A Case for Pre-trained Language Models in Systems Engineering

by

Shao Cong Lim

B.Sc. Chemistry, University of Manchester (2016)
M.Sc. Business Analytics and Operations Research, University of Manchester (2017)

Submitted to the System Design and Management Program
in partial fulfillment of the requirements for the degree of

Masters of Science in Engineering and Management
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2022

© 2022 Massachusetts Institute of Technology. All rights reserved.

Author …….
System Design and Management Program
August 5, 2022

Certified by …….
Dr. Eric S. Rebentisch
Research Associate, Sociotechnical Systems Research Center
Thesis Supervisor

Accepted by …….
Joan S. Rubin
Executive Director, System Design and Management Program

[Page intentionally left blank]

A Case for Pre-trained Language Models in Systems Engineering

by

Shao Cong Lim

Submitted to the System Design and Management Program
on August 5, 2022 in partial fulfillment of the
requirements for the degree of
Masters of Science in Engineering and Management

Abstract

Modern engineered systems are immensely complex. Extensive sets of natural language requirements guide the development of such systems. As such, tools to assist system engineers in managing and extracting information from these requirements must also scale to match the complexity of these systems. However, the systems engineering community has lagged in adopting advanced natural language processing techniques. Pre-trained language models, such as BERT, represent state-of-the-art in the field. This thesis seeks to understand if these pre-trained language models can achieve higher model performance at a lower computational and manpower cost than earlier techniques. The results show that adapting these language models through task-adaptive pretraining leads to consistent improvements in model performance and greater model robustness. These results indicate the potential of applying such language models in the systems engineering domain. However, much work remains to improve model performance and expand possible applications.

Thesis Supervisor: Eric S. Rebentisch
Title: Research Associate

Acknowledgements

I am grateful to Dr Eric Rebentisch for the opportunity to scope this thesis freely. I am also thankful for the insightful conversations with other researchers working on the related project. Those conversations have shaped my appreciation of this topic's usefulness and importance.

I am grateful to the Singapore Armed Forces for the opportunity to undergo the System Design and Management Program at MIT.

Lastly, I owe a debt of gratitude to my wife for her sacrifices to accompany me throughout my year in Cambridge.

Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
BERT	Bidirectional Encoder Representations from Transformers
BPE	Byte Pair Encoding
BOW	Bag-of-words
CR	Concept Recognition
CRF	Conditional Random Field
DAPT	Domain-Adaptive Pretraining
DE	Digital Engineering
ESA	European Space Agency
FR	Functional Requirement
LM	Language Models
LSTM	Long Short-Term Memory
ML	Machine Learning
MBSE	Model-based Systems Engineering
MLM	Masked Language Modelling
MRD	Machine Readable Dictionaries
NASA	National Aeronautics and Space Administration
NFR	Non-Functional Requirement
NLP	Natural Language Processing
NER	Named Entity Recognition
OOV	Out-of-vocabulary
POS	Parts-of-speech
RE	Requirements Engineering

PPL	Perplexity
RegEx	Regular Expression
RNN	Recurrent Neural Network
S&R	Search and Retrieval
SE	Systems Engineering
TAPT	Task-Adaptive Pretraining
ULMFit	Universal Language Model Fine-tuning

Chapter 1

1 Introduction

1.1 Background and Motivations

User needs for modern engineered products are increasingly sophisticated. The development of these complex products is in turn guided by large sets of system requirements and specifications. The accurate and comprehensive decoding of information from requirements ensures that the eventual product meets the specified requirements. However, while the sets of requirements have grown, the technologies and techniques used to decode information from them have remain largely stagnant. The use of engineers to perform this dull[1] and tedious task is not scalable and represents a poor use of their expertise. This thesis investigates the use of state-of-the-art natural language processing (NLP) techniques to bring forth a high-performing and scalable approach towards managing requirements. This allows expensive engineering resource to be better invested into other high value-adding functions.

In engineering, system requirements are conventionally specified and communicated using natural language. Natural language is governed by linguistic rules – rules which are culturally evolved and learned, rather than specified a priori (Spike, 2018). This means that natural language can be deeply subjective. For example, the ability to understand sarcasm is shaped by the receiver's social and interpersonal factors (Kreuz & Caucci, 2007). In contrast, machine (programming) languages have a pre-defined grammar and structure, allowing syntactic rules to be stated comprehensively. This means that a specific piece of code communicates the same functional meaning with all its users. This subjective nature of natural language creates challenges in decoding information from requirements.

[1] This is an intended reference to the 3D (Dull, Dirty and Dangerous) characteristics of tasks that are well suited for automation.

The field of NLP has long sought to advance the technological goal[2] of processing natural language (Levesque, 2014). At the same time, engineers have sought out technologies to extract information encoded within requirements. The alignment of goals between the domains has led to the diffusion of NLP technologies into systems engineering (SE) over time – one reference from each of the three preceding decades illustrates this (Arellano et al., 2015; Kof, 2005; Ryan, 1993).

The adoption of NLP technologies within system engineering is therefore influenced by both technological maturity and engineering needs. The recent survey of NLP applications in requirements engineering indicates that most of the work involves syntactic analysis such as detection of requirement statements from longer documents, categorical classification of requirements, and assessment of requirement quality (Ferrari et al., 2021). This is expected as the formulation of such syntactic problems is well understood, with developed technologies and tools to enable its application. In contrast, semantic applications, while present, are less prevalent. The reasons for this observation are less understood. The same survey also indicates that active academic research has not been widely applied within the industry. These findings shape the intended scope of this thesis. Therefore, this investigates the utility of various state-of-the-art language models that have been widely adopted in other industries within systems engineering and requirements engineering.

1.2 Research Questions

This thesis seeks to address the following research questions:

- **RQ1**: *Which off-the-shelf pre-trained language models are most suitable for application within the systems engineering domain?*
- **RQ2:** *To what extent does task-adaptive pretraining of language models improve classification performance within the systems engineering domain?*

[2] Technological goals are distinct from scientific goals. The former describes a goal of developing solutions that are sufficient for a defined problem, while scientific goals describe a purist pursuit of understanding what really is. In the context of NLP, scientific goals seek to describe how language is understood by humans, while technological goals seek to develop solutions to perform specific language related tasks.

1.3 Thesis Structure

This thesis contains six sections – this section and five others. Sections 2 and 3 present a literature review of SE and NLP, with emphasis on the state of practice and the envisioned future state. Section 4 described empirical work performed to evaluate three language models (LM) and their adapted analogs to three requirements engineering tasks. Finally, Section 5 discusses the implications of the results and proposes future work.

Chapter 2

2 Evolution of System Engineering

This chapter is organized into four main sections: Section 2.1 provides a brief overview of Systems Engineering, Section 2.2 describes the movement from a traditional document-based systems engineering to model-based systems engineering, Section 2.3 summarizes the future of systems engineering as articulated by INCOSE, and Section 2.4 describes how the recent developments in Systems Engineering paves the way for greater adoption of artificial intelligence and natural language processing tools within the field.

2.1 Systems Engineering

Systems Engineering is an "interdisciplinary, iterative and sociotechnical" approach to maximize a system's value to stakeholders throughout its lifecycle by managing "complexity and change" and "reducing risk associated with new systems or modifications to complex systems" (INCOSE, 2015). SE involves the performance of 14 technical processes (Table 2-1) throughout its lifecycle to achieve the stated outcomes. Holistically, these technical processes enable the system engineer to bridge across engineering disciplines to develop a mutually agreeable set of system requirements and system solutions that fulfill prioritized stakeholder needs (INCOSE, 2015).

Table 2-1: 14 System Engineering Technical Processes (INCOSE, 2015).

Technical Process	**Purpose**[3]	**Example of Process Artifacts**[4]
Business or mission analysis process	"To define the business or mission problem or opportunity, characterize the solution space, and determine potential solution class(es) that could address a problem or take advantage of an opportunity."	• Problem or opportunity statement • Major stakeholder identification • Business requirements

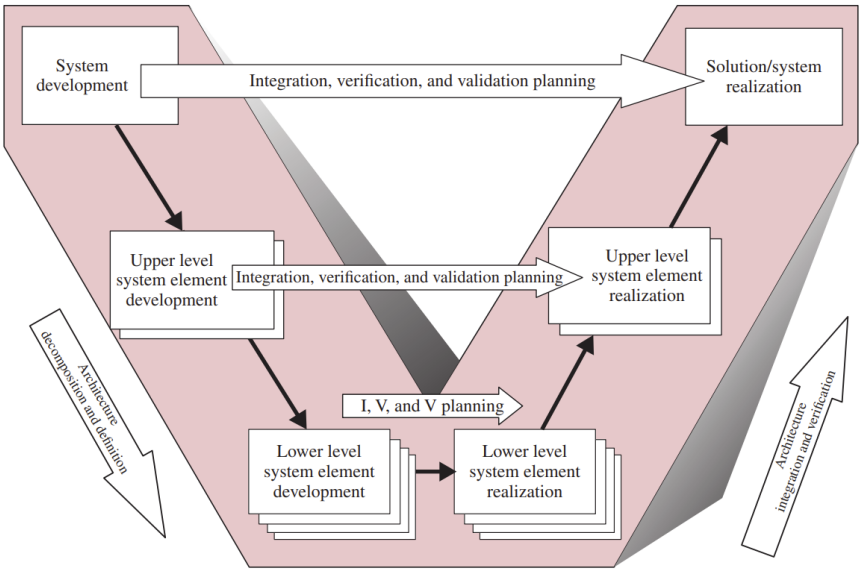
[3] Cited verbatim from INCOSE Systems Engineering Handbook (INCOSE, 2015).
[4] Cited verbatim from INCOSE Systems Engineering Handbook (INCOSE, 2015).

Technical Process	Purpose[3]	Example of Process Artifacts[4]
Stakeholder needs and requirements definition process	"To define the stakeholder requirements for a system that can provide the capabilities needed by users and other stakeholders in a defined environment."	• Stakeholder needs and requirements definition strategy • Stakeholder requirements
System requirements definition process	"To transform the stakeholder, user-oriented view of desired capabilities into a technical view of a solution that meets the operational needs of the user."	• System function definition • System requirements • System functional interface identification
Architecture definition process	"To generate system architecture alternatives, to select one or more alternative(s) that frame stakeholder concerns and meet system requirements, and to express this in a set of consistent views."	• System architecture description • Preliminary interface definition
Design definition process	"To provide sufficient detailed data and information about the system and its elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture."	• System design description • System element descriptions
System analysis process	"To provide a rigorous basis of data and information for technical understanding to aid decision-making across the life cycle."	• System analysis strategy • System analysis record
Implementation process	"To realize a specified system element."	• Implementation strategy • Implementation constraints
Integration process	"To synthesize a set of system elements into a realized system (product or service) that satisfies system requirements, architecture, and design."	• Integration enabling system requirements • Integration procedure
Verification process	"To provide objective evidence that a system or system element fulfils its specified requirements and characteristics."	• Verification strategy • Verification procedure • Verification record
Transition process	"To establish a capability for a system to provide services specified by stakeholder requirements in the operational environment."	• Transition strategy • Transition procedure
Validation process	"To provide objective evidence that the system, when in use, fulfills its business or mission objectives and stake-holder requirements, achieving its intended use in its intended operational environment."	• Validation strategy • Validation procedure • Validated requirements • Validation record

Technical Process	Purpose[3]	Example of Process Artifacts[4]
Operation process	"To use the system to deliver its services."	• Operation strategy • Operation constraints
Maintenance process	"To sustain the capability of the system to provide a service."	• Maintenance strategy • Maintenance procedure
Disposal process	"To end the existence of a system element or system for a specified intended use, appropriately handle replaced or retired elements, and to properly attend to identified critical disposal needs."	• Disposal strategy • Disposal procedure

The order in which these technical processes are performed depends on the way SE practices are organized. This is well illustrated by the SE "Vee" model (Figure 2-1). In this model, system maturity increases from left to right. Moving down the left arm of the "Vee", the system definition processes (such as architecture definition and design definition) are performed in phases, first at the system level and subsequently at the sub-system level. Moving up the right arm, integration, verification, and validation processes are then performed at each level of abstraction, starting at the sub-system level before completion at the system level.

Figure 2-1: Systems Engineering "Vee" Model (INCOSE, 2015).



2.2 Movement towards Model-based Systems Engineering

2.2.1 Traditional Document-based Systems Engineering

The SE approach is iterative in nature and involves multiple stakeholders. This approach requires knowledge, decisions, and work products to be captured in various artifacts that serve as a single source of truth for all stakeholders. Traditionally, these artifacts exist as documents leading to a document-based systems engineering (DBSE) approach. The reliance on documents can also be attributed to legal acceptance standards (Logan et al., 2012) or ease of understanding by non-expert stakeholders.

However, the document-centric nature of DBSE does not imply an absence of models. Instead, much of the engineering work continues to revolve around models. This reliance on both documents and models demands the consistent translation of knowledge between the two to maintain an authoritative source of truth. There are several inherent weaknesses with this approach. First, the disparate use of models means that the underlying assumptions and semantics within those models are often inconsistent (Madni & Sievers, 2018). The resulting documents that are generated from these models inherit the same inconsistencies. Second, each of the 14 SE processes produces many artifacts (Table 2-1) that can continually evolve over time. Therefore, documents and models "have their own lifecycles and tend to drift apart over time" (Norheim et al., 2022). These weaknesses dilute the value proposition of the SE approach.

2.2.2 Model-based Systems Engineering

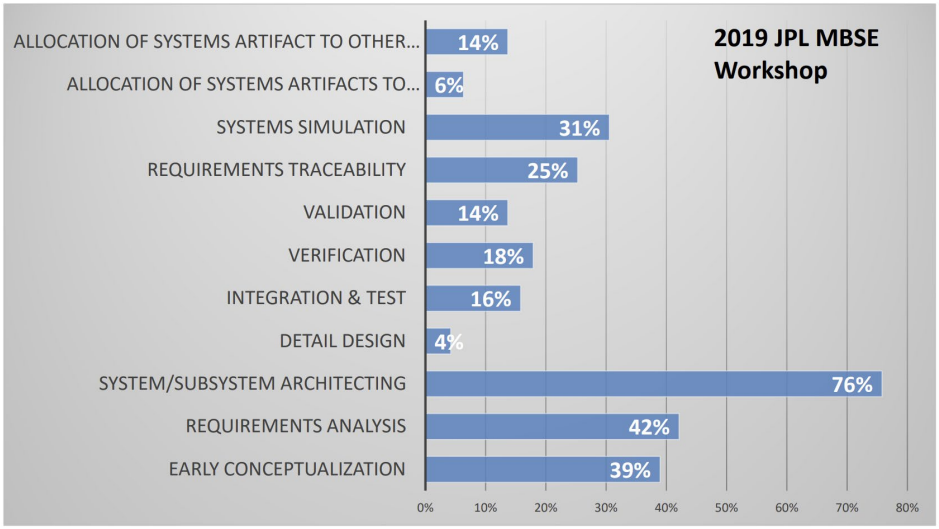
This introduction of model-based systems engineering (MBSE) is intended to tackle the above weaknesses. MBSE is formally defined as "the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases." (INCOSE, 2007) MBSE involves the use of a set of connected models which collectively represent the "primary artifact of the SE process" (INCOSE, 2015). Therefore, MBSE poses the system models as the authoritative source of truth, displacing documents from this role. It is posited that MBSE will foster and enhance communication effectiveness between stakeholders by creating a shared context and lexicon through the models (Madni & Sievers, 2018).

2.2.3 State of Practice of Model-based Systems Engineering

A series of surveys (Cameron & Adsit, 2020; Cloutier, 2015, 2019; Huldt & Stenius, 2019; Kim et al., 2019; Madni & Sievers, 2018) indicate that MBSE adoption within industry is ongoing. However, the primary MBSE adopters continue to come from industries that have traditionally practiced SE; such industries include defense and aerospace (Cloutier, 2019).

Amongst adopters, MBSE is primarily practiced at the front-end in the areas of architecture modelling and requirements management, with lesser adoption on rear-end processes such as trade studies and verification planning (Cloutier, 2019). This can be attributed to several sociotechnical reasons. First, the MBSE tools required for front-end processes such as requirements management are domain-agnostic (Cloutier, 2019) and present a larger target market. This in turn drives development and tool maturity. In comparison, rear-end processes require more specialized tools which may not be available. Second, there is a common perception amongst practitioners that MBSE holds the most promise in systems conceptualization and architecting, and requirements management (Figure 2-2). Taken together, these factors create a positive reinforcing loop that facilitates adoption of MBSE at the front-end of SE.

Figure 2-2: SE practitioner responses to "Where do we believe MBSE holds the most promise?" (Cloutier, 2019).



2.3 Future State of Systems Engineering

The International Council on Systems Engineering (INCOSE) release vision papers periodically to "inspire and guide the strategic direction of systems engineering for the global systems community" (INCOSE, 2021). This series of vision papers outline several key ideas such as the overarching SE imperatives (Figure 2-3), key global trends that impact the practice of SE, and an evolving outlook of the SE practice in the future. While the SE imperatives are grouped into five main categories, they share two underlying themes: to enable the widespread adoption of SE at scale across domains and to improve the efficiency of SE processes by increasing knowledge reuse and integration with other digital tools.

Figure 2-3: INCOSE Systems Engineering Imperatives (INCOSE, 2021).

In terms of global trends, the recent *Systems Engineering Vision 2035* paper highlighted the disruptive and transformative potential of the ongoing digital transformation (INCOSE,

2021). First, the digital transformation will bring about a movement to construct "robust digital representations of enterprise information, and semantically integrating information" throughout the organization (INCOSE, 2021). The SE community must adapt to the larger shift – the DBSE approach may become obsolete as it becomes incompatible with the way organizations of the future work. As such, tools to support MBSE must be developed to support this shift. Second, the proliferation of artificial intelligence (AI) applications will also accelerate, driven by the increase in computational power, data, and software (INCOSE, 2021). The infusion of AI into systems engineering tools and processes is an eventuality.

2.4 Concrete Steps for Artificial Intelligence Applications in Systems Engineering

This thesis aims to make a tangible step in applying state-of-the-art AI technology within SE in support of the SE imperatives. Specifically, this thesis will investigate the application of NLP technologies in requirements management. The application of NLP within requirements management is not novel. However, by the end of Chapter 3, it should become apparent that the gap between state-of-the-art AI/NLP and those in use within requirements management today is substantial. The bridging of this gap can facilitate the infusion of AI into SE. Considering that most of MBSE applications today reside in the front-end processes (such as requirements management) focusing on AI applications for such processes has the potential to deliver the most significant impact.

Chapter 3

3 Natural Language Processing

NLP sits at the intersection of linguistics and computer science, with technologies developed to decode information within natural language artifacts. NLP research is rich and diverse whose scope cannot be reproduced accurately or in whole. In the subsequent sections, emphasis is placed on discussing specific concepts that scaffold subsequent discussions on adoption strategies in Chapter 4.

This chapter is organized into four main sections: Section 3.1 provides an overview of NLP by presenting relevant linguistics concepts (such as syntax and semantics) and providing examples of NLP problem formulations. Section 3.2 illustrates traditional symbolic NLP approaches that have been adopted within the SE domain. Section 3.3 illustrates machine learning approaches that have become the cornerstone of modern-day NLP. Lastly, Section 3.4 provides a discussion of how NLP techniques can be applied in other specialized domains such as biomedicine and legal.

