wu and Hua Xu (2014). ‘Domain Adaptation for Semantic Role Labeling of Clinical Text’. In: *Journal of the American Medical Informatics Association : JAMIA* 22, pp. 967–79 (cit. on p. 41).
- Zhang, Yuhao, Yuhui Zhang, Peng Qi, Christopher D. Manning and Curtis P. Langlotz (2020). *Biomedical and Clinical English Model Packages in the Stanza Python NLP Library*. arXiv: [2007.14640](https://arxiv.org/abs/2007.14640) [cs.CL] (cit. on p. 52).

# Index of Terms

- ABNF Parser Generator (APG), 32  
ACT, 12, 34, 35, 37  
Action Computational Tree Logic (ACTL), 17  
AGT, 12, 34, 35, 37  
Air Traffic Control (ATC), 8  
Albumin Creatinine Ratio (ACR), 106, 108–110, 114  
Annals of the Rheumatic Disease (ARD), 64, 65  
Another Tool for Language Recognition (ANTLR), 32
- Bidirectional Encoder Representations from Transformers (BERT), 58–63, 67, 68, 71, 72, 74–80  
Boolean Satisfiability Problem (SAT), 18
- CAC, 12, 35, 37, 92  
CACT, 34, 35, 37  
Central Processing Unit (CPU), 76  
CFV, 12  
Chronic Kidney Disease (CKD), 106–110  
Clinical Decision Support (CDS), 16  
CMD, 12, 35, 37  
Colorectal Cancer (CRC), 16  
Communicating Sequential Processes (CSP), 10  
Computation Tree Logic (CTL), 83, 84, 88, 89, 97  
Construction of Useful Parsers (CUP), 32  
Context-Free Grammar (CFG), 10, 23, 24  
Controlled Natural Language (CNL), 4, 10, 12, 23, 27, 30, 32, 34, 37–40, 51, 112, 113, 115
- Convolutional Neural Network (CNN), 56  
CPT, 12, 35, 37, 91  
CTV, 12, 35, 37, 91, 92
- Data-Flow Reactive Systems (DFRS), 10, 12
- Electronic Health Records (EHR), 16  
Extensible Markup Language (XML), 17, 91, 95, 101
- First-Order Logic (FOL), 83, 85, 89  
Frame Element (FE), 33
- Global Vector (GloVe), 49, 50, 54, 56, 59, 70, 71, 76, 77  
Glycated Haemoglobin (HbA1c), 29–32, 36, 37, 90–93, 95, 100, 101  
Graphics Processing Unit (GPU), 76, 77, 80
- Intensive Care Unit (ICU), 19  
Inter-Annotator Agreement (IAA), 66, 67  
Intermediate Model Representation (IMR), 10  
Intravenous Catheters (IRC), 17, 18
- Lexical Unit (LU), 33  
Linear Temporal Logic (LTL), 17, 83, 84, 86–89  
Long Short-Term Memory (LSTM), 54, 58, 59, 61, 70, 113
- Machine Learning (ML), 13, 39  
Model-Based Testing (MBT), 9



Named-Entity Recognition (NER), 5, 19, 22, 51–54, 58, 59, 62, 69–73, 75, 77, 80, 106, 113, 116  
 National Institute for Health and Care Excellence (NICE), 1, 64, 65  
 Natural Language Processing (NLP), 3, 5, 6, 9, 10, 15, 16, 44, 52, 63  
 Natural Language Requirements to Test Cases (NAT2TEST), 9, 10, 12  
 Noun Phrase (NP), 40  
 Part-Of-Speech (POS), 10, 15, 27, 44, 50, 59, 70, 71  
 PAT, 12, 34, 35, 37, 91, 92  
 Process Meta Language (PROMELA), 17  
 Propositional Logic (PL), 83  
 Recurrent Neural Network (RNN), 54, 59, 61  
 Satisfiability Modulo Theory (SMT), 18, 104–106, 109, 114, 115  
 Scaffolding Scalable Software Services (S-CASE), 13–15  
 Scottish Intercollegiate Guidelines Network (SIGN), 1, 64, 65  
 Semantic Role Labelling (SRL), 5, 13, 40–42, 46, 50, 51, 67–70, 112, 113  
 Shared Annotated Resources (ShARe), 19  
 Software Cost Reduction (SCR), 10  
 TOV, 12, 34, 35, 37, 91  
 Transition-based Medical Recommendations for detecting Interactions (TMR4I), 20  
 Type 2 Diabetes (T2D), 1, 23, 25, 26, 28, 32, 35–37, 91–96, 99  
 Unified Medical Language System (UMLS), 18  
 Unified Modeling Language (UML), 7, 37



# Penn and Brown Corpora Tagset

## A.1 Penn Corpus Tagset

Table A.1: Penn Corpus Tagset.