3.1 What is Natural Language Processing?

3.1.1 Natural Language Concepts from Linguistics

The field of linguistics comprises of many subfields that analyze natural language from different aspects and at different levels of abstraction. The two that are of relevance to this thesis are syntax and semantics. Syntax refers to "the study of formation and internal structure of sentences". Semantics refers to "the study of the meaning of sentences" (Bender, 2013).

The differences between these concepts can be illustrated with sentences [1] and [2]. The only difference lies in their last words.

[1]: The animal didn't cross the street because it was too tired.

[2]: The animal didn't cross the street because it was too wide.

The syntactic structure is obtained from the analysis of the text's syntax. As illustrated below, the syntactic structure is common between both sentences, with common phrase

structures comprising of common Noun Phrases (NP) and Verb Phrases (VP). The syntactic structure allows useful information to be extracted. For example, the entities referenced (e.g. the animal) and the predications about them (e.g. didn't cross).

[1]: The animal [NP] didn't cross [VP] the street [NP] because it [NP] was too tired [VP].

[2]: The animal [NP] didn't cross [VP] the street [NP] because it [NP] was too wide [VP].

The semantic meaning is obtained from the analysis of the text's semantics. It was illustrated earlier that the two sentences differed only by one word and shared the same syntactic structure. However, these sentences had very different meanings, with the first attributing the non-crossing to the animal's fatigue, while the second attributed the non-crossing to the width of the street. In essence, the replacement of the last word modified the meaning of the referent ("it"), giving the sentences a distinct meaning. Hence, semantic analysis is loosely described as understanding the text in context.

[1]: The animal didn't cross the street because it was too tired.

[2]: The animal didn't cross the street because it was too wide.

This contextual understanding is also important in the understanding of homonyms and polysemes. Homonyms refer to words with different meanings. For example, "bank" can refer to both the bank of a river or the bank as a financial institution[5]. Polysemes refer to words with different yet related meanings. For example, in the phrase "Let's get a drink", the word "drink" could adopt different meanings depending on the context created by adjacent statements. Within engineering, similar phenomena is observed where technical terms such as "bandwidth" can have different meaning to control engineers and computer engineers (Madni & Sievers, 2018).

3.1.2 Natural Language Processing Applications

NLP has been applied to solve a wide range of tasks. In all applications, the natural language text is taken in as an input and transformed into the desired output. The natural language text input is fundamentally a sequence - it contains constituent components (such as words and punctuations), and these components are arranged in a meaningful

[5] A related example encountered during this thesis involves the phrase "1830 battery". This can be understood as (1) 1830 [Quantifier] battery [Noun], or (2) 1830 battery [Noun]. In that context, the latter is the intended meaning as 1830 describes the model of the battery.

manner. However, the outputs take various forms. For checking of requirements quality, the output can be binary (acceptable or unacceptable) or continuous (on a scale of 1-5). For the generation of test cases from requirements documents, the output can be another natural language text (which is a sequence that is distinct from the original input). In this section, these applications will be organized into three broader classes: sequence classification, sequence labeling, and sequence-to-sequence. The concept behind each of these categories will be elaborated on. It should be noted that while these concepts are described without reference to specific techniques, it is implied that the more complex applications necessitate the use of more advanced ones.

3.1.2.1 Sequence Classification

Sequence classification seeks to transform the input sequence into a single output (Graves, 2012). Single output can refer to a single categorical class within multi-class classification problems or a continuous output variable. This class of applications are the most restrictive of the three classes due to the nature of its outputs. As shown in Table 3-1, functional requirement/ non-functional requirement (FR/NFR) classification (Hey et al., 2020), ambiguity detection in requirements (Ezzini et al., 2021), and quality assessment of requirements problems are all variants of sequence classification problems.

Table 3-1: Examples of Sequence Classification Applications in Requirement Engineering.

Use Case	**Input Sequence**	**Single Output**
FR/NFR Classification (Hey et al., 2020)	The system shall refresh the display every 60 seconds.	Functional/ ~~Non-Functional~~
	The search results shall be returned no later than 30 seconds after the user has entered the search criteria	~~Functional/~~ Non-Functional
Ambiguity in Requirements	Service availability shall measure the outage of LEO satellites and terminals.	Coordination Ambiguity[6] (Yes/~~No~~)

[6] Coordination Ambiguity (CA) describes ambiguity that arises from the use of a coordinating conjunction (i.e. "and" or "or"). In this example, the phrase "LEO satellites and terminals" creates ambiguity as it is unclear if the modifier "LEO" applies to element "terminals" (Ezzini et al., 2021).

Use Case	Input Sequence	Single Output
(Ezzini et al., 2021)	The outage management platform shall provide administrators with the ability to categorize outages with discrete tags.	Prepositional-phrase Attachment Ambiguity[7] (Yes/~~No~~)

3.1.2.2 Sequence Labeling

Sequence labeling seeks to label target segments within the input sequence based on a predefined labeling scheme (Graves, 2012). In NLP, target segments normally refer to words or a sequence of words[8].

Table 3-2: Examples of Sequence Labeling.

Sentence	My	name	is	Shao	and	I	live	in	Cambridge
POS	DET	NOUN	AUX	PROPN	CCONJ	PRON	VERB	ADP	PROPN
NER	O	O	O	PER	O	O	O	O	LOC

In Table 3-2, the following sentence, "My name is Shao and I live in Cambridge" is used to illustrate sequence labeling in the context of part-of-speech (POS) tagging and named entity recognition (NER). In both sequence labeling tasks, the sentence is used as the input sequence, with every word representing a target segment. In POS tagging, the objective is to assign every target segment to one of the pre-defined POS categories: determiner [DET], noun [NOUN], auxiliary [AUX], proper noun [PROPN], coordinating conjunction [CCONJ], pronoun [PRON], verb [VERB], adposition [ADP], adjective [ADJ], etc. In NER, the objective is similar but with some nuance. Unlike POS tagging where the POS categories are universal, named entities are context and domain dependent. In this example, the objective is to identify target segments that can be identified as persons [PER], location [LOC], or organization [ORG]. In NER tagging schemes, there is an additional empty token [O] used to categorize target segments which do not fall into any of the target categories. It is useful to note that NER tagging schemes also include modifiers to the target categories to categorize target segments based on their positions

[7] Prepositional-phrase Attachment Ambiguity (PPA) describes ambiguity that arises from the use of a PP. In this example, the term "outages" is the PP. The two resulting interpretations are: (1) to "categorize" "outages with discrete tags" or (2) to "categorize outages with" "discrete tags" (Ezzini et al., 2021).

[8] These segments can also be composed of sub-word tokens. The ideas of "sub-word" and "tokens" will be discussed subsequently in Section 3.3.3.

within multi-word named entities. The two common modifiers are the Inside-Outside-Beginning (IOB) and the Inside-Outside-Beginning-End-Single (IOBES) schemes. These schemes are illustrated with "Massachusetts Institute of Technology" in Table 3-3.

Table 3-3: Illustration of IOB and IOBES schemes.

Example	Massachusetts	Institute	of	Technology
IOB	B-ORG	I-ORG	I-ORG	I-ORG
IOBES	B-ORG	I-ORG	I-ORG	E-ORG

These sequence labeling tasks have been applied within the SE domain. For example, a recent survey found POS tagging to be the most applied technique in RE research (Zhao et al., 2021). NER was also applied for the extraction of named entities to populate SysML requirement tables (Riesener et al., 2021).

3.1.2.3 Sequence-to-Sequence

Sequence-to-sequence is the final class of application. Such applications seek to transform an input sequence into a separate output sequence. In the context of NLP, both sequences are natural language text. Table 3-4 highlights a set of sequence-to-sequence applications such as machine translation, text summarization and question-answering. The complexity of sequence-to-sequence applications is self-evident from these examples. First, for the outputs to be appropriate and coherent a representation of semantic meaning is required. Second, the absence of constraints on the output sequence dimensions poses unique architectural challenges on the choice of models. Lastly, the output sequence can be related but not represented within the input sequence.

Table 3-4: Illustration of Sequence-to-Sequence Applications.

Use Cases	**Input**	**Output**
Machine Translation	"How are you?"	"你好吗?"

Use Cases	Input	Output
Text Summarization (L. Wang & Ling, 2016)	"Joe Strummer: The Future Is Unwritten. The late punk rock legend Joe Strummer is rendered fully human in Julian Temple's engrossing and all-encompassing portrait. The movie fascinates not so much because of Strummer but because of the way Temple organized and edited the film. One of the most compelling documentary portraits of a musician yet made."	"Fascinating and insightful, Joe Strummer: The Future Is Unwritten is a thoroughly engrossing documentary."
Question-Answering	"What is the temperature today?"	"60 degrees Fahrenheit"

A survey of the literature did not identify any published research on sequence-to-sequence applications within systems engineering. However, there is some literature on the use of requirements for test case generation to support the verification and validation process (Moitra et al., 2019; Sinha et al., 2015)[9]. This presents a potential use case for sequence-to-sequence applications in systems engineering.

3.2 Symbolic Approaches to Natural Language Processing

Symbolic approaches work by "carrying out a series of logic-like reasoning steps over language representations" (Garnelo & Shanahan, 2019). The "logic-like reasoning" requires the language to be represented in human-readable formats, and rules decomposed and specified as a set of defined steps. As a result, these approaches are easily interpretable. Interpretability is an important ility that can lead to an increase in model confidence – an important trait in sociotechnical systems. However, the reliance of these models on hand-crafted representations represents a key weakness. First, as Polanyi's paradox suggests, "we know more than we can tell" - this implies that there is an inherent inability to translate the human understanding of knowledge fully into comprehensive rules, even if they exist. Second, this lack of a comprehensive set of rules means that input sequences that fulfil these rules will be correctly identified (high

[9] These studies were not classified as sequence-to-sequence applications as the test cases were generated from ontologies rather from requirements itself. As such, requirements only form part of the input sequence.

precision[10]). However, the sequences which do not fulfil these rules cannot be detected (low recall[11]). This creates an unknown unknown problem[12].

The remainder of this section will introduce two symbolic approaches: lexicon-based (Section 3.2.1) and rules-based (Section 3.2.2).

3.2.1 Lexicon-based Approaches

Lexicon-based approaches are based on the notion that individual words or word strings contain information that can be decoded from natural language (Guthrie et al., 1996; Wilks, 1993). In practice, lexicon-based approaches rely on machine readable dictionaries (MRD) of relevant lexicons to extract such information. The approach is premised on the intuition that much of our collective knowledge can be condensed and described by such libraries, making this a useful approach (Guthrie et al., 1996). When identifying systems or subsystems from requirement statements, the MRDs serve as look-up tables, assigning a Boolean (binary variable) to each word. When analyzing sentiment within a statement, the MRDs serve to assign a sentiment score (continuous variable) to relevant words.

These MRDs vary in naming and form, with variation observed across and within domains.

- **Systems Engineering** – MRDs are represented as ontologies within SE. A recent survey of known SE ontologies showed significant differences in the structure and properties captured within each ontology (Yang et al., 2019). In addition, the authors also found that the SE ontologies did not specify the methodology used in its generation, were described using natural language rather than represented using models[13], provided insufficient detail for it to be reproduced formally, and resultantly has low possibility of reuse (Yang et al., 2019).

[10] Precision describes the proportion of positive identifications that are correct. This is expressed mathematically as *[True Positive / (True Positive + False Positive)]*.

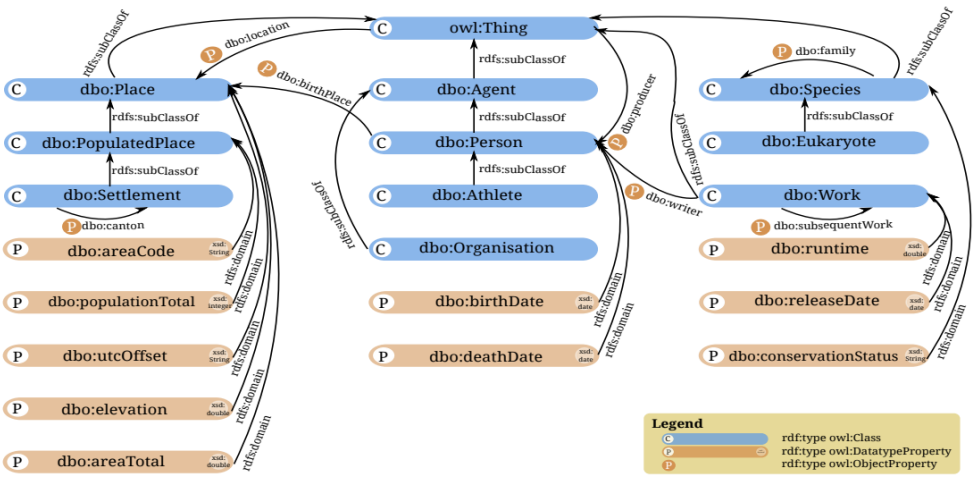
[11] Recall describes the proportion of actual positive identifications are correctly identified. This is expressed mathematically as *[True Positive / (True Positive + False Negatives)]*.

[12] Unknown unknowns refer to things that we are neither aware of nor understand.

[13] The authors of the survey stated that of the 57 papers describing SE ontologies, only 2 authors made their ontologies available for open access.

- **Generic** – These MRDs, which are referred to as both ontologies and gazetteers, are not directly useful for SE. However, the characteristics of such MRDs contrast starkly with those developed within SE, providing useful lessons for the SE community. These MRDs (Hamilton et al., n.d.; Lehmann et al., 2012; Manning et al., 2014) are formally specified, openly accessible, with methodologies for its generation specified. One example comes from the DBpedia project which maps Wikipedia infoboxes into a single shared ontology consisting of 685 classes and 2,795 properties (*DBpedia*, n.d.); a subset of this ontology is illustrated with Figure 3-1.

Figure 3-1: Ontology example from DBpedia (Lehmann et al., 2012).



To complete the above comparisons, it is important to acknowledge that the analysis is incomplete. As stated by the authors of the survey, SE ontologies serve other purposes such as enabling communication across disciplines and between stakeholders (Yang et al., 2019). Hence, comparing SE ontologies against MRDs that are specifically designed for model deployment is not a fair one. However, as the SE domain shifts towards DE, it is perhaps important to develop ontologies with model reuse and compatibility in mind.

3.2.2 Rule-based Approaches

Rule-based approaches seek to match the natural language corpus against predefined rules or patterns. This method is heavily influenced by the field of linguistics. There is

broad consensus that natural language exhibits many recurring rule-like structures, and these structures allow the elicitation of information (Spike, 2018). It is also acknowledged that many peculiarities in natural language cannot be described in the same manner (Spike, 2018). Therefore, this approach seeks to develop a finite set of rules that are sufficient to facilitate the extraction of information from natural language corpus. The prevalence of such rule-based approaches is evidence of their utility.

These rule-based patterns can be defined at the individual words level or based on a word sequence. At the word level, techniques such as regular expressions (RegEx) can be used to identify specific sentence compositions. For example, the combination use of "and" and "or" in proximity can lead to ambiguous association between subjects. This can be identified using the expression stated in Figure 3-2.

Figure 3-2: Use of RegEx for Ambiguity Detection in in Requirements (Gleich et al., 2010).

Regular expression	Matched ambiguity	Example
up to (?!including|excluding)	Up to with unclear inclusion	The system shall support up to five concurrent users.
everybody .* their, everybody .* its	Either language error or ambiguous plural	Everybody uses their login id.
^both	At the beginning of the sentence, both has a unclear reference	Both should be documented.
(all|each|every) .* (a|his|her|its|their|they)	Dangerous plural with ambiguous reference	Every student thinks she is a genius.
and .* or, or .* and	The combination of "and" and "or" leads to unclear associativity	The system shall read HTML and PDF or DOC files.

At the input sequence level, the use of part-of-speech (POS) patterns have also been used. This is illustrated with the example stated in Figure 3-3; the last five words of this requirement contains a prepositional-phrase attachment ambiguity (PAA) as it allows for two possible interpretations. This form of ambiguity can be detected using various POS patterns stated in Figure 3-4. Specifically, the last five words of the requirement fit the verb-noun-preposition-noun POS pattern.

Figure 3-3: Example of Ambiguity in Requirements due to Prepositional-phrase Attachment Ambiguity (Ezzini et al., 2021).

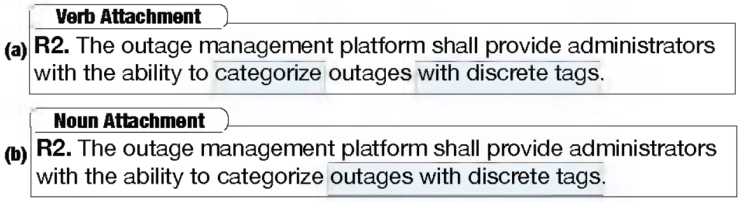


Figure 3-4: Use of Pattern Matching (Rules) for Ambiguity Detection in Requirements (Ezzini et al., 2021).

CA	1	nn n_1 c n_2	11	n_1 c n_2 p dt/adj nn	21	adv adj_1 c adj_2
	2	n_1 c n_2 nn	12	v_1 c v_2 nn	22	adj_1 c adj_2 adv
	3	nn p n_1 c n_2	13	nn p v_1 c v_2	23	adj_1 c adj_2 adj nn
	4	n_1 c n_2 p nn	14	v_1 c v_2 p nn	24	nn *dt* n_1 c *dt* n_2
	5	v n_1 c n_2	15	v to v_1 c v_2	25	nn *p dt* n_1 c *dt* n_2
	6	n_1 c n_2 v	16	v_1 c v_2 to v	26	*dt* n_1 c *dt* n_2 p nn
	7	nn n_1 c n_2 nn	17	adv v_1 c v_2	27	nn *dt* n_1 c *dt* n_2 nn
	8	adj n_1 c n_2	18	v_1 c v_2 adv	28	adj *dt* n_1 c *dt* n_2
	9	adj nn n_1 c n_2	19	v_1 c v_2 p dt/adj nn	29	*adj* nn *dt* n_1 c *dt* n_2
	10	adj adj n_1 c n_2	20	dt/adj nn p v_1 c v_2		

PAA	1	v n_1 *p* n_2	6	v n_1 *p dt adj* n_2
	2	v n_1 *p dt/adj* n_2	7	v *n* n_1 *p dt/adj* n_2
	3	v *dt/adj* n_1 *p* n_2	8	v *n* n_1 *p dt adj* n_2
	4	v *dt/adj* n_1 *p dt/adj* n_2	9	v *dt adj* n_1 *p* n_2
	5	v *n* n_1 *p* n_2	10	v *dt adj* n_1 *p dt/adj* n_2

n_1, n_2, nn: noun, **v**: verb, **adv**: adverb, **adj**: adjective, **dt**: determiner, **p**: preposition, **/**: or.

3.3 Machine Learning Approaches to Natural Language Processing

Frederick Jelinek, an automatic speech recognition pioneer and natural language processing researcher, was famously quoted saying: "Whenever I fire a linguist our system performance improves." Statistical and symbolic approaches fundamentally differ in the way they obtain their language representations. The former relies on hand-crafted, pre-specified representations, while the latter relies on learnt statistical representations. As such, machine learning approaches are a formal subset of statistical approaches.

Machine learning (ML) approaches employ three core ideas. First, inputs and outputs are translated into a mathematical form which can take the form of scalars, vectors, or tensors. Second, an objective function is developed to describe what is mathematically desirable. Third, a learning algorithm is used to learn the relations between inputs and outputs, resulting in the optimization of the objective function. For NLP, the first step, therefore, requires natural language text to be represented in an explicit mathematical form, an approach that is not seen in the previous approaches. In Section 3.3.1, various forms of representations are outlined in increasing complexity and abstraction.

3.3.1 Vectorized Representations of Natural Language

The way natural language text is translated into a mathematical form determines the information that is retained in that representation. Information that is not captured within that representation by extension cannot be used by the learning algorithm.

3.3.1.1 One-Hot Word Representations

One-hot representations are used to indicate the presence or absence of a feature. For a given set of vocabulary $\{x_1, x_2, \ldots, x_n\}$, where n is the size of the vocabulary, each element (x_i) can be represented in a sparse n x 1 vector, where only 1 element can be 1, with others 0. This can be illustrated with the sentence: "The animal didn't cross the street because it was too tired."

Table 3-5: Example of One-hot Vectors.

Word	One-hot Vector		Word	One-hot Vector
The	[1 0 0 0 0 0 0 0 0 0 0 0 0]		The	[1 0 0 0 0 0 0 0 0 0 0 0 0]
animal	[0 1 0 0 0 0 0 0 0 0 0 0 0]		animal	[0 1 0 0 0 0 0 0 0 0 0 0 0]
didn't	[0 0 1 0 0 0 0 0 0 0 0 0 0]		didn't	[0 0 1 0 0 0 0 0 0 0 0 0 0]
cross	[0 0 0 1 0 0 0 0 0 0 0 0 0]		cross	[0 0 0 1 0 0 0 0 0 0 0 0 0]
the	[0 0 0 0 1 0 0 0 0 0 0 0 0]		the	[0 0 0 0 1 0 0 0 0 0 0 0 0]
street	[0 0 0 0 0 1 0 0 0 0 0 0 0]		street	[0 0 0 0 0 1 0 0 0 0 0 0 0]
because	[0 0 0 0 0 0 1 0 0 0 0 0 0]		because	[0 0 0 0 0 0 1 0 0 0 0 0 0]
it	[0 0 0 0 0 0 0 1 0 0 0 0 0]		it	[0 0 0 0 0 0 0 1 0 0 0 0 0]
was	[0 0 0 0 0 0 0 0 1 0 0 0 0]		was	[0 0 0 0 0 0 0 0 1 0 0 0 0]
too	[0 0 0 0 0 0 0 0 0 1 0 0 0]		too	[0 0 0 0 0 0 0 0 0 1 0 0 0]
tired	[0 0 0 0 0 0 0 0 0 0 1 0 0]		wide	[0 0 0 0 0 0 0 0 0 0 0 0 1]

The most intuitive one-hot encoding uses single words as the minimum unit (as shown in Table 3-5); this is commonly referred to as a bag-of-words (BOW). In this example, the word "animal" is represented by the same vector in both sentences. This means that these vectors only capture the mere presence of these words, without any accompanying syntactic or semantic information. The notion that the "animal" was "tired" or the notion that the "animal didn't cross the street" are both omitted. The following gibberish sentence, "The tired cross didn't because too street it was the" will return the same representation as the original sentence because the constituent words remain the same. The other weakness of this representation is due to the dimensionality of the vocabulary. The natural language vocabulary is vast, but the frequency of words used in natural language is extremely long-tailed. For example, the use of pronouns ("I", "he", "she", "they"), determiners ("this", "the", "my"), and auxiliaries ("may", "can", "be") outstrip the usage of rare words (such as "rapscallions" or "zeugma") by orders of magnitude. However, each of these words occupies a dimension regardless of their frequency of use. As such, as a practical approach, the size of the vocabulary is often capped, and rare words are encoded as a placeholder token (commonly denoted as a [UNK]). While this conclusion may appear to preclude its application within domains with a niche vocabulary, the computation efficiency of this representation allows vocabularies and vectors to be generated from scratch for a given body of text.

This method is commonly extended in two ways to tackle each of the two main weaknesses. The first extension is called the n-gram; this method seeks to encode some sequence information by capturing the word sequences up to length n.

Table 3-6: Illustration of a n-gram.

1-word tokens	2-word tokens
"Boston", "is", "in", "New", "England"	"Boston is", "is within", "within New", "New England"

The intuition can be explained with the following sentence: "Boston is within New England" (Table 3-6). Using $n = 2$, the 2-gram (or commonly known as a bi-gram) encodes in additional information such as "New England". In comparison, the bag-of-words method would have associated "Boston" with "England", while the bi-gram allows the association to be made with "New England" across the Atlantic. However, this method aggravates the vocabulary size limitation due to the addition of n-length tokens.

The second extension involves character-level encoding. This directly tackles the vocabulary size limitation. For example, the word "Systems" will be encoded as {"s", "y", "s", "t", "e", "m", "s"}. As a result, the vocabulary size is directly capped at 70, comprising of the 26 letters in the English alphabet, 10 digits and 33 special characters. This allows any word to be represented with a finite sized vocabulary. However, this also breaks the information encoded in both the sentence and the words themselves. While this approach may appear to be counter-intuitive, there is empirical evidence of its effectiveness (Jozefowicz et al., 2016; Zhang & LeCun, 2015).

3.3.1.2 Distributed Word Representations

Distributed word representations are founded on the distributional hypothesis. In essence, the hypothesis states that "there is a correlation between distributional similarity and meaning similarity" (Sahlgren, 2008). In simpler terms, this means that linguistic elements which have similar behaviors have similar meanings. As such, semantic meaning can be inferred by virtue of their distributional behavior. In vector space, words with similar meanings can therefore be viewed as clusters. Two key representations will be outlined: GLoVe (Pennington et al., 2014) and Word2Vec (Mikolov et al., 2013). Unlike Section 3.3.1.1, these representations will not be elaborated in full. Within this thesis, the key ideas to be retained relate to its utility and implications rather than its mathematical workings.

GLoVe and Word2Vec produce word vectors. Each word has a unique vector that is a dense representation of its meaning. This is a dense representation because the dimension of the word vectors greatly differs from one-hot encoded variants. For illustration, we can compare the vectors for the word "bank" (Table 3-7). Two observations are immediately apparent. First, the GLoVe vectors are much lower in dimension, with <1% of the elements. However, information is densely encoded within each of the 50 elements, as compared to the one-hot encoding, where information is only encoded by one element. This also illustrates a weakness - such dense representations are not interpretable.

Table 3-7: Comparison of BOW and GLoVe Vectors for the word "bank".

Representation	Dimension	Word Vector
Bag-of-words (30,000-word vocabulary)	(30000, 1)	[0, 0, …., 1, 0, 0, …, 0, 0], where only 1 element = 1

Representation	Dimension	Word Vector
GLoVe[14] (50-dimension[15])	(50, 1)	[0.66488, -0.11391, 0.67844, 0.17951, 0.6828, -0.47787, -0.30761, 0.17489, -0.70512, -0.55022, 0.1514, 0.10214, -0.45063, -0.33069, 0.056133, 1.2271, 0.55607, -0.68297, 0.037364, 0.70266, 1.9093, -0.61483, -0.83329, -0.3023, -1.1118, -1.55, 0.2604, 0.22957, -1.0375, -0.31789, 3.5091, -0.25871, 1.0151, 0.65927, -0.18231, -0.75859, -0.30927, -0.91678, 1.0633, -0.66761, -0.37464, -0.29143, 0.65606, -0.44642, -0.075495, -1.0552, -0.60501, 0.73582, 1.0139, -0.27749]

These dense representations are generated by learning relations between words from an extremely large corpus of text. For example, GLoVe was trained on several datasets consisting of 55 billion tokens. Word2Vec was trained on an internal Google data set of 100 billion words. This allows the meaning of each word to be learned from a diverse set of uses contained within the text corpus. Hence, the robustness and utility of such representations are premised on the text corpus that it is trained on. As a result, these representations cannot be purposefully generated from scratch without a comparably sized text corpus for training. These models suffer several weaknesses. First, it aggregates the meaning of words into a single vector - for example, the meaning of "bank" is context-dependent. However, it is represented as a single vector regardless of the intended use. Second, these models lack the ability to represent the meaning of niche domain-specific vocabulary. As every word is represented as a unique vector, there is a finite number of word vectors covering the most common textual elements, while niche expressions are represented by a generic out-of-vocabulary (OOV) token[16].

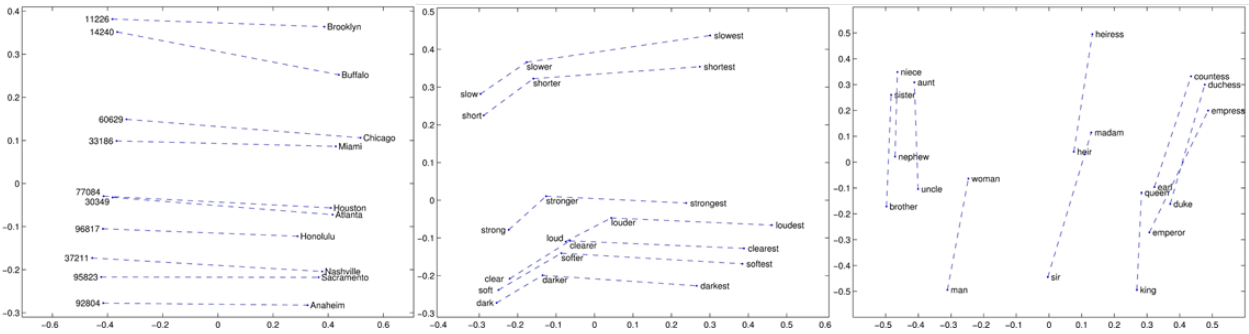
GLoVe stands for Global Vectors for Word Representation, and is trained to capture the corpus statistics directly. At a high level, it determines the co-occurrence probabilities that each pair of words will co-occur in a natural language text (Pennington et al., 2014). For example, consider three words: "ice", "water" and "steam". We could expect "ice" to be more strongly related and therefore more likely to co-occur with "water". This relation will be weaker between "ice" and "steam". As such, "ice" is more distributionally similar to "water" than to "steam".

[14] A notebook showing the extraction of GLoVe and Word2Vec vectors is available here.

[15] GLoVe vectors are available in many dimensions n = 50, 100, 200, 300.

[16] There is an extension of Word2Vec developed by Facebook, FastText, which uses sub-word representations to handle OOV words. However, this will not be discussed in this chapter.

Figure 3-5: Visualization of GLoVe Vectors (Pennington et al., 2014).



This notion of distributional and meaning similarity can be visually observed from the word vectors (Figure 3-5). Each of these diagrams was generated through dimensionality reduction to reduce the high-dimensional vector into 2-dimensions. From left to right, the resulting text vectors were shown to make associations between zip codes and their cities, capture comparatives, as well as distinguish man-woman references (Pennington et al., 2014).

Word2Vec consists of two models which are trained on different modes: continuous bag-of-words (CBOW) and skip-gram. The mathematical workings are based on neural networks and significantly differ from GLoVe. However, the similarities and associations illustrated in Figure 3-5 have also been shown in Word2Vec.

3.3.1.3 Contextualized Word Representations

From Sections 3.3.1.1 and 3.3.1.2, we observed increasing amounts of information being encoded from natural language text into numerical representations. However, it should be noted that the earlier representations do not materially encode the sequential nature of natural language. The absence of sequential information also means that context cannot be accurately encoded. These shortfalls are addressed by this final class of contextualized word representations. In line with earlier elaborations, this will be described with less mathematical details but supported with more illustrations.

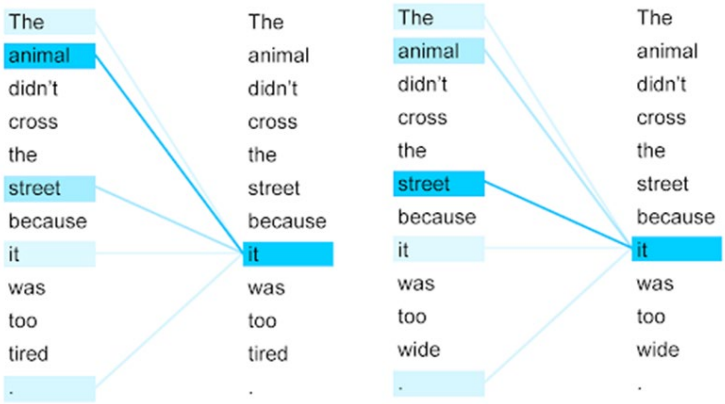
There have been multiple deep learning-based models introduced in recent years. ELMo (Peters et al., 2018) was developed based on a bidirectional LSTM architecture, while BERT (Devlin et al., 2018) and GPT-2 (Radford et al., 2019) were developed based on the Transformers architecture (Vaswani et al., 2017). The key idea within the Transformers

architecture is the self-attention mechanism - the training process using this mechanism to learn the key relationships between different linguistic entities within the input sequence. This is best illustrated by re-visiting the example cited in Section 3.1.1 - it was shown that the semantic meaning of "it" differs between these two sentences.

[1]: The animal didn't cross the street because it was too tired.

[2]: The animal didn't cross the street because it was too wide.

Figure 3-6: Contextual representation of "it".



The ability to distinguish this was demonstrated by Google with Figure 3-6 (Uszkoreit, 2017). When the encoder layers were analyzed, the following associations between the referent "it" and the linguistic entities ("animal" and "street") were observed. However, it should be noted that this illustration does not imply that transformer architecture is interpretable. For example, analysis of other encoder layers within the model returns associations that are difficult to rationalize or reason with. A subsequent study confirmed representations from ELMo, BERT and GPT-2 were all highly contextual and is responsible for successive state-of-the-art performance in established NLP benchmark tasks (Ethayarajh, 2020)[17].

Another issue that plagues other forms of representations lies in the handling of OOV words. ELMo overcomes this with character-level embedding through the mechanism explained in Section 3.3.1.1. BERT and GPT-2 both implement sub-word tokenization

[17] Ongoing research shows that many of these models exhibit concerning attributes such as bias against gender or race. These issues are not called out explicitly as they have little relevance to the systems engineering use cases being considered. Reader(s) are encouraged to familiarize themselves with ongoing debates.

approaches, albeit with different but related algorithms. This allows OOV words to be decomposed into a series of sub-word tokens.

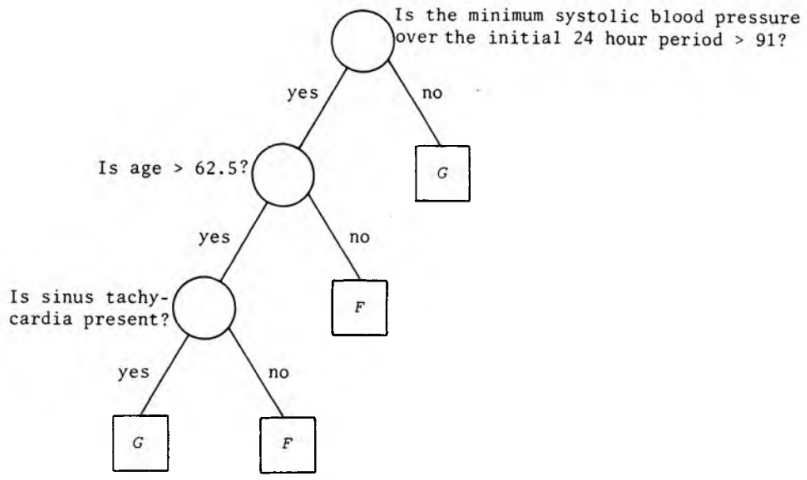
3.3.2 Traditional Machine Learning Approaches

In the introduction of Section 3.3, it was stated that ML models require a numerical representation of the input and output, an objective function, and a learning algorithm to optimize the objective function. Although the remaining two components are tightly coupled, the emphasis of the next two chapters is on the learning algorithm, with implicit discussion of the associated objective functions.

In the constituent Sections of 3.3.1, there was emphasis on the ability of various representations to encode syntactic and semantic information. However, to ensure such information is retained, the architecture of the learning algorithm similarly allows the syntactic or semantic information to be preserved.

As an example, classification and regression trees (CART) is a well-understood learning algorithm which employs a greedy heuristic to segment the training data into a tree-like structure. This method can be employed for classification (prediction of a categorical output) and/or regression (prediction of a continuous variable).

Figure 3-7: Illustration of CART (Breiman et al., 2017).



In the NLP context, the nature of the output appears to be a possible tool for sequence classification applications (recall Section 3.1.2). However, as CART only accepts scalar

inputs, all vectorized inputs must be flattened into individual scalar values. As a result, any information encoded in the vectorized form is not retained by the learning algorithm.

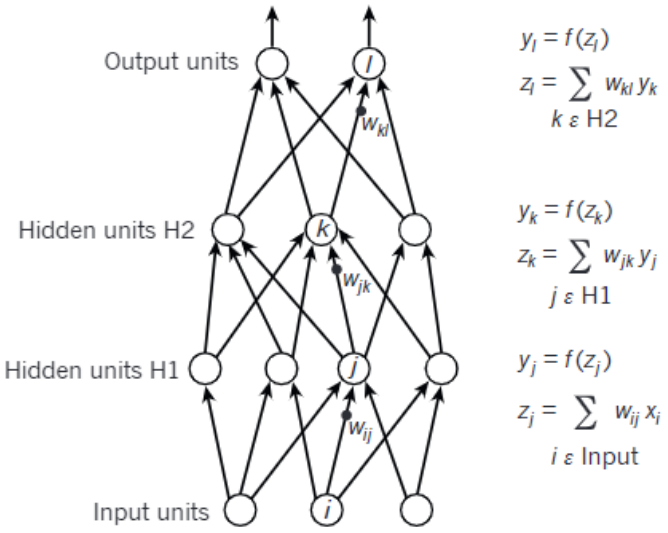
The inability to handle sequential data is a universal weakness of traditional machine learning approaches as well as simple neural network architectures (such as dense feedforward networks). Hence, the applications of traditional machine learning techniques within natural language processing typically involve feature engineering or use of one-hot (or multi-hot) representations for textual inputs.

Despite these limitations, traditional ML algorithms continue to be widely applied include naïve bayes (Song et al., 2009) and support vector machines (Joachims, 1998). Within the SE domain, traditional machine learning algorithms have also been applied for sequence classification applications (Binkhonain & Zhao, 2019).

3.3.3 Deep Learning Approaches

Earlier sections have reiterated the difficulty in learning representations of natural language. For instance, rule-based approaches rely heavily on hand-crafted features. However, the effectiveness of such approaches is hamstrung by nuances and idiosyncrasies that may not be stated plainly.

Figure 3-8: Example of a Multilayer Neural Network (LeCun et al., 2015).

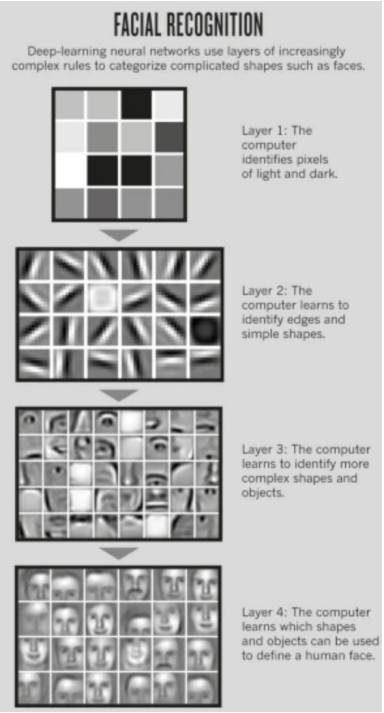


Deep learning overcomes this using a multi-layered neural network, comprising of input, hidden and output layers. These layers work in concert to learn representations that map inputs to outputs, as well as representations of the input itself (Bengio et al., 2015).