Tag	Description
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun
PRP\$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to

Continued on next page

**Table A.1 – continued from previous page**

<b>Tag</b>	<b>Description</b>
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VCN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WP	Wh-pronoun
WP\$	Possessive wh-pronoun
WRB	Wh-adverb

## A.2 Brown Corpus Tagset

Table A.2: Brown Corpus Tagset.

<b>Tag</b>	<b>Description</b>
.	sentence closer
(	left parenthesis
)	right parenthesis
*	not, n't
–	dash
,	comma
:	colon
ABL	pre-qualifier
ABN	pre-quantifier
ABX	pre-quantifier
AP	post-determiner
AT	article
BE	be
BED	were
BEDZ	was
BEG	being
BEM	am
BEN	been
BER	are, art
BEZ	is
CC	coordinating conjunction
CD	cardinal numeral
CS	subordinating conjunction
DO	do
DOD	did
Continued on next page	

**Table A.2 – continued from previous page**

<b>Tag</b>	<b>Description</b>
DOZ	does
DT	singular determiner
DTI	singular or plural determiner/quantifier
DTS	plural determiner
DTX	determiner/double conjunction
EX	existential there
FW	foreign word (hyphenated before regular tag)
HL	word occurring in headline (hyphenated after regular tag)
HV	have
HVD	had (past tense)
HVG	having
HVN	had (past participle)
HVZ	has
IN	preposition
JJ	adjective
JJR	comparative adjective
JJS	semantically superlative adjective
JJT	morphologically superlative adjective
MD	modal auxiliary
NC	cited word (hyphenated after regular tag)
NN	singular or mass noun
NN\$	possessive singular noun
NNS	plural noun
NNS\$	possessive plural noun
NP	proper noun or part of name phrase
NP\$	possessive proper noun
NPS	plural proper noun
NPS\$	possessive plural proper noun
NR	adverbial noun
NRS	plural adverbial noun
OD	ordinal numeral
PN	nominal pronoun
PN\$	possessive nominal pronoun
PP\$	possessive personal pronoun
PP\$\$	second (nominal) possessive pronoun
PPL	singular reflexive/intensive personal pronoun
PPLS	plural reflexive/intensive personal pronoun
PPO	objective personal pronoun
PPS	3rd. singular nominative pronoun
PPSS	other nominative personal pronoun
QL	qualifier
QLP	post-qualifier
RB	adverb
RBR	comparative adverb

Continued on next page

**Table A.2 – continued from previous page**

<b>Tag</b>	<b>Description</b>
RBT	superlative adverb
RN	nominal adverb
RP	adverb/particle
TL	word occurring in title (hyphenated after regular tag)
TO	infinitive marker to
UH	interjection, exclamation
VB	verb, base form
VBD	verb, past tense
VBG	verb, present participle/gerund
VBN	verb, past participle
VBZ	verb, 3rd. singular present
WDT	wh- determiner
WP\$	possessive wh- pronoun
WPO	objective wh- pronoun
WPS	nominative wh- pronoun
WQL	wh- qualifier
WRB	wh- adverb

# B

## Annotation Guidelines

### B.1 Entity Annotation Guide

- ↻ If an entity spans multiple words, the first word would be marked with B-tag while the rest of the constituents are marked with I-tag. If an entity only spans a single word, it should be marked with B-tag.
- ↻ If a word does not belong to any entity, it should be marked with O.
- ↻ A noun or a noun phrase taking an action should be marked as an actor (acr). Note that we do not include the determiner as a part of the entity, i.e., *a*, *the*, *some*, etc.

For people who cannot tolerate aminosalicylates, consider a time-limited course of a topical or an oral corticosteroid.

In this example, the word "person" is the one tolerating, hence it is marked as B-acr.

- ↻ A noun or a noun phrase receiving an action from a verb should be marked as an actor (acr).

If the person is aged under 80 years with stage 1 hypertension and has target organ damage, offer antihypertensive drug treatment.

In this example, the word "person" is receiving the offer, hence it is marked as B-acr.

- ↻ An active verb (not stative predicate) should be marked as an action (acn).

If the person is aged under 80 years with stage 1 hypertension and has target organ damage, offer antihypertensive drug treatment.