The intuition behind how the multi-hidden layer architecture learns such representations is illustrated with a facial detection example in Figure 3-9. In this example, the first hidden layer (Layer 2) learns to detect a series of edges and simple shapes. The second layer (Layer 3) combines the edges and shapes to identify facial features (eyes, jaw, etc). The third hidden layer (Layer 4) then combines the facial features into parts of faces. This verbose explanation is intended to outline the ability of deep learning to learn representations by combining simpler ones. In other words, complex concepts such as natural language are learnt from simple concepts captured within input text sequences.

However, the exact way natural language is encoded by the Transformer architecture is only understood in a broad manner (Rogers et al., 2020). For instance, it is known that the earliest layers contain information about linear word order, while middle layers contain information about syntactic information (such as subject-verb agreement), while the final layers are specific to the task that the LM is trained on (Rogers et al., 2020). There is also insufficient research on how semantic information is encoded (Rogers et al., 2020).

Figure 3-9: Illustration of representation learning in a multi-layer network (Jones, 2014).



3.3.3.1 Neural Network Architectures

Deep learning comprises of a wide range of neural network architectures designed for different applications and domains. In NLP, the preservation of sequence is clearly important. Without sequence, "cat sat on the mat" and "mat sat on the cat" are equivalent. In NLP, the common sequence architectures are recurrent neural networks (RNN), gated recurrent units (GRU), long short-term memory networks (LSTM), and Transformers. Another related architecture choice involves the choice of autoregressive (or unidirectional) or bidirectional networks. As a comparison, the three models detailed within Section 3.3.1.3 have different architectures - ELMo is based on a bidirectional-LSTM, BERT is based on a bidirectional Transformer architecture, while GPT-2 is based on an autoregressive Transformer architecture.

3.3.3.2 Learning from Scratch vs Transfer Learning

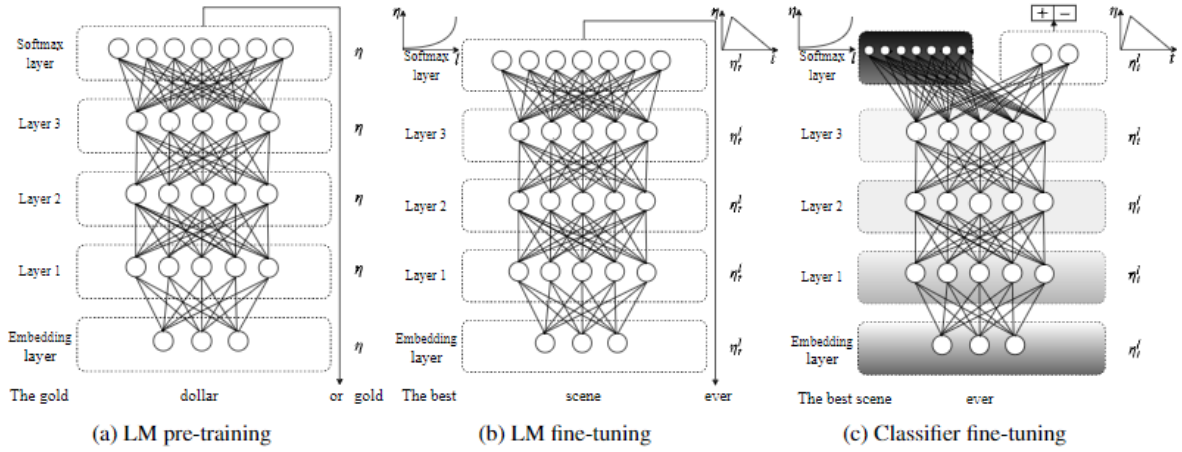
The NLP applications (Section 3.1.2) require the mapping of an input text to an application-dependent output. In ML, this is typically achieved through supervised learning. The learning algorithm is provided with labeled input-output pairs to learn representations between inputs and outputs. While this training scheme is typical for conventional ML algorithms, this poses significant challenges for deep learning.

The increasing performance of deep learning models has been attributed to three key factors: significant amounts of data, hardware improvements, and new algorithmic ideas. A large amount of training data allows deep learning models to be trained on representative datasets allowing the representations to be accurate and robust (Bengio et al., 2015). However, generating relevant labeled training data is hugely expensive as it requires human annotators (Aroyo & Welty, 2015). The increase in computational performance has enabled the use of large modern deep learning models - the large models allow representations to be learned more effectively. As a natural consequence, significant computational resources are required to achieve competitive model performance. Taken together, supervised learning from scratch is a challenging endeavor in most deep learning applications.

This however can be mitigated by inductive transfer learning. Inductive transfer learning refers to the transfer of knowledge from one domain into a different one. In NLP, this means that representations that a deep learning model learns from a specific natural

language task can be transferred onto a different task. This transfer of knowledge significantly reduces the requirement for both training data and computational resources.

Figure 3-10: Training Scheme Proposed in ULMFiT (Howard & Ruder, 2018).



Universal Language Model Fine-tuning (ULMFit) (Howard & Ruder, 2018) outlined an approach (Figure 3-10) which facilitated the adoption of transfer learning within NLP. This comprises of three stages (summarized in Table 3-8): LM pretraining, LM fine-tuning and classifier fine-tuning.

Table 3-8: Summary of Pretraining and Fine-tuning Stages.

Stage	Require Labeled Data	Type of Training Data	Size of Training Data	Training Mechanism
LM pretraining	No	General	Massive	Self-supervised
LM fine-tuning (DAPT)	No	Domain-specific	Large	Self-supervised
LM fine-tuning (TAPT)	No	Task-specific	Small	Self-supervised
Classifier fine-tuning	Yes	Task-specific	Small	Supervised

The LM pretraining is performed using general-domain data. This stage uses a massive text corpus[18] using self-supervision to learn representations of natural language. Self-supervision is a process where the model can learn representations without the need for any labeled data, alleviating the data annotation cost. However, due to the model size and the quantity of training data, this stage is the most computationally expensive.

[18] The ULMFiT model was trained on 28,295 Wikipedia articles and 103 million words from the Wikitext-103 dataset.

Contextual word representations (ELMo, BERT and GPT-2) are examples of such pre-trained models.

The next step, LM fine-tuning, performs fine-tuning on LM using domain-specific text corpus. Within the literature, this stage is described using other terms such as mixed domain pretraining (Gu et al., 2021). Like the first stage, fine-tuning is performed with unlabeled data using self-supervision. This enables the LM to adapt the representations from a generic-domain one to a domain-specific one. This stage has been further divided by researchers into two complementary phases: domain-adaptive pretraining (DAPT) and task-adaptive pretraining (TAPT) (Gururangan et al., 2020). The difference between these phases lies in the domain-specific text corpus used; the former uses generic domain-specific data while the latter uses task-specific data. In the requirements engineering context, DAPT can be performed using engineering journal papers, while TAPT should be performed using requirements datasets only.

In the final stage, classifier fine-tuning, an additional classifier or regression head is added to the pre-trained model to enable the language model to be fine-tuned to the specific natural language task. The classifier or regression head consists of fully connected feed-forward neural networks.

3.3.4 Foundation Models

The ability to pre-train and fine-tune contextualized word representations to new application domains has shifted the landscape in NLP. The use of BERT is no longer novel and has become the new normal. Many performance enhancements achieved by variants of BERT (such as RoBERTa, BART, DistilBERT) which sought to address specific weaknesses of BERT (Rogers et al., 2020).

However, ELMo, BERT, GPT-2 and their analogs have been overtaken by a different class of foundation models such as GPT-3 (Brown et al., 2020), GLaM (Du et al., 2021), and Gropher (Rae et al., 2022). The number of model parameters have grown by orders of magnitude, with associated increases in computational cost. The key benefit of these models was the ability to perform many NLP tasks with little (few-shot classification) to no fine-tuning (no-shot classification). These models have also displayed emergent behavior that was unexpected (Bommasani et al., 2021). For example, GPT-3 exhibited in-context learning, where "the language model can be adapted to a downstream task simply by providing it with a prompt".

This mention of foundation models is intended to illustrate the rate of development in NLP - Transformers architecture was introduced in 2017, followed by BERT in 2018, and foundation models in 2020. NLP research is a speeding train that should be leveraged on to advance adoption within SE.

3.4 Domain-Specific Natural Language Processing

The discussions preceding this section draws from NLP research in the generic domain, with a few asides to discuss the impact of domain-specific factors. However, as a technological goal, it is perhaps more important to understand what stands between transfer of generic domain NLP technologies into the systems engineering use case. This section seeks to draw on lessons from other domains.

3.4.1 Domain-specific factors

Generic-domain text has been the feedstock for NLP research due to the ease of accessibility and abundance. However, the generic-domain text (also referred to as Plain English) differs significantly from domain-specific text in the linguistic features they exhibit (Gray, 2013). In extreme cases, domain-specific uses, such as in the legal domain, have been classified as a sublanguage (Kittredge & Lehrberger, 1982). In this section, some of the salient factors and their associated implications are outlined.

3.4.1.1 Syntax

"In the event of any sale of such interest or transfer of such rights and upon the assumption, in writing, of the obligations of Landlord under this Lease by such assignee or transferee, Landlord herein named (and in case of any subsequent transfer, the then assignor) shall be automatically freed and relieved from and after the date of such transfer of all liability in respect of the performance of any of Landlord's covenants and agreements thereafter accruing, and such transferee shall thereafter be automatically bound by all of such covenants and agreements, subject, however, to the terms of this Lease; it being intended that Landlord's covenants and agreements shall be binding on Landlord, its successors and assigns, only during and in respect of their successive periods of such ownership." (Shaghaghian et al., 2020)

Legalese, the formal and technical legal language, is constructed to ensure that terms and grammatical constructions cannot be misinterpreted. Its syntax is highly complex, precise, and unique that non-legal professionals struggle to comprehend (Kittredge & Lehrberger, 1982). The extract above is clearly different from what is considered generic-domain text - if this was not yet noticed, the entire passage is a single 129-word sentence. This syntax also means that semantic relations show very long-term dependencies (Tagarelli & Simeri, 2021). In contrast, an example of a system requirement from the James Webb Telescope project shows highly regular syntactic structures.

"*The Observatory coordinate system axes are labeled J1, J2, and J3. This system is a right-handed, observatory body fixed system, with its origin located at the center of the LV-to-Observatory interface ring. The J1 and J2 axes are on the interface plane, with the J1 axis pointing in the direction of the OTE boresight. The J3 axis is perpendicular to the LV-to-Observatory interface plane, with its positive direction oriented towards the Observatory. Figure 3-3 illustrates this system.*" (Bogenberger, 2007)

The implication of a domain-specific syntax is rather fundamental as it makes the use of certain approaches (such as rule-based) less feasible. It could also imply that representations learnt from generic domain text may be less transferrable. However, at this point, the impact of syntax cannot be isolated and quantified.

3.4.1.2 Vocabulary

All domains use specialized vocabularies, even though the size and uniqueness of these vocabularies differs. The presence of these vocabularies alone does not pose significant issues - for instance, BOW (Section 3.3.1.1) allows a word representation to be generated efficiently for every term. However, in the era of deep learning and transfer learning, the contextualized word representations (Section 3.3.1.3) used in BERT and GPT-2 have a far more restricted vocabulary. These models represent rare, or OOV, words with combinations of sub-word tokens. This is illustrated in Figure 3-11 and Figure 3-12.

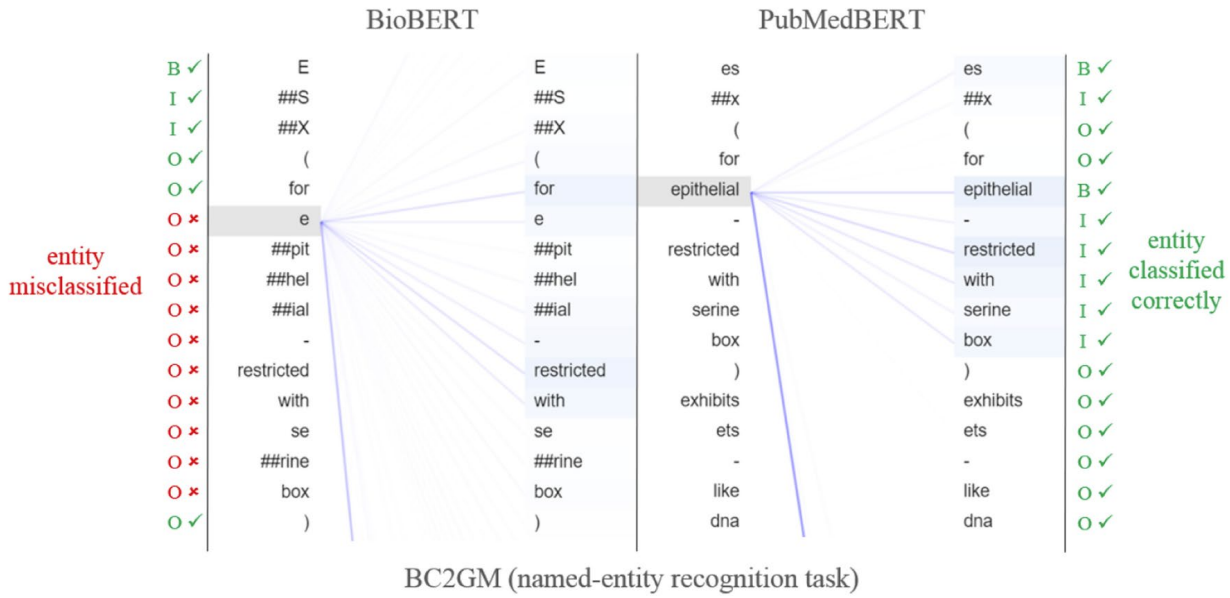
Figure 3-11: Comparison of Word Representations in across BERT variants (Gu et al., 2021).

Biomedical Term	Category	BERT	SciBERT	PubMedBERT (Ours)
diabetes	disease	✓	✓	✓
leukemia	disease	✓	✓	✓
lithium	drug	✓	✓	✓
insulin	drug	✓	✓	✓
DNA	gene	✓	✓	✓
promoter	gene	✓	✓	✓
hypertension	disease	hyper-tension	✓	✓
nephropathy	disease	ne-ph-rop-athy	✓	✓
lymphoma	disease	l-ym-ph-oma	✓	✓
lidocaine	drug	lid-oca-ine]	✓	✓
oropharyngeal	organ	oro-pha-ryn-ge-al	or-opharyngeal	✓
cardiomyocyte	cell	card-iom-yo-cy-te	cardiomy-ocyte	✓
chloramphenicol	drug	ch-lor-amp-hen-ico-l	chlor-amp-hen-icol	✓
RecA	gene	Rec-A	Rec-A	✓
acetyltransferase	gene	ace-ty-lt-ran-sf-eras-e	acetyl-transferase	✓
clonidine	drug	cl-oni-dine	clon-idine	✓
naloxone	drug	na-lo-xon-e	nal-oxo-ne	✓

A ✓ indicates the biomedical term appears in the corresponding vocabulary; otherwise, the term will be broken into word pieces separated by a hyphen. These word pieces often have no biomedical relevance and may hinder learning in downstream tasks.

A recent study in the biomedicine showed that many common biomedical terms did not feature in the BERT vocabulary and were recomposed with sub-word tokens (Gu et al., 2021). For example, “hypertension” was recomposed of “hyper” and “tension”, while acetyltransferase was recomposed of seven sub-word tokens (“ace”, “ty”, “lt“, “ran”, “sf”, ”eras”, “e”). This meant that the contextual meaning of such rare words was similarly recomposed from a series of unrelated tokens. The impact of this was not only theoretical.

Figure 3-12: NER misclassification resulting from sub-word tokenization (Gu et al., 2021).



BC2GM (named-entity recognition task)

This is illustrated in Figure 3-12 using a NER task. NER, as a sequence labeling problem, involves the assignment of a target label to every token. In this example, "epithelial" was the target entity to be classified. In the BioBERT model, a BERT model adapted for biology, the "epithelial" was divided into four sub-word tokens ("e", "pit", "hel", "ial"). When framed as a sequence labeling problem, it required all four sub-word tokens to be correctly labeled to the target class for it to be correctly identified as an entity. In contrast, "epithelial" was included in the PubMedBERT[19] vocabulary, allowing it to be processed as a single "token". This provides empirical evidence that the underlying generic-domain vocabulary can negatively impact downstream applications. It is therefore reasonable to posit that a higher mismatch between the domain vocabulary and the LM vocabulary will lead to poorer model performance.

It is useful to note that this vocabulary mismatch problem has been studied across multiple domains in relation to Word2Vec and GLoVe (Section 3.3.1.2). This resulted in numerous variants of domain-specific distributed word representations in engineering (Braun et al., 2021; Efstathiou et al., 2018), biomedicine (Y. Wang et al., 2018), and law (Chalkidis & Kampas, 2019; Dhanani et al., 2022).

[19] PubMedBERT is a custom BERT-based language model produced by the research.

There are two reported strategies to overcome this mismatch problem. First, the existing vocabulary of existing models can be extended with tokens of the new domain (Webersinke et al., 2021). Thereafter, the model can be fine-tuned in two steps to perform the intended task. However, the literature today only illustrates a proof-of-concept without sufficient analysis how the additional tokens should be chosen, and how the associated training scheme to balance between domain fine-tuning and catastrophic forgetting[20]. This presents difficulty in implementing this strategy. Second, a new LM can be pre-trained from scratch using a domain specific vocabulary and sub-word tokenization scheme. This approach is well understood but computationally expensive to execute.

3.4.2 Lessons from Biomedicine

Biomedicine is one of the leading domains for AI and NLP application, with the biomedical AI market sized at US$ 6.9 billion in 2021 and expected to grow by 10x to US$ 67.4 billion in 2027, at a CAGR[21] of 46.2% (MarketAndMarkets, 2021). However, this was not case just over a decade ago. In 2011, researchers articulated six barriers that inhibited NLP (and by extension AI) application within the biomedical domain highlighted - lack of access to shared data, lack of annotated datasets for training and benchmarking, insufficient common conventions and standards for annotations, lack of reproducibility, lack of collaboration, and lack of user-centered development and scalability. Unlike domain-specific issues (Section 3.4.1) that could be solved with technical solutions, many of the barriers that were identified were social-technical in nature. Today, some of these barriers have eroded, while the progress on others is less verifiable (Table 3-9).

[20] Catastrophic forgetting describes the phenomenon where the fine-tuning changes the parameters of the LM so significantly that it forgets (overrides) the learnt language representations.
[21] Compounded Annual Growth Rate (CAGR)

Table 3-9: Reviewing Barriers for NLP Adoption.

Barriers	Verifiable Artefacts in 2022	Released
Lack of access to shared data	Breast Cancer Screening AI - Contention occurred between Google Health researchers and other medical researchers over access to anonymized datasets and reproducibility. The ensuing exchanges published on *Nature* indicated that these issues remain a work in progress (Haibe-Kains et al., 2020; McKinney, Karthikesalingam, et al., 2020; McKinney, Sieniek, et al., 2020).	N.A
Lack of reproducibility		N.A
Lack of annotated datasets for training and benchmarking	BLURB - Biomedical NLP benchmark with 13 datasets in 6 tasks (Gu et al., 2021)	2020
	Public repository of 400 manually curated and annotated biomedical and clinical datasets (Blagec et al., 2022)	2021
Insufficient common conventions and standards for annotations	MetaMap - Standardized conventions for biomedical map pioneered by the National Library of Medicine (Aronson, 2001)	2001
Lack of collaboration	Big Tech role in biomedicine AI research - Google Health. Amazon Comprehend Medical, Microsoft's AI for Health is a clear indication of research expanding beyond academic research silos.	N.A
Lack of user-centered development and scalability	Unable to assess	N.A

Returning from a detour into the history of biomedical AI applications, it is useful to assess the architectures that underpin state-of-the-art NLP models in biomedicine. From the discussions in Sections 3.3.3 and 3.3.4, it is unsurprising that LMs based on BERT and its derivatives sit at the top of the leaderboards (Gu et al., 2021; Minaee, 2021).

3.4.3 Lessons from Legal

In comparison to biomedicine, the market for legal AI is approximately 20x smaller - sized at US$ 0.32 billion in 2019 and expected to grow to US$ 1.24 billion in 2024, at a CAGR of 31.3% (MarketsAndMarkets, 2019). As a second point of comparison, the 2019 research and development (R&D) budgets of Boeing and Airbus stand at US$ 3.2 billion and US$ 3.15 billion [22] respectively, approximately 10x of the entire legal AI industry. Therefore, applications of NLP within the legal domain provide a vignette of the

[22] EUR 2.816 billion at 2019 exchange rates (Average EUR/USD exchange rate of 1.1196)

possibilities even without sizeable investments. In other words, it illustrates the significant democratization of AI.

A literature survey of NLP applications in the legal domain provides several observations. First, the volume of research surveyed appears to be generated by independent researchers; it was difficult to identify researchers or institutions of eminence. Second, there is rapid and widespread adoption of LM for legal NLP applications within the research community. For instance, in the Competition on Legal Information Extraction and Entailment (COLIEE) 2019, an annual legal NLP competition for researchers, all submissions gravitated towards BERT-based approaches or ensemble approaches (including BERT) after the introduction of BERT (Rabelo et al., 2020). The subsequent competitions in 2020 and 2021 showed similar trends (Rabelo et al., 2021, 2022). Third, by 2022, various research groups have developed independently pre-trained and fine-tuned various BERT-based models to assess the best means to adopt LMs within the legal domain (Chalkidis et al., 2020; Ha Thanh & le Minh, 2021; Shaghaghian et al., 2020; Zheng et al., 2021). These observations indicate that state-of-the-art technologies from machine learning are rapidly transfused into NLP research within the legal domain.

3.5 Accelerating NLP Adoption in Systems Engineering

In a recent paper, a legal NLP researcher stated that "legal professionals often think about how to solve tasks from rule-based and symbol-based methods, while NLP researchers concentrate more on data-driven and embedding methods" (Zhong et al., 2020). However, I find this quote a more appropriate description of the state of NLP research in SE and RE. A 2021 mapping study indicates that the majority of RE research continues to rely on statistical and rule-based methods (Zhao, Ferrari, et al., 2021). In the same study, the authors stated that: "Our initial investigation suggests that most long tail NLP techniques are **nascent**[23], so their application in NLP4RE[24] might be forthcoming. For example, various deep learning techniques such as Word Embedding, Doc2Vec, LSTM[25], CNN,

[23] The emphasis placed on the word "nascent" was made by the paper's authors.

[24] Natural Language Processing for (4) Requirements Engineering

[25] LSTMs were introduced in 1997 during the second AI winter (Hochreiter & Schmidhuber, 1997). The landmark paper on LSTMs that precipitated wide-spread applications was written in 2017 (Greff et al., 2017). It was neither nascent nor novel in 2021.

and RNN[26], are novel. Google's vector representation of words (Word2Vec) was only developed in 2013" (Zhao, Ferrari, et al., 2021). Considering that applications of BERT-based models have become commonplace by 2021 in other specialized domains, SE and RE appears to be a laggard in adopting state-of-the-art NLP tools.

While the field is slow to progress, some RE researchers have performed empirical research using LMs - five papers employ LMs on sequence classification problems (Chatterjee et al., 2021a; Deshpande et al., 2021; Hey et al., 2020; Sainani et al., 2020; Varenov & Gabdrahmanov, 2021), one paper used a BERT-as-a-service platform for similarity detection between requirements (Abbas et al., 2022), and one paper using contextualized word embeddings from BERT as an input to detect coreferent entities in requirements (Y. Wang et al., 2022). The results from all seven papers concluded that LMs outperform other methods considered within their research. This validates the potential to expand LM adoption within the field.

Going beyond the technical, the authors also identified other issues in the field, such as a very narrow focus on established NLP problems without consideration for potential alternatives, a lack of annotated datasets and benchmark datasets, studies being limited to the application of tools off-the-shelf, and low reproducibility of research as the tools were not available for open use (Zhao, Ferrari, et al., 2021). These findings show great similarity with those reported in Table 3-9. While these conclusions are easy to agree with, they are difficult to solve. At the same time, making progress in these areas does not necessarily translate into real impact. For example, researchers have noted that while benchmark datasets serve as a yardstick for technological progress in research, it is not a direct proxy for the downstream impact of such technologies (Blagec et al., 2022; Paullada et al., 2021). However, a complete absence of benchmarking practices is also non-ideal; researchers can be motivated to perform empirical research with new technologies such as LMs and claim state-of-the-art performance. In these scenarios, the inability to objectively compare between models prevents the field from separating the true signal from the noise, and identify models that are most worthy of further research.

[26] The inability to learn long-term dependencies by simple RNNs was shown in 1994, leading to limited usage within NLP research (Bengio et al., 1994). LSTMs were the proposed solutions to overcome this weakness exhibited by RNNs. Again, there is scant evidence that RNNs were nascent in 2021.

Considering the above, there is significant room for the SE and RE community to leverage on the democratization of NLP tools and explore the application of such tools in a more rapid manner.

Chapter 4

4 Application of Pre-trained Language Models in Systems Engineering

Language Models represent the frontier of NLP research today. Generic off-the-shelf variants perform well across most applications. Various adaptation techniques also allow these generic variants to be adapted for specific domains for improved model performance and robustness. With this understanding, this chapter investigates the use of these LMs in SE.

This chapter is organized into four main sections: Section 4.1 states the proposed research questions and their intended value. Section 4.2 provides details of the experimental set-up. Section 4.3 describes the experiments in detail. Lastly Section 4.4 provides a summary of the results.

4.1 Research Questions

The scope of experimental work is designed to answer the following research questions:

- **RQ1**: *Which off-the-shelf pre-trained language models are most suitable for application within the systems engineering domain?*
- **RQ2:** *To what extent does task-adaptive pretraining of language models improve classification performance within the systems engineering domain?*

Many off-the-shelf pre-trained LMs have become available in the recent years, with the LMs differing in terms of model size, pretraining text corpus, construction of underlying vocabulary, and pretraining technique. These differences have led to differing model performance across domains and tasks. **RQ1** seeks to assess a subset of these LMs in the systems engineering domain.

A series of domain-adaptation research have also demonstrated that pretraining a LM from scratch with a domain-specific text corpus delivers the improved model performance (Beltagy et al., 2019; Gu et al., 2021). However, the time and resource limitations of this thesis prohibits the implementation of such an approach, leaving the author to focus on continual pretraining of off-the-shelf LMs, specifically TAPT. **RQ2**

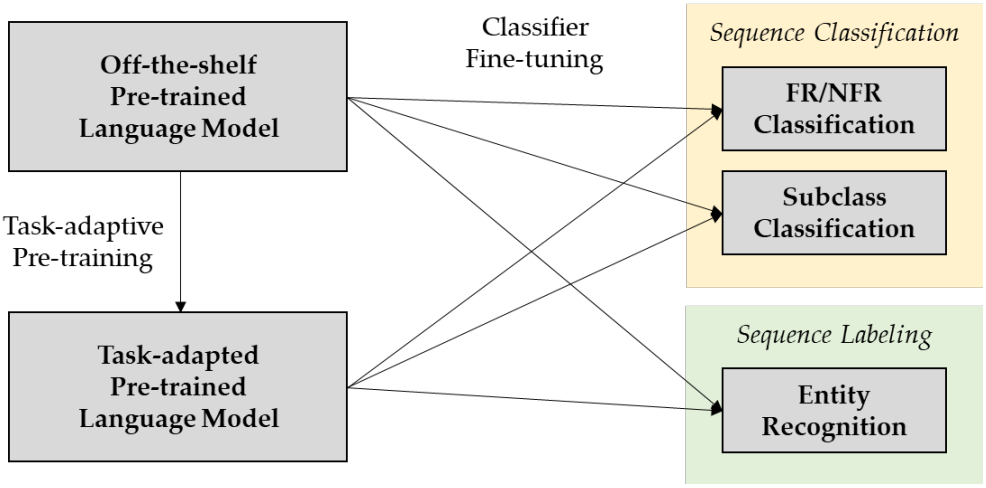
seeks to assess the efficacy of performing TAPT on each of the off-the-shelf LMs in comparison to direct fine-tuning for classification.

4.2 Experimental Set-up

The experimental set-up is illustrated in Figure 4-1. The off-the-shelf pre-trained LM will be adapted for the systems engineering domain using TAPT. Thereafter, the off-the-shelf variant and the task-adapted variant will be fine-tuned on four tasks – two sequence classification tasks and two sequence labeling tasks. This experimental workflow will be replicated for each of the off-the-shelf pre-trained LMs. This experimental set-up will allow the efficacy of TAPT to be assessed across a range of tasks for each of the off-the-shelf pre-trained LMs (**RQ2**). Concomitantly, comparisons across the off-the-shelf pre-trained LMs can also be made (**RQ1**).

The remainder of this section will provide the rationale for the choice of LMs (Section 4.2.1), describe the model architectures for each the performance of TAPT and classifier fine-tuning (Section 4.2.2), and give an overview of the training corpora used in the experiment (Section 4.2.3).

Figure 4-1: Overview of Experimental Set-up.

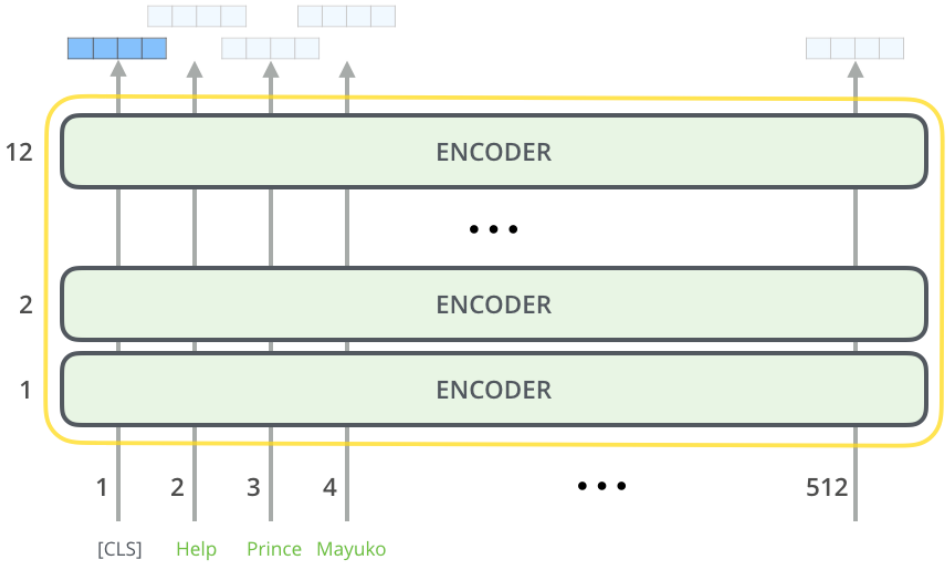


4.2.1 Off-the-shelf Language Models: BERT$_{BASE}$, RoBERTa$_{BASE}$ & SciBERT

Three off-the-shelf LMs are used in this experiment: BERT$_{BASE}$ (Devlin et al., 2018), RoBERTa$_{BASE}$ (Liu et al., 2019) and SciBERT (Beltagy et al., 2019). The choice of these three LMs is intended to allow a fair comparison between sufficiently distinct models.

Model performance is impacted by model architecture, with larger deep learning models often outperforming smaller ones. Hence, the three models share the same underlying model architecture (based on BERT$_{BASE}$) comprising of 12 transformer encoder layers, hidden size of 768, and 12 attention heads (see Figure 4-2)[27]. This allows a fair comparison across the three LMs. It should be noted that LARGE variants of BERT and RoBERTa also exist comprising of 24 transformer encoder layers, hidden size of 1024, 16 attention heads, and accepts up to sequences of up to 512 tokens. However, the use of LARGE variants incurs a significantly higher computational cost[28] and would exclude SciBERT (which is only developed in the BASE variant).

Figure 4-2: Illustration of BERT$_{BASE}$, RoBERTa$_{BASE}$ & SciBERT Model Architecture (Alammar, n.d.).



[27] BERT$_{BASE}$ and SciBERT has 110 M parameters. RoBERTa$_{BASE}$ has 123 M parameters. This is due to a difference in vocab size.

[28] The number of BERT$_{LARGE}$ parameters (345M) is approximately three times of BERT$_{BASE}$ (110M).

Despite architecture similarities, there are several differences between the three LMs attributed to their respective pretraining text corpora (and by extension the corpora text domain), underlying vocabulary, and training methodology. There is an intrinsic relationship between the pretraining corpora and the vocabulary that is traced to the way words are deconstructed into sub-word tokens. For the pre-determined vocabulary size, each of the model constructs a token vocabulary that allows the complete representation of all words. Such tokenization procedures include the Wordpiece model (Wu et al., 2016) used in BERT, and Byte Pair Encoding (BPE) (Sennrich et al., 2016) used in RoBERTA and SciBERT. In general, the composition of the token vocabulary is determined in a greedy fashion which allows the most frequent words to be represented as full tokens. This has two implications. First, the larger the vocabulary size, fewer sub-word tokens are required. Second, when the underlying text corpus changes, the most frequent words change as well, affecting the composition of the token vocabulary.

Table 4-1: Off-the-shelf Language Models.

Model	Pretraining Domain	Pretraining Corpora	Vocab size (# Tokens)
$\text{BERT}_{\text{BASE}}$	Generic	BookCorpus & Wikipedia passages	30,522
$\text{RoBERTa}_{\text{BASE}}$	Generic	BookCorpus, CC-News, OpenWebText & Stories	50,265
SciBERT	Biomedicine & Computer Science	Full text of academic research papers	30,522

The choice of these three LMs allows both factors to be represented. First, the expanded vocabulary size of $\text{RoBERTa}_{\text{BASE}}$ (50k) relative to $\text{BERT}_{\text{BASE}}$ and SciBERT (30k) would allow for a complete representation of a larger proportion of words. Second, the scientific language of the SciBERT pretraining corpus represents a domain shift in comparison to $\text{BERT}_{\text{BASE}}$ and $\text{RoBERTa}_{\text{BASE}}$. This means that the manner that engineering domain corpus is tokenized differs across all three models. For the above reasons, these three LMs are included.

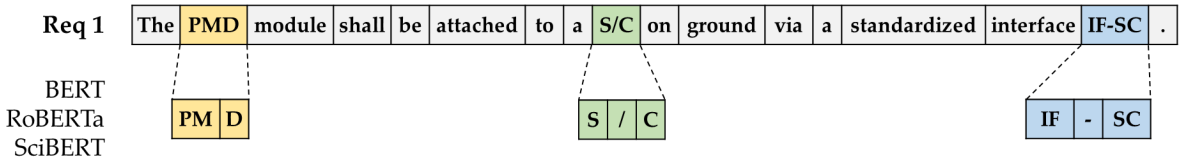
The similarities and differences in tokenization across these models can be illustrated with the two following requirements:

- **Requirement 1**: *The PMD module shall be attached to a S/C on ground via a standardized interface IF-SC.* (16 words)

- **Requirement 2**: *Thermally induced seeing degradation caused by temperature differences shall be minimized by a suitable combination of natural ventilation, insulation, surface emissivity, daytime air conditioning, limiting daytime air leakage, and minimizing thermal inertia of the enclosure interior. The goal is to allow the interior to follow the night-time ambient air temperature as closely as practical.* (54 words)

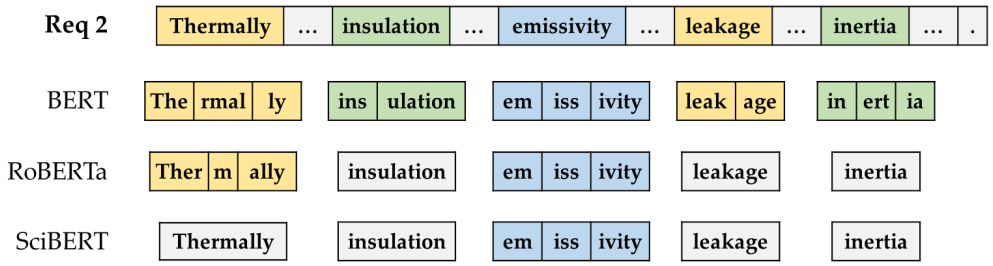
For **Requirement 1**, all three models were tokenized in the same manner (Figure 4-3). The 16-word sentence was tokenized into 21 tokens, with the three abbreviated terms ("PMD", "S/C", and "IF-SC") broken into sub-word tokens. Due to the heavy use of abbreviations in the engineering corpus, their tokenization is commonplace.

Figure 4-3: Tokenization of Requirement 1.



For **Requirement 2**, several differences in the tokenization of five words ("Thermally", "insulation", "emissivity", "leakage", and "inertia") were observed. The remaining 49 words were not broken into sub-word tokens. It can be observed that BERT$_{BASE}$ tokenized all five words into 13 sub-word tokens. RoBERTa$_{BASE}$, which has a larger token vocabulary, retained three intact words. SciBERT, with a scientific token vocabulary, retained four intact words.

Figure 4-4: Tokenization of Requirement 2.



4.2.2 Model Architectures for Masked Language Modelling, Sequence Classification & Sequence Labeling

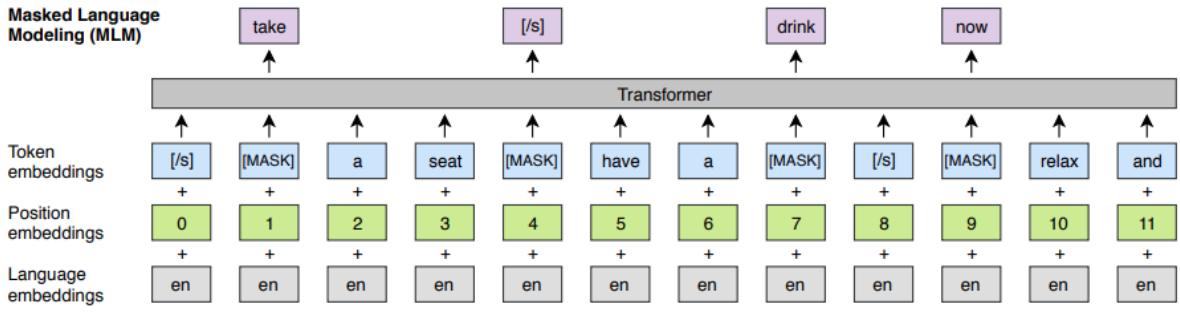
The generic architecture of the three LMs is illustrated in Figure 4-2. The following sections will describe how this generic architecture is applied to perform masked language modelling (MLM), sequence classification and sequence labeling.

4.2.2.1 Masked Language Modelling

MLM is a self-supervised training approach that is used for TAPT of LMs. The objective of MLM is to learn the language representation by predicting tokens that were intentionally masked from the input sequence. As the masked token is known, these tokens serve as the target output. This allows unlabeled text corpora to be used for self-supervised training.