In this example, the verb "has" and "offer" are marked as B-acn.

- ↻ In an action with auxiliary, such as "has established cardiovascular disease", we only mark "established" as B-acn.

- ✎ If there is a verb following another one, such as "consider tapering bDMARDs", we only mark "tapering" as B-acn.
- ✎ A noun or a noun phrase denoting a thing than an actor is having or an action is giving (not receiving) will be marked as an object (obj). The object can have preposition "of" as its constituent. This can include numbers as well.

If the person is aged under 80 years with stage 1 hypertension and has target organ damage, offer antihypertensive drug treatment.

In this example, "stage 1 hypertension" is marked as B-obj, I-obj, and I-obj respectively. The same with "target organ damage" that is marked as B-obj, I-obj and I-obj and "drug treatment" that is marked as B-obj and I-obj.

- ✎ A word or a phrase that serves as an adjective or an adverb modifying an actor, an object or an action, should be marked as modifier (mod).

If the person is aged under 80 years with stage 1 hypertension and has target organ damage, offer antihypertensive drug treatment.

In this example, "antihypertensive" is marked as B-mod.

## B.2 Relation Annotation Guide

- ✎ There are eight relationships: actor-of, acts-on, choice, joint, modifier, other, owner, and receiver.
- ✎ acts-on, modifier, owner and receiver could have two different instances based on the direction of the relationship (further discussed below).
- ✎ Given two entities, e1 and e2, if e1 is the entity who does an action e2, the relationship is marked as actor-of(e1,e2).

If the <e1> person </e1> is aged under 80 years with stage 1 hypertension and <e2> has </e2> target organ damage, offer antihypertensive drug treatment.

In this example, the relationship for "person" and "has" is actor-of(e1,e2).

- ✎ Between to entities e1 and e2, if one entity is an action and another entity is affected by the action, the relationship should be marked as acts-on.

If the person is aged under 80 years with stage 1 hypertension and has target organ damage, <e1> offer </e1> antihypertensive drug <e2> treatment </e2>.

In this example, as the action "offer" comes before the object "treatment", their relationship would be acts-on(e1,e2).

<e1> Treatment </e1> should be <e2> aimed </e2> at reaching the target of remission.

In this example, the action "aimed" comes after the object "Treatment", hence their relationship is acts-on(e2,e1).

- ⚡ If the two entities are joined together logically or explicitly by the word "and", their relationship would be joint, as shown in the example below between "bevacizumab" and "paclitaxel".

Women with platinum resistant relapsed ovarian cancer should be offered <e1> bevacizumab </e1> in combination with <e2> paclitaxel </e2>.

- ⚡ If the two entities are in an alternative relationship, they should be marked as choice, as shown between "AS" and "PsA" in the example below.

CVD risk estimation for patients with <e1> AS </e1> or <e2> PsA </e2> should be performed according to national guidelines.

- ⚡ If an adjective entity modifying another entity, their relationship should be marked as modifier.

The exercise tolerance test should not be used routinely as a <e1> first </e1> line diagnostic <e2> tool </e2>.

In the previous example, the word "first" is modifying "tool", hence their relationship is modifier(e1,e2).

<e1> Request </e1> the oestrogen receptor, and progesterone receptor <e2> simultaneously </e2>.

In the previous example, the modifier "simultaneously" comes after the entity "Request", hence their relationship is modifier(e2,e1).

- ⚡ If one of the entity is owning the other, their relationship would be owner.

<e1> Patients </e1> with known <e2> CAD </e2> should be given treatment.

Here, the relationship between "Patients" and "CAD" is owner(e1,e2).

Request tests to find <e1> CAD </e1> in <e2> patients </e2> with age under 80 years.

In the previous example, because "CAD" comes before "patients", their relationship would be owner(e2,e1).

- ⚡ If an entity is receiving an action from another entity, their relationship should be marked as receiver.

<e1> Patients </e1> with type 2 diabetes should be <e2> given </e2> metformin.

In this example, the "patients" comes before the action "given", hence their relationship is receiving(e1,e2).

<e1> Maintain </e1> a blood glucose concentration between 4 and 11 mmol/litre in <e2> people </e2> with acute stroke.

In this example, the "people" is receiving the action "Maintain", hence it is marked as receiving(e2,e1).



✎ Only put `other` as the relationship between two entities if it cannot be deduced. The example below illustrates this situation.

Do not offer `<e1> azathioprine </e1>`, mercaptopurine or methotrexate as `<e2> monotherapy </e2>` to induce remission.