MLM is performed by masking a proportion of tokens within each sequence; this involves the replacement of the token with a [MASK] token (see Figure 4-5); 15% of tokens are masked by convention (Wettig et al., 2022). The LM is then trained in a self-supervised manner with the masked sequences and the true tokens. The performance of the LM is evaluated by the commonly used perplexity (PPL) metric.

Figure 4-5: Model Architecture for Task-adaptive Pretraining using Masked Language Modelling (Conneau & Lample, 2019).



4.2.2.2 Sequence Classification and Sequence Labeling

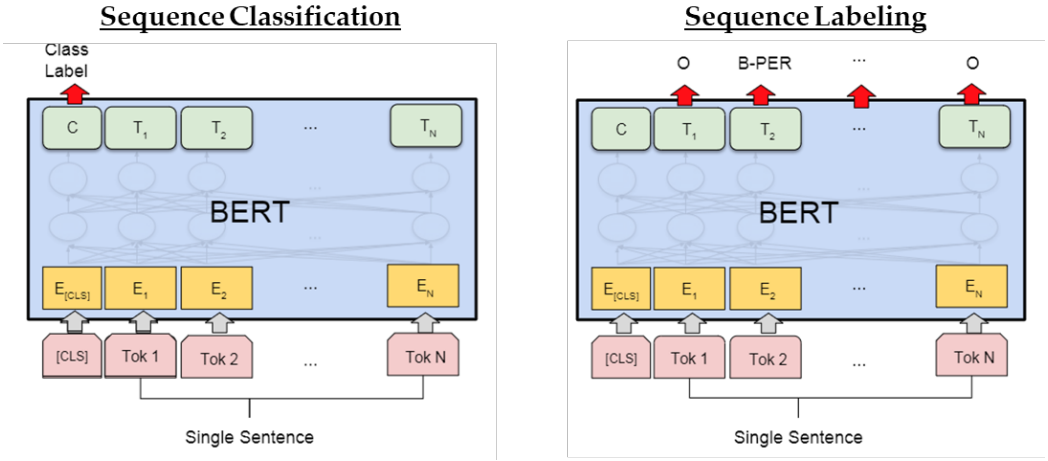
Sequence classification and sequence labeling are supervised training tasks. For both tasks, the input sequence of tokens that are passed into the model returns a sequence of

output vectors of an equivalent length. All sequence classification and labeling tasks are performed using the sequence of output vectors. For these three LMs, each output vector is of dimension (768, 1). In the LM architecture, an additional token, [CLS], is automatically appended at the start of the input sequence. The corresponding output vector is taken as a dense representation of the entire input sequence.

In sequence classification, the objective is to accurately predict the class labels for each input sequence. This is done by passing the first vector of (768, 1), which is a dense representation of the entire sequence, into a dense output layer with a softmax activation function (see Figure 4-6). This dense output layer outputs probabilities (or logits) that correspond to every output class. The predicted class corresponds to the class with the highest probability. The LM is fine-tuned for sequence classification using pairs of input sequences and the corresponding class labels with a categorical cross-entropy loss function. The categorical cross-entropy loss function decreases when the difference between the predicted probability and actual class is minimized.

In sequence labeling, the objective is to accurately predict the labels for every input token. Every vector in the output sequence, except the one used for sequence classification is passed into a dense layer with a softmax activation function (see Figure 4-6). As each of these vectors of dimension (768, 1) is a contextual representation of the corresponding input token, this allows a class to be predicted for every token. The LM is fine-tuned using input token sequences and a sequence of class labels, with both sequences of equal length. The training scheme seeks to minimize the categorical cross-entropy loss.

Figure 4-6: Model Architectures for LM fine-tuning for Sequence Classification (right) and Sequence Labeling (left) (Devlin et al., 2018).



4.2.3 Training Corpora

4.2.3.1 Unlabeled Corpora for Task-adaptive Pretraining

The assembly of this training corpora is limited by three factors. First, while unlabeled text can generally be generated inexpensively through web-scraping or APIs, the document-based nature of systems engineering renders the approach largely infeasible. Second, access to enterprise requirement management tools (such as IBM DOORS) was also not possible, rendering large scale extraction from a structured database was similarly infeasible. Third, the performance of TAPT requires a task-specific text corpus. As such, parsing of requirement documents alone is not sufficient. A secondary step of extracting requirement statements from these documents is necessary.

Table 4-2: Summary of Unlabeled Corpora Sources.

S/N	Data Source	# Docs	# Reqs	# Words
1	PROMISE	15	~600	~12k
2	PURE	20	~3k	~117k
3	ECSS	1	~27k	~765k
4	Web-sourced	42	~10k	~219k
	Total	**88**	**~41k**	**~1.11M**

The training unlabeled training corpora of 1.1M words from 41k requirement statements was assembled from four sources (see Table 4-2). The PROMISE Software Engineering Repository (University of Ottawa, n.d.) is a widely used requirements dataset within the requirements engineering domain. The Public Requirements dataset (PURE) is a set of 79 natural language open-source requirement documents (Ferrari et al., 2017). However, most of the requirements are not phrased in a manner consistent with system engineering standards and therefore excluded. The European Cooperation for Space Standardization (ECSS) requirements dataset was generated by the European Space Agency to "define a coherent and single set of standards for all European space activities" (Berquand & Riccardi, 2021). The final set of 42 requirement documents are scraped by the author from open sources (see Table 4-3), and represents one of the contributions of this thesis. These requirement documents mainly involve spacecrafts and telescopes, and are made available by the ESA, National Aeronautics and Space Administration (NASA), and other organizations. These requirements are also generated during different stages of the

systems engineering process – spanning across science requirements, mission requirements and system requirements. This is intended to allow the task-adapted model to be applied to tasks at various stages of the system engineering process. However, this hypothesis is unverified in this thesis and is left for future work.

Table 4-3: Web-sourced Unlabeled Corpora.

S/N	Dataset	System Type
1	CCI+ Biomass system requirements (ESA, 2019a)	Spacecraft
2	CCI+ High Resolution Land Cover ECV system requirements (ESA, 2020)	Spacecraft
3	CCI+ Permafrost system requirements (ESA, 2021a)	Spacecraft
4	CCI+ Sea Ice system requirements (ESA, 2012c)	Spacecraft
5	CCI+ Sea Salinity system requirements (ESA, 2019b)	Spacecraft
6	CCI+ Sea Surface Temperature system requirements (ESA, 2012a)	Spacecraft
7	CCI+ Water Vapor Temperature system requirements (ESA, 2021b)	Spacecraft
8	Copernicus Sentinels 4 & 5 mission requirements traceability (ESA, 2017)	Spacecraft
9	Cross-scale TRS mission requirements (ESA, 2007)	Spacecraft
10	DUE GlobBiomass system requirements (ESA, 2015)	Spacecraft
11	EarthCARE Project system requirements (ESA, 2008)	Spacecraft
12	EChO mission requirements (ESA, 2013b)	Spacecraft
13	EChO science requirements (ESA, 2013a)	Spacecraft
14	Galileo Galilei mission requirements (ThalesAleniaSpace, 2009)	Spacecraft
15	Gateway system requirements (NASA, 2019)	Spacecraft
16	Giant Magellan Telescope observatory architecture (Walls, Sitarski, et al., 2021)	Telescope
17	Giant Magellan Telescope observatory requirements (Walls, Bouchez, et al., 2021)	Telescope
18	Gravity Recovery & Climate Experiment science and mission requirements (University of Texas, 1998)	Spacecraft
19	Herschel Ground Segment interface requirements (ESA, 2006)	Spacecraft
20	Herschel-Planck operations interface requirements (ESA, 2003)	Spacecraft
21	Herschel-Planck system requirements specification (ESA, 2004)	Spacecraft
22	James Webb Space Telescope mission requirements (NASA, 2007)	Spacecraft
23	Joint Polar Satellite System Ground Segment data product specification (NOAA & NASA, 2019a)	Spacecraft
24	Joint Polar Satellite System Level I requirements (NOAA & NASA, 2019b)	Spacecraft
25	Joint Polar Satellite System Level I requirements supplement (NOAA & NASA, 2019c)	Spacecraft

S/N	Dataset	System Type
26	Joint Polar Satellite System processing requirements (US DOC et al., 2018a)	Spacecraft
27	Joint Polar Satellite System science requirements (US DOC et al., 2018b)	Spacecraft
28	LOFT mission requirements (ESA, 2013c)	Spacecraft
29	MarcoPolo-R mission requirements (ESA, 2012b)	Spacecraft
30	MarcoPolo-R science requirements (ESA, 2011)	Spacecraft
31	Mobile Surveillance System technical requirements (UN, n.d.)	Communication
32	RITA core system requirements specification (RITA, 2011)	Communication
34	SKA Level 0 science requirements (SKA Organisation, 2015b)	Spacecraft
35	SKA Level I system requirements specification (SKA Organisation, 2015a)	Spacecraft
36	SPICA science requirements (ESA, 2009)	Spacecraft
37	STE-QUEST mission requirements (ESA, 2012d)	Spacecraft
38	TeSeR post-mission disposal subsystem requirement (Airbus DS GmbH, 2016)	Spacecraft
39	Thirty Meter Telescope observatory architecture (TMT, 2021a)	Telescope
40	Thirty Meter Telescope operations requirements (TMT, 2021b)	Telescope
41	Thirty Meter Telescope science requirements (TMT, 2021c)	Telescope
42	WISE mission operations system requirements (NASA, 2005)	Spacecraft

4.2.3.2 Labeled Corpora for Supervised Classifier Fine-tuning

Three different labeled datasets are used for the LM fine-tuning (Table 4-4). Annotations for the first two datasets were obtained from other research while the third dataset was produced as part of this thesis. To address the two research questions, it is ideal to evaluate each LM on diverse datasets covering a range of sequence classification and sequence labeling tasks. An extensive experiment will support a more rigorous examination of these LMs. However, due to time limitations, this thesis only uses three datasets.

Table 4-4: Labeled Corpora for Experiments.

S/N	Dataset	Task	Annotation	# Reqs
1	Combined requirements dataset – PROMISE, Leeds, Dronology, ReqView & WASP	FR/NFR classification (Sequence classification)	From source	956
2	PROMISE dataset	Subclass classification (Sequence classification)	From source	625

S/N	Dataset	Task	Annotation	# Reqs
3	Hybrid entity recognition dataset	Entity recognition (Sequence labeling)	Author generated	923

The combined requirements dataset (Table 4-4, S/N 1) is hybrid dataset created by combining five separate datasets used in a requirements classification research paper (Dalpiaz et al., 2019). The five datasets are: (1) PROMISE dataset consisting of student generated requirements, (2) Leeds dataset which details the requirements for Leeds University's library online management system, (3) Dronology which contains requirements for a unmanned aerial system, (4) ReqView which details the requirement specifications for the ReqView system, and (5) WASP which details the requirements for the Web Architecture for Services platform (WASP). Every requirement in this dataset is labeled as a functional requirement (FR) or non-functional requirement (NFR).

The PROMISE dataset (University of Ottawa, n.d.) (Table 4-4, S/N 2) is subset of the one mentioned previously and follows a different annotation scheme. The NFRs are labeled as one of the following four subclasses: O (operations), PE (performance), SE (security) and US (useability). The FRs are labeled as F (functional).

The hybrid entity recognition dataset (Table 4-4, S/N 3) is generated as part of this thesis. It draws on 923 requirements from six different datasets (Table 4-3, S/N 11, 16, 21, 27, 35, 39). Subjects within the requirements are assigned one of the following labels: ACT (action), ATTR (attribute), RELOP (relative operator), QUANT (quantity), ENT (entity), or O for words that do not fit into any of the above labels. This set of labels are developed by Ajisafe, F. and Norheim, J. to facilitate the extraction of structured data from requirements to facilitate SE modelling. In this experiment, multi-word subjects are annotated using the IOB2 scheme. Due to resource limitations, the annotation was performed by a single annotator using Labelbox (Labelbox, 2022). As such, the annotation process lacks the rigor demanded of typical NLP annotation tasks, and any reuse of this dataset must be done with caution.

4.2.4 Comparison with Related Works

The application of LMs within SE and RE is not novel. The experiments performed for this thesis serves as an extension of related work by other researchers (Table 4-5). However, the main differences between this thesis and earlier works are as follows. First,

it is the first study which seeks to assess the robustness of pre-trained LMs across both sequence classification and sequence labeling tasks, in the SE/ RE context. This approach is the de facto standard in NLP research that is not commonplace within the RE community. Second, this is also the first formal application of NER using LMs in the SE/ RE context within the literature.

Table 4-5: Comparison with related applications of Language Models within Systems Engineering and Requirements Engineering.

Related Research	LMs assessed	Pretraining Steps			Fine-tuning Applications	
		From Scratch	DAPT	TAPT	Sequence Classification	Sequence Labeling
Reference	**BERT RoBERTa SciBERT**	**x**	**x**	**√**	**√**	**√**
Hey et al., 2020	BERT	x	x	x	√	x
Chatterjee et al., 2021	BERT	x	x	√	√	x
Varenov & Gabdrahmanov, 2021	BERT DistilBERT XLnet	x	x	√	√	x
Deshpande et al., 2021	BERT	x	x	√	√	x
Sainani et al., 2020	BERT	x	x	√	√	x
Berquand et al., 2021	BERT RoBERTa SciBERT	x	√	x	x	√

4.3 Experiments

All experiments were performed with the following frameworks (Table 4-6). The computing workload is run on a n1-standard-4 Google Cloud Vertex AI instance with a single NVIDIA Tesla P100 GPU.

Table 4-6: Python Frameworks.

Framework	Version
Python	3.7.10
TensorFlow	2.8.2
Transformers	4.19.2
Datasets	2.2.2

Framework	Version
Tokenizers	0.12.1
Seqeval	1.2.2

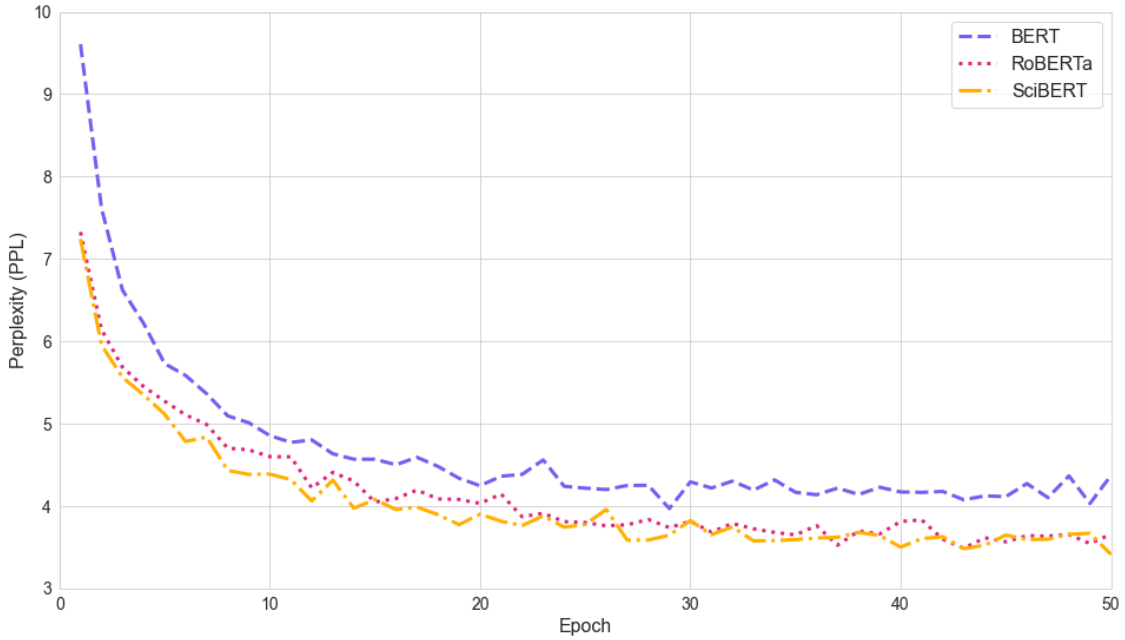
4.3.1 Task-Adaptive Pretraining of Language Models

This dataset is preprocessed into a training (95%, 37,797 requirements) and validation (5%, 1990 requirements) dataset for TAPT using MLM. The exact composition of this dataset can be accessed at https://hf.co/datasets/limsc/mlm-tapt-requirements. TAPT was performed on the three LM ($BERT_{BASE}$, $RoBERTa_{BASE}$ & SciBERT) using the hyperparameters listed in Table 4-7, taking an average of four hours for each LM. The training process is monitored using the PPL metric of the validation dataset (Figure 4-7).

Table 4-7: Hyperparameters for Task-adaptive Pretraining of $BERT_{BASE}$, $RoBERTa_{BASE}$ & SciBERT using Masked Language Modelling.

Hyperparameter	Value
Masking probability	0.15
Chunk size (tokens)	128
Initial learning rate	2E-5
Weight decay	0.01
# warm-up steps	1000
Epochs	50
Random seed	1

Figure 4-7: Perplexity Scores for Task-adaptive Pretraining of BERT$_{BASE}$, RoBERTa$_{BASE}$ & SciBERT.



It can be observed from Figure 4-7 that the PPL of the validation dataset decays rapidly within the first 10 epochs, and the rate of decrease becomes more gradual in later epochs. While a low PPL score can lead to improved model performance, it can also lead to overfitting. As such, early stopping is performed to achieve a balance between the two. BERT$_{BASE}$, RoBERTa$_{BASE}$ and SciBERT were trained for 29, 43 and 20 epochs respectively. The resulting post-TAPT LMs are hereon referred to as ReqBERT (https://hf.co/limsc/reqbert-tapt-epoch29), ReqRoBERTa (https://hf.co/limsc/reqroberta-tapt-epoch43), and ReqSciBERT (https://hf.co/limsc/reqscibert-tapt-epoch20) respectively.

4.3.2 Task 1: Classification of Functional and Non-functional Requirements

The six LMs were evaluated using the combined requirements dataset (Table 4-4, S/N 1). This dataset is preprocessed into a training (70%, 669 requirements), validation (15%, 143 requirements) and testing (15%, 144 requirements) datasets. The exact composition of this dataset can be accessed at https://hf.co/datasets/limsc/fr-nfr-classification.

The fine-tuning methodology used for Task 1 is common to all four tasks. A set of hyperparameters (Table 4-8) are used to determine the best performing variant of each of the six models. Hyperparameters such as weight decay, number of warm-up steps and

number of epochs are kept constant, while a full grid search was performed for the initial learning rate and batch size. For a set of hyperparameter for each model, the fine-tuning was performed 15 times with using a set of random seeds. The choice of batch size and initial learning was informed by the original BERT paper (Devlin et al., 2018). While the total number of epochs is kept constant, model loss is evaluated at every evaluation allowing for early stopping. The conduct of 15 runs is informed by a separate study which noted the impact of random seeds on model fine-tuning performance (Dodge et al., 2020).

Table 4-8: Hyperparameters for model selection for Task 1.

Hyperparameter	Value
Initial learning rate	{2E-5, 3E-5, 5E-5}
Weight decay	0.01
# warm-up steps	0
Epochs	5
Batch size	{16, 32}
Frozen BERT encoder layers	Nil

This model selection process is explained for $\text{BERT}_{\text{BASE}}$ using Figure 4-8. $\text{BERT}_{\text{BASE}}$ is fine-tuned with the six combinations of hyperparameters listed in Table 4-8. For each combination of hyperparameters, the fine-tuning process performed 15 times, each with a randomly initialized set of model weights for the classification head. The model loss for the training and validation datasets are tracked at the end of every epoch to give the results in Figure 4-8. It can be observed that beyond 2 epochs, the training loss continually decreases, while the validation loss remains stable or increases. This indicates that overfitting has occurred. As such, models fine-tuned beyond 2 epochs are excluded from consideration. After which, the preferred hyperparameters are chosen based on the lowest average validation loss. In this instance, $\text{BERT}_{\text{BASE}}$ returns the best results with a batch size of 16 and initial learning rate of 2E-5 after 2 epochs. This process is performed for each of the six models to determine the preferred fine-tuning parameters that are reflected in Table 4-9.

Figure 4-8: Average loss (n = 15) for various BERT$_{BASE}$ fine-tuning hyperparameters for Task 1.

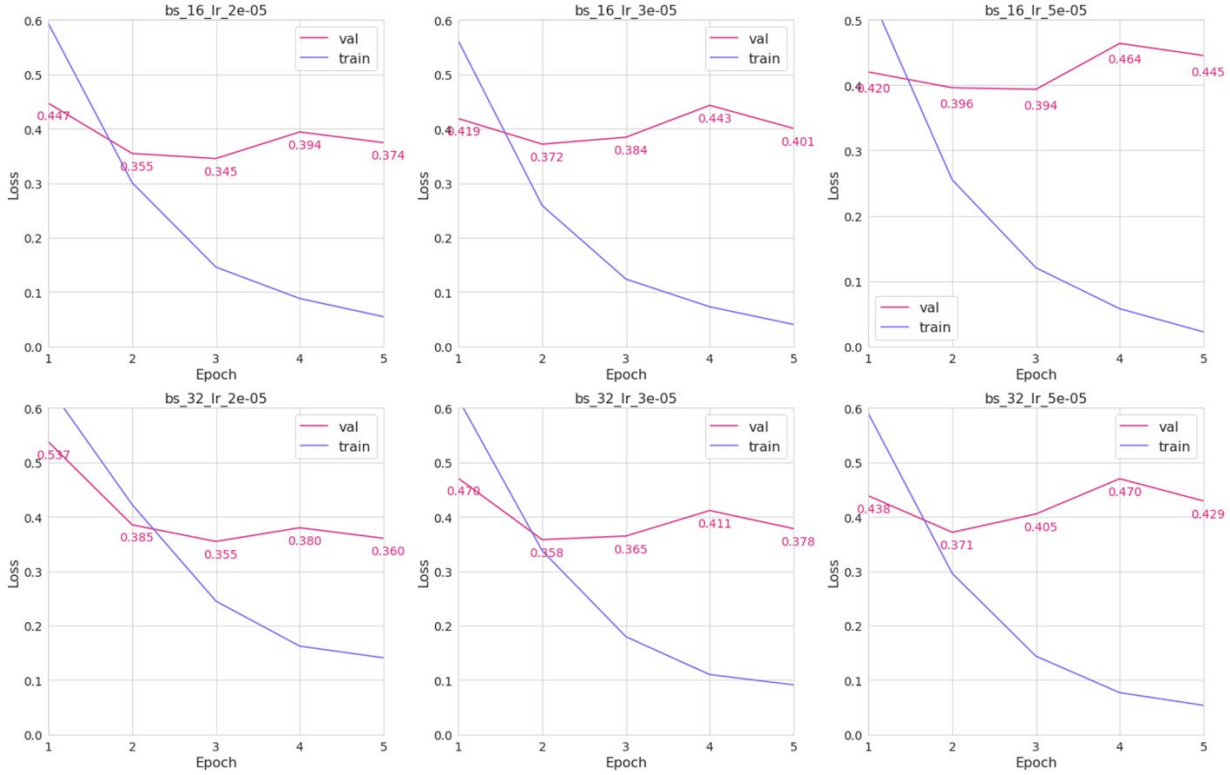


Table 4-9: Selected fine-tuning hyperparameters for Task 1.

LM	Batch size	Initial learning rate	Epochs
BERT	16	2E-5	2
RoBERTa	32	3E-5	2
SciBERT	32	3E-5	2
ReqBERT	16	2E-5	2
ReqRoBERTa	16	2E-5	2
ReqSciBERT	32	5E-5	2

The six LMs were fine-tuned using the preferred fine-tuning hyperparameters and evaluated on the holdout testing dataset. The average F1-scores are reported in Table 4-10. To compare the performance difference between the original and fine-tuned variants of

each LM, the Welch's t-test[29] is performed and reported as well. From the analysis, it is observed that SciBERT's outperforms all other models in terms of F1-score both at the aggregate and categorical level (FR or NFR). Comparing $BERT_{BASE}$ to ReqBERT and $RoBERTa_{BASE}$ to ReqROBERTA, the results suggest that TAPT is beneficial to model performance. However, the improvement in F1-score varies, ranging from 0.3% to 1.8%. However, comparing SciBERT to ReqSciBERT, this trend is reversed with SciBERT producing superior performance both at the aggregate and categorical level. However, nearly all differences were not found to be statistically significant.

Table 4-10: Average testing dataset F1-scores for $BERT_{BASE}$, $RoBERTa_{BASE}$, SciBERT, ReqBERT, ReqRoBERTa & ReqSciBERT for Task 1. Standard deviation in F1-scores are included in subscript. Impact of TAPT is included in parentheses and annotated with * if statistically significant.

F1-score	BERT	RoBERTa	SciBERT	ReqBERT	ReqRoBERTa	ReqSciBERT
FR	$86.9_{1.3}$	$86.9_{3.0}$	$87.7_{1.2}$	$87.4_{0.8}$ (+0.5)	$87.6_{0.8}$ (+0.7)	$87.1_{0.7}$ (-0.6)
NFR	$79.4_{4.0}$	$76.8_{13.3}$	$81.2_{2.2}$	$79.7_{1.8}$ (+0.3)	$80.0_{1.3}$ (+3.2)	$78.8_{1.2}$ (-2.4)*
Weighted Avg	**$83.8_{2.4}$**	**$82.6_{7.3}$**	**$85.0_{1.6}$**	**$84.1_{1.2}$ (+0.3)**	**$84.4_{1.0}$ (+1.8)**	**$83.6_{0.9}$ (-1.4)***

4.3.3 Task 2: Classification of Requirement Subclasses

The six LMs were evaluated using the PROMISE dataset (Table 4-4, S/N 2). This dataset is preprocessed into a training (70%, 352 requirements), validation (15%, 76 requirements) and testing (15%, 76 requirements) datasets. The exact composition of this dataset can be accessed at https://hf.co/datasets/limsc/subclass-classification.

The fine-tuning methodology and choice of hyperparameters (Table 4-11) is identical to Task 1. Following the fine-tuning methodology, the hyperparameters for each of the LMs were selected (Table 4-12).

[29] Welch's t-test is an adaptation of the Student's t-test that allows unequal sample sizes and sample variances. These assumptions of Student's t-test are unlikely to hold in this analysis.

Table 4-11: Hyperparameters for model selection for Task 2.

Hyperparameter	Value
Initial learning rate	{2E-5, 3E-5, 5E-5}
Weight decay	0.01
# warm-up steps	0
Epochs	5
Batch size	{16, 32}
Frozen encoder layers	Nil

Table 4-12: Selected fine-tuning hyperparameters for Task 2.

LM	Batch size	Initial learning rate	Epochs
BERT	16	5E-5	3
RoBERTa	16	5E-5	4
SciBERT	16	5E-5	3
ReqBERT	16	5E-5	3
ReqRoBERTa	16	5E-5	3
ReqSciBERT	16	5E-5	3

The six LMs were fine-tuned using the preferred fine-tuning hyperparameters and evaluated on the holdout testing dataset. The aggregated F1-score statistics and Welch t-test results are reported in Table 4-13.

Table 4-13: Average F1-scores for $BERT_{BASE}$, $RoBERTa_{BASE}$, SciBERT, ReqBERT, ReqRoBERTa & ReqSciBERT for Task 2. Standard deviation in F1-scores are included in subscript. Impact of TAPT is included in parentheses and annotated with * if statistically significant.

F1-score	BERT	RoBERTa	SciBERT	ReqBERT	ReqRoBERTa	ReqSciBERT
F	$85.6_{5.9}$	$91.7_{6.8}$	$92.3_{2.9}$	$91.0_{1.6}$ (+5.4)*	$95.1_{1.3}$ (+3.4)	$95.6_{1.1}$ (+3.3)*
O	$48.3_{29.0}$	$71.1_{29.5}$	$73.8_{12.4}$	$79.1_{7.3}$ (+30.8)*	$84.4_{7.2}$ (+13.3)	$88.1_{4.2}$ (+14.3)*
PE	$47.9_{39.1}$	$75.7_{26.7}$	$95.7_{5.7}$	$87.7_{7.3}$ (+39.8)*	$84.1_{8.3}$ (+8.4)	$92.8_{4.8}$ (-2.9)
SE	$62.7_{27.2}$	$80.0_{23.3}$	$81.3_{1.0}$	$82.7_{8.1}$ (+20.0)*	$87.7_{8.9}$ (+7.7)	$89.5_{6.2}$ (+8.2)*

F1-score	BERT	RoBERTa	SciBERT	ReqBERT	ReqRoBERTa	ReqSciBERT
US	54.0$_{25.8}$	79.1$_{23.8}$	81.1$_{10.5}$	77.2$_{10.3}$ (+23.2)*	85.7$_{5.9}$ (+6.6)	89.7$_{5.1}$ (+8.6)*
Weighted Avg	**70.0$_{13.8}$**	**83.9$_{15.1}$**	**86.6$_{5.0}$**	**86.0$_{3.8}$ (+16.0)***	**90.5$_{3.1}$ (+6.6)**	**92.7$_{1.8}$ (+6.1)***

It can be observed that all pre-trained LMs outperform their original variants by a significant margin, this suggests that TAPT is beneficial for model performance. The sole exception is the PE category, where SciBERT performs best. However, beyond these summary statistics, the results from this task also allow conclusions to be drawn on the robustness of each of these LMs.

As F1-score is computed as the harmonic mean of precision and recall, a low value in either can lead to lower F1-score. Precision and recall scores are in turn dependent on their shared numerator (# True Positive); the inability to accurately classify true labels has the most significant impact on the eventual F1-score. Taking BERT for example, the F1-score across 15 runs for the O category has an average of 48.3 with a standard deviation of 29.0. The low average and high standard deviation were due to three runs having a F1-score of 0. This also means that in three of the runs, BERT model failed to correctly predict any of the samples belonging to the O category. This observation was observed to impact both BERT and RoBERTa, while SciBERT and all three pre-trained variants were unaffected. The prevalence of this observation is summarized in Table 4-14. The occurrence of this observation for a specific LM would suggest a lack of robustness.

Table 4-14: Number of runs for which a F1-score of 0 was observed.

Category	BERT	RoBERTa	SciBERT	ReqBERT	ReqRoBERTa	ReqSciBERT
F	0	0	0	0	0	0
O	3	2	0	0	0	0
PE	5	1	0	0	0	0
SE	2	1	0	0	0	0
US	2	1	0	0	0	0

There are two possible inferences from Table 4-14. First, that differences between BERT$_{BASE}$, RoBERTa$_{BASE}$ and SciBERT could be attributed to their underlying differences in training corpus and vocabulary size. Therefore, LMs constructed with larger vocabularies (in the case of RoBERTa$_{BASE}$) or with a more relevant training corpus (in the

case of SciBERT) could perform more robustly when adapted for a different domain. Second, the pretraining of $\text{BERT}_{\text{BASE}}$ and $\text{RoBERTa}_{\text{BASE}}$ allowed its lack of robustness to be resolved. As such, TAPT is beneficial for domain adaptation.

4.3.4 Task 3: Entity Extraction from Requirements

The six LMs were evaluated using the hybrid entity recognition dataset dataset (Table 4-4, S/N 3). This dataset is preprocessed into a training (70%, 646 requirements), validation (15%, 138 requirements) and testing (15%, 139 requirements) datasets. The exact composition of this dataset can be accessed at https://hf.co/datasets/limsc/requirements-entity-recognition.

Table 4-15: Hyperparameters for model selection for Task 3.

Hyperparameter	Value
Initial learning rate	{2E-5, 3E-5, 5E-5}
Weight decay	0.01
# warm-up steps	0
Epochs	5
Batch size	{8, 16}
Frozen encoder layers	Nil

Table 4-16: Selected fine-tuning hyperparameters for Task 3.

LM	Batch size	Initial learning rate	Epochs
BERT	8	3E-5	2
RoBERTa	8	5E-5	2
SciBERT	8	3E-5	2
ReqBERT	8	3E-5	2
ReqRoBERTa	8	5E-5	2
ReqSciBERT	8	3E-5	2

The six LMs were fine-tuned using the preferred fine-tuning hyperparameters and evaluated on the holdout testing dataset. The aggregated F1-score statistics and Welch t-test results are reported in Table 4-17. In aggregate, it can be concluded that the pre-trained variants perform better than their original variants, with statistically significant

improvements observed for ReqRoBERTa and ReqSciBERT. At the categorical level, RoBERTa$_{BASE}$, ReqRoBERTa and ReqSciBERT performs best across different categories.

Table 4-17: Average F1-scores for BERT$_{BASE}$, RoBERTa$_{BASE}$, SciBERT, ReqBERT, ReqRoBERTa & ReqSciBERT for Task 3. Standard deviation in F1-scores are included in subscript. Impact of TAPT is included in parentheses and annotated with * if statistically significant.

F1-score	BERT	RoBERTa	SciBERT	ReqBERT	ReqRoBERTa	ReqSciBERT
ACT	$89.7_{2.5}$	$92.6_{1.9}$	$91.0_{1.0}$	$87.9_{4.4}$ (-1.8)	$93.2_{2.2}$ (+0.6)	$92.8_{1.3}$ (+1.8)*
ATTR	$85.9_{1.7}$	$90.3_{1.3}$	$88.4_{1.3}$	$84.6_{1.5}$ (-1.3)*	$89.3_{1.6}$ (-1.0)	$88.2_{1.7}$ (-0.2)
RELOP	$88.4_{0.8}$	$92.0_{1.1}$	$89.5_{1.2}$	$89.8_{0.9}$ (+1.4)*	$91.9_{0.1}$ (-0.1)*	$90.0_{0.7}$ (+0.5)
QUANT	$86.4_{0.8}$	$89.0_{0.9}$	$89.2_{1.4}$	$88.3_{0.8}$ (+1.9)*	$90.8_{1.3}$ (+1.8)*	$90.9_{0.5}$ (+1.7)*
ENT	$84.5_{1.1}$	$85.4_{1.3}$	$84.1_{1.2}$	$84.0_{1.6}$ (-0.5)*	$86.7_{1.4}$ (+1.3)	$84.9_{1.4}$ (+0.8)
Weighted Avg	**$87.0_{0.7}$**	**$90.0_{0.5}$**	**$88.5_{0.1}$**	**$87.2_{0.9}$ (+0.2)**	**$90.5_{0.5}$ (+0.5)***	**$89.5_{0.7}$ (+1.0)***

4.4 Summary of Results

With the results from Tasks 1, 2 and 3, it is timely to review the original research questions:

- **RQ1**: *Which off-the-shelf pre-trained language models are most suitable for application within the systems engineering domain?*
- **RQ2:** *To what extend does task-adaptive pretraining of language models improve classification performance within the systems engineering domain?*

For **RQ1**, the results indicate that RoBERTa$_{BASE}$ and SciBERT consistently outperform BERT$_{BASE}$ at the aggregate and categorical level. Hence, for applications within the SE domain, the larger underlying vocabulary of RoBERTa$_{BASE}$ and the scientific training corpus of SciBERT are beneficial for model performance.

For **RQ2**, the conclusions are more ambiguous. In general, TAPT is beneficial for model performance; except for SciBERT in Task 1, all LMs outperform their original variants after TAPT. However, the extent of improvement varies on the downstream application. For example, TAPT improves model performance of BERT$_{BASE}$ by 0.3% in Task 1, 16.0%

in Task 2, and 0.2% in Task 3. The improvement in model performance is not always observed. For instance, in Task 1, the performance of SciBERT decreases by 1.4% after TAPT. In Tasks 2 and 3, TAPT can also lead to lower model performance in some categories.

Chapter 5

5 Discussion & Future Work

5.1 Discussion of Results

The future of SE demands greater reuse of knowledge and tools, and the highly specialized symbolic methods developed in yesteryears will become increasingly obsolete. The increased volume of natural language requirements of modern engineered systems also demands methods to be more scalable. The use of pre-trained LMs in Chapter 4 sought to validate its applicability towards a diverse range of sequence classification and sequence labeling tasks in a scalable manner. The remainder of this section will provide a discussion of the empirical results.

From the empirical work, it was found that off-the-shelf pre-trained LMs produced competitive model performance with minimal computational cost. These LMs also do so consistently across different tasks without intervention. This illustrates the general robustness of LMs that were trained on extremely large text corpus of a generic domain. With TAPT, these pre-trained LMs produced improved model performance and exhibited greater robustness (**RQ2**). The TAPT process was inexpensive to perform as the unlabeled text corpus (~41k requirements, 1.1M words) was easy to assemble and the compute times (~4 hours per LM) was reasonably short. Taken together, the results indicate that pre-trained LMs can be quickly adapted for systems engineering applications. In aggregate, it was also observed that, with or without TAPT, RoBERTa and SciBERT consistently outperforms BERT (**RQ1**). This indicates that a larger vocabulary (in the case of RoBERTa) or a more relevant vocabulary (in the case of SciBERT) can be beneficial to model performance. Taken together, this means that domain adaptation of RoBERTa and SciBERT through TAPT should provide SE and RE researchers with a well-performing model with reasonable effort.

However, it remains difficult to assess if the utility of LMs due to three reasons. First, the absence of benchmark datasets and openly available implementation of existing methods renders it impossible to compare the performance of these LMs against earlier reported approaches (see Table 3-9 for a detailed discussion). Second, this experiment adopts weighted average F1-score as the sole evaluation metric. This decision implies that

precision and recall are given equal weight. In the information extraction context, this assumption may not hold true. For instance, the failure to extract all required information may result in the construction of an incomplete system model, affecting its functionality and desired ilities. In this specific context, recall should be prioritized over precision. However, such decisions are ultimately context specific. Lastly, this thesis lacks a practitioner's view of the minimum level of model performance required. When the model has a recall of 90%, this implies that 10% of all required information is not correctly extracted. The failure to extract this 10% of information is associated with various forms of risks such as safety or compliance risk. As such, without a defined performance threshold, it remains difficult to justify if these models perform sufficiently well for use in industry.

5.2 Future Work

Notwithstanding the limitations above, there remains much room for additional research. The scope of future work can be focused on improving model performance or expanding possible applications. These two lines of effort will allow LMs to be employed more effectively across a broader range of tasks.

The methods to achieve improved model performance can be informed by the literature. These methods are described in increasing complexity or computational cost.

First, the large variants of LMs (such as $\text{BERT}_{\text{LARGE}}$ and $\text{RoBERTa}_{\text{LARGE}}$) can be used. The large variants of LMs contain more model parameters and have conventionally performed better than the variants used in this thesis. However, the computational cost of pretraining and fine-tuning such models is substantially higher. For future thesis work, such an approach may not be worthwhile for several reasons. First, while large variants are likely to return superior performance, the absence of a define performance threshold means that the results cannot be interpreted in a meaningful manner. Second, even with the democratization of cloud computing resources, the significant increase in computational cost is likely to produce performance gains that are disproportionately lower.

Second, DAPT (Section 3.3.3.2) can be employed to adapt pre-trained LMs to a specific domain. Coupling DAPT with TAPT has been shown to return improved model performance (Gururangan et al., 2020). This thesis excluded the use of DAPT as it requires the assembly of an extremely large domain-specific text corpus. Furthermore, the

computational cost of performing DAPT is also significantly greater than TAPT. However, this could be feasibly done as a separate study. This would allow the cost benefit analysis of DAPT to be determined.

Third, a vocabulary extension approach can be considered to expand the generic domain token vocabulary with an extension module containing domain-specific tokens (Tai et al., 2020). As RoBERTa and SciBERT consistently outperforms BERT in the experiments, it can be postulated that the larger vocabulary of RoBERTa and context-specific vocabulary of SciBERT is beneficial for model performance. The expansion of the existing vocabulary with domain-specific tokens allows both elements to be incorporated. In the cited study, this approach was shown to return performance comparable to domain specific LMs with 10% of the computational cost. As such, this approach should provide a computationally efficient way to achieve good model performance.

In this thesis, only two sequence classification and one sequence labeling tasks were investigated. However, this represents a sliver of possible applications. For example, sequence generation applications were omitted from this thesis as bi-directional Transformers models are less suitable for autoregressive Transformers models (such as GPT-2). Possible use cases of sequence generation include the generation of V&V test cases from requirements, or rewriting of ambiguous and low-quality requirement statements.

These two lines of effort seed greater use of LMs within the SE domain.

Bibliography

Abbas, M., Ferrari, A., Shatnawi, A., Enoiu, E., Saadatmand, M., & Sundmark, D. (2022). On the relationship between similar requirements and similar software: A case study in the railway domain. *Requirements Engineering*, *1*, 1–25. https://doi.org/10.1007/S00766-021-00370-4/TABLES/4

Airbus DS GmbH. (2016). *TeSeR (Technology for Self-Removal of Spacecraft) Post-Mission Disposal (PMD) system and subsystem requirement document*. https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5ad44134f&appId=PPGMS

Alammar, J. (n.d.). *The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)*. Retrieved June 4, 2022, from https://jalammar.github.io/illustrated-bert/

Arellano, A., Carney, E., & Austin, M. A. (2015). Natural Language Processing of Textual Requirements. *ICONS 2015: The Tenth International Conference on Systems*.

Aronson, A. R. (2001). Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap program. *Proceedings. AMIA Symposium*, 17–21. http://www.ncbi.nlm.nih.gov/pubmed/11825149

Aroyo, L., & Welty, C. (2015). Truth Is a Lie: Crowd Truth and the Seven Myths of Human Annotation. *AI Magazine*, *36*(1), 15–24. https://doi.org/10.1609/AIMAG.V36I1.2564

Beltagy, I., Lo, K., & Cohan, A. (2019). SciBERT: A Pretrained Language Model for Scientific Text. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, 3615–3620. https://doi.org/10.18653/V1/D19-1371

Bender, E. M. (2013). Linguistic Fundamentals for Natural Language Processing: 100 Essentials from Morphology and Syntax. In *Synthesis Lectures on Human Language Technologies* (Issue 3). https://doi.org/10.2200/S00493ED1V01Y201303HLT020

Bengio, Y., Goodfellow, I., & Courville, A. (2015). *Deep Learning*.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166. https://doi.org/10.1109/72.279181

Berquand, A., Darm, P., & Riccardi, A. (2021). SpaceTransformers: Language Modeling for Space Systems. *IEEE Access*, *9*, 133111–133122. https://doi.org/10.1109/ACCESS.2021.3115659

Berquand, A., & Riccardi, A. (2021). *Data for "SpaceTransformers: language modeling for space systems."* https://doi.org/10.15129/3c19e737-9054-4892-8ee5-4c4c7f406410

Binkhonain, M., & Zhao, L. (2019). A review of machine learning algorithms for identification and classification of non-functional requirements. *Expert Systems with Applications: X*, *1*, 100001. https://doi.org/10.1016/J.ESWAX.2019.100001

Blagec, K., Kraiger, J., Frühwirt, W., Samwald, M., & Samwald, A. M. (2022). *Benchmark datasets driving artificial intelligence development fail to capture the needs of medical professionals*. https://doi.org/10.48550/arxiv.2201.07040

Bogenberger, B. (2007). *James Webb Space Telescope Project Mission Requirements Document*. https://ngin.jwst.nasa.gov/

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., … Liang, P. (2021). *On the Opportunities and Risks of Foundation Models*. http://arxiv.org/abs/2108.07258

Braun, D., Klymenko, O., Schopf, T., Akan, Y. K., Matthes, F., & Matthes, F.-R. (2021). The Language of Engineering Training a Domain-Specific Word Embedding Model for Engineering. *2021 3rd International Conference on Management Science and Industrial Engineering*. https://doi.org/10.1145/3460824

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (2017). Classification And Regression Trees. In *Classification and Regression Trees*. Routledge. https://doi.org/10.1201/9781315139470

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D.

(2020). Language Models are Few-Shot Learners. *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*. https://commoncrawl.org/the-data/

Cameron, B., & Adsit, D. M. (2020). Model-Based Systems Engineering Uptake in Engineering Practice. *IEEE Transactions on Engineering Management*, *67*(1), 152–162. https://doi.org/10.1109/TEM.2018.2863041

Chalkidis, I., Fergadiotis, M., Malakasiotis, P., Aletras, N., & Androutsopoulos, I. (2020). LEGAL-BERT: The Muppets straight out of Law School. *Findings of the Association for Computational Linguistics Findings of ACL: EMNLP 2020*, 2898–2904. https://doi.org/10.18653/V1/2020.FINDINGS-EMNLP.261

Chalkidis, I., & Kampas, D. (2019). Deep learning in law: early adaptation and legal word embeddings trained on large corpora. *Artificial Intelligence and Law*, *27*(2), 171–198. https://doi.org/10.1007/s10506-018-9238-9

Chatterjee, R., Ahmed, A., Rose Anish, P., Suman, B., Lawhatre, P., & Ghaisas, S. (2021a). A Pipeline for Automating Labeling to Prediction in Classification of NFRs. *Proceedings of the IEEE International Conference on Requirements Engineering*, 323–333. https://doi.org/10.1109/RE51729.2021.00036

Chatterjee, R., Ahmed, A., Rose Anish, P., Suman, B., Lawhatre, P., & Ghaisas, S. (2021b). A Pipeline for Automating Labeling to Prediction in Classification of NFRs. *Proceedings of the IEEE International Conference on Requirements Engineering*, 323–333. https://doi.org/10.1109/RE51729.2021.00036

Cloutier, R. (2015). Current Modeling Trends in Systems Engineering. *INSIGHT*, *18*(2), 10–13. https://doi.org/10.1002/INST.12013

Cloutier, R. (2019). 2018 Model Based Systems Engineering Survey. In *MBSE Workshop at the INCOSE International Workshop (IW)*.

Conneau, A., & Lample, G. (2019). Cross-lingual Language Model Pretraining. *Advances in Neural Information Processing Systems (NeurIPS)*. https://doi.org/10.5555/3454287

Dalpiaz, F., Dell'Anna, D., Aydemir, F. B., & Cevikol, S. (2019). Requirements Classification with Interpretable Machine Learning and Dependency Parsing. *2019 IEEE 27th International Requirements Engineering Conference (RE)*, *2019-September*, 142–152. https://doi.org/10.1109/RE.2019.00025

DBpedia. (n.d.). Retrieved June 15, 2022, from http://wikidata.dbpedia.org/

Deshpande, G., Sheikhi, B., Chakka, S., Zotegouon, D. L., Masahati, M. N., & Ruhe, G. (2021). Is BERT the New Silver Bullet? - An Empirical Investigation of Requirements Dependency Classification. *Proceedings of the IEEE International Conference on Requirements Engineering*, *2021-September*, 136–145. https://doi.org/10.1109/REW53955.2021.00025

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. http://arxiv.org/abs/1810.04805

Dhanani, J., Mehta, R., & Rana, D. (2022). Effective and scalable legal judgment recommendation using pre-learned word embedding. *Complex & Intelligent Systems 2022*, 1–15. https://doi.org/10.1007/S40747-022-00673-1

Dodge, J., Ilharco, G., Schwartz, R., Farhadi, A., Hajishirzi, H., & Smith, N. (2020). *Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping*. https://doi.org/10.48550/arxiv.2002.06305

Du, N., Huang, Y., Dai, A. M., Tong, S., Lepikhin, D., Xu, Y., Krikun, M., Zhou, Y., Wei, A., Firat, Y. O., Zoph, B., Fedus, L., Bosma, M., Zhou, Z., Wang, T., Wang, Y. E., Webster, K., Pellat, M., Robinson, K., … Cui, C. (2021). *GLaM: Efficient Scaling of Language Models with Mixture-of-Experts*.

Efstathiou, V., Chatzilenas, C., & Spinellis, D. (2018). Word embeddings for the software engineering domain. *Proceedings of the 15th International Conference on Mining Software Repositories*, 38–41. https://doi.org/10.1145/3196398.3196448

ESA. (2003). *Herschel/Planck Operations Interface Requirements Document*. https://www.cosmos.esa.int/documents/12133/1028864/Operations+Interface+Requirements+Document

ESA. (2004). *HERSCHEL / PLANCK System Requirements Specification [SRS]*. https://www.cosmos.esa.int/documents/12133/1028864/Herschel+-+Planck+System+Requirements+Specification

ESA. (2006). *Herschel Ground Segment Interface Requirements Document*. https://www.cosmos.esa.int/documents/12133/1028864/Herschel+Ground+Segment+Interface+Requirements+Document

ESA. (2007). *Cross-Scale TRS Mission Requirements Document*. https://sci.esa.int/documents/34923/36148/1567256170461-MRD_SCIA_2005_073_CS_3_2_ext.pdf

ESA. (2008). *EarthCARE Project System Requirements Document for Phases B, C/D, E1*. https://earth.esa.int/eogateway/documents/20142/37627/EarthCARE-Project-SRD-Phases-B-C-D-E1.pdf?text=requirements

ESA. (2009). *SPICA - Space Infrared Telescope for Cosmology and Astrophysics, Cryogenic Telescope Assembly, Science Requirements Document*. https://sci.esa.int/documents/34314/36242/1567257686494-SPICA_Cryogenic-Telescope-Assembly_Science-Requirements-Document.pdf

ESA. (2011). *MarcoPolo-R Science Requirements Document*. https://sci.esa.int/documents/34944/36626/1567258938910-ECHO_SciRD_v3-2.pdf

ESA. (2012a). *CCI-SST System Requirements Document*. https://climate.esa.int/media/documents/SST_cci_SRD_Issue_1.2_2012_05_07.pdf

ESA. (2012b). *MarcoPolo-R Mission Requirements Document*. https://sci.esa.int/documents/33920/36043/1567259284851-MarcoPolo-R_Mission_Requirements_Document_v3-2.pdf

ESA. (2012c). *Sea Ice Climate Change Initiative: Phase 1, D5.1 System Requirement Document (SRD)*. http://esa-cci.nersc.no/?q=webfm_send/215

ESA. (2012d). *STE-QUEST Mission Requirements Document*. https://sci.esa.int/documents/33940/36014/1567254687407-STE-QUEST_MRD-SRE-PA-2011-074_I2R3_AO.pdf

ESA. (2013a). *EChO - Science Requirements Document*. https://sci.esa.int/documents/34944/36626/1567258938910-ECHO_SciRD_v3-2.pdf

ESA. (2013b). *EChO Mission Requirements Document*. https://sci.esa.int/documents/34944/36626/1567259237366-ECHO_MRD_i3-2.pdf

ESA. (2013c). *LOFT Mission Requirements Document*. https://sci.esa.int/documents/33900/35959/1567259289699-LOFT_MRD_v3-6.pdf

ESA. (2015). *DUE GlobBiomass D8 System Requirements Document*. https://globbiomass.org/wp-content/uploads/DOC/Deliverables/D8/GlobBiomass_D8_SRD_V01.pdf

ESA. (2017). *Copernicus Sentinels 4 and 5 Mission Requirements Traceability Document*. https://sentinel.esa.int/documents/247904/2506504/Copernicus-Sentinels-4-and-5-Mission-Requirements-Traceability-Document.pdf/b15b6786-88cd-4f1d-a67e-a1da70ed595b?t=1531155774000

ESA. (2019a). *CCI BIOMASS System Requirements Document Year 1 Version 1.0*. https://climate.esa.int/sites/default/files/biomass_D3.1_System_Requirements_Document__SRD__V1.0.pdf

ESA. (2019b). *Climate Change Initiative+ (CCI+) Phase 1 Sea Surface Salinity System Requirement Document (SRD)*. https://climate.esa.int/sites/default/files/filedepot/SSS_cci-D3.1-SRD-v1.1-signed.pdf

ESA. (2020). *Climate Change Initiative Extension (CCI+) Phase 1 New Essential Climate Variables (NEW ECVS) High Resolution Land Cover ECV System Requirement Document (SRD)*. https://climate.esa.int/media/documents/CCI_HRLC_Ph1-D3.1_SRD_v2.1.pdf

ESA. (2021a). *CCI+ Phase 1 Permafrost D3.1 System Requirement Document (SRD)*. https://climate.esa.int/media/documents/CCI_PERMA_SRD_v3.0.pdf

ESA. (2021b). *Water Vapour Climate Change Initiative - CCI+ Phase 1 System Requirements Document (SRD)*. https://climate.esa.int/media/documents/Water_Vapour_cci_D3.1_SRD_v3.0.pdf

Ethayarajh, K. (2020, March 24). *BERT, ELMo, & GPT-2: How Contextual are Contextualized Word Representations? | SAIL Blog*. The Stanford AI Lab Blog. https://ai.stanford.edu/blog/contextual/

Ezzini, S., Abualhaija, S., Arora, C., Sabetzadeh, M., & Briand, L. C. (2021). Using domain-specific corpora for improved handling of ambiguity in requirements. *Proceedings - International Conference on Software Engineering*, 1485–1497. https://doi.org/10.1109/ICSE43902.2021.00133

Ferrari, A., Spagnolo, G. O., & Gnesi, S. (2017). PURE: A Dataset of Public Requirements Documents. *Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017*, 502–505. https://doi.org/10.1109/RE.2017.29

Ferrari, A., Zhao, L., & Alhoshan, W. (2021, May 25). NLP for Requirements Engineering: Tasks, Techniques, Tools, and Technologies. *43rd International Conference on Software Engineering*.

Garnelo, M., & Shanahan, M. (2019). Reconciling deep learning with symbolic artificial intelligence: representing objects and relations. *Current Opinion in Behavioral Sciences*, *29*, 17–23. https://doi.org/10.1016/J.COBEHA.2018.12.010

Gleich, B., Creighton, O., & Kof, L. (2010). Ambiguity Detection: Towards a Tool Explaining Ambiguity Sources. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6182 LNCS*, 218–232. https://doi.org/10.1007/978-3-642-14192-8_20

Graves, A. (2012). *Supervised Sequence Labelling with Recurrent Neural Networks* (Vol. 385). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-24797-2

Gray, B. (2013). More than discipline: uncovering multi-dimensional patterns of variation in academic research articles. *Corpora*, *8*(2), 153–181. https://doi.org/10.3366/cor.2013.0039

Greff, K., Srivastava, R. K., Koutnik, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2222–2232. https://doi.org/10.1109/TNNLS.2016.2582924

Gu, Y., Tinn, R., Cheng, H., Lucas, M., Usuyama, N., Liu, X., Naumann, T., Gao, J., & Poon, H. (2021). Domain-Specific Language Model Pretraining for Biomedical Natural Language Processing. *ACM Transactions on Computing for Healthcare (HEALTH)*, *3*(1). https://doi.org/10.1145/3458754

Gururangan, S., Marasovic, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., & Smith, N. A. (2020). *Don't Stop Pretraining: Adapt Language Models to Domains and Tasks*. 8342–8360. https://doi.org/10.18653/V1/2020.ACL-MAIN.740

Guthrie, L., Pusteljovsky, J., Wilks, Y., & Slator, B. M. (1996). The Role of Lexicons in Natural Language Processing. *Communications of the ACM*, *39*(1), 63–72. https://doi.org/10.1145/234173.234204

Ha Thanh, N., & le Minh, N. (2021). *Sublanguage: A Serious Issue Affects Pretrained Models in Legal Domain*.

Haibe-Kains, B., Adam, G. A., Hosny, A., Khodakarami, F., Shraddha, T., Kusko, R., Sansone, S. A., Tong, W., Wolfinger, R. D., Mason, C. E., Jones, W., Dopazo, J., Furlanello, C., Waldron, L., Wang, B., McIntosh, C., Goldenberg, A., Kundaje, A., Greene, C. S., … Aerts, H. J. W. L. (2020). Transparency and reproducibility in artificial intelligence. *Nature 2020 586:7829*, *586*(7829), E14–E16. https://doi.org/10.1038/s41586-020-2766-y

Hamilton, W. L., Clark, K., Leskovec, jure, & Jurafsky, D. (n.d.). *SocialSent: Domain-Specific Sentiment Lexicons*. Retrieved April 14, 2022, from https://nlp.stanford.edu/projects/socialsent/

Hey, T., Keim, J., Koziolek, A., & Tichy, W. F. (2020). NoRBERT: Transfer Learning for Requirements Classification. *2020 IEEE 28th International Requirements Engineering Conference (RE)*, *2020-August*, 169–179. https://doi.org/10.1109/RE48521.2020.00028

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/NECO.1997.9.8.1735

Howard, J., & Ruder, S. (2018). Universal Language Model Fine-tuning for Text Classification. *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, *1*, 328–339. https://doi.org/10.18653/V1/P18-1031

Huldt, T., & Stenius, I. (2019). State-of-practice survey of model-based systems engineering. *Systems Engineering*, *22*(2), 134–145. https://doi.org/10.1002/SYS.21466

INCOSE. (2007). *Systems Engineering Vision 2020*.

INCOSE. (2015). *Systems Engineering Handbook* (D. Walden, G. Roedler, K. Forsberg, R. Hamelin, & T. Shortell, Eds.; Fourth). Wiley.

INCOSE. (2021). *Systems Engineering Vision 2035*. www.incose.org/sevision

Joachims, T. (1998). *Text categorization with Support Vector Machines: Learning with many relevant features*. 137–142. https://doi.org/10.1007/BFB0026683

Jones, N. (2014). Computer science: The learning machines. *Nature*, *505*(7482), 146–148. https://doi.org/10.1038/505146A

Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., & Wu, Y. (2016). *Exploring the Limits of Language Modeling*. http://arxiv.org/abs/1602.02410

Kim, S. Y., Wagner, D., & Jimenez, A. (2019, September). Challenges in Applying Model-Based Systems Engineering: Human-Centered Design Perspective. *INCOSE Human Systems Integration 2019 Conference*.

Kittredge, R., & Lehrberger, J. (1982). *Sublanguage: Studies of language in restricted semantic domains*. W. de Gruyter.

Kof, L. (2005). Natural Language Processing: Mature Enough for Requirements Documents Analysis? *Lecture Notes in Computer Science*, *3513*, 91–102. https://doi.org/10.1007/11428817_9

Kreuz, R. J., & Caucci, G. M. (2007). Lexical Influences on the Perception of Sarcasm. *Proceedings of the Workshop on Computational Approaches to Figurative Language*, 1–4. https://aclanthology.org/W07-0101/

Labelbox. (2022). *Labelbox*. https://labelbox.com/

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., & Bizer, C. (2012). DBpedia-A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web*, *1*, 1–5. http://www.alexa.com/topsites.

Levesque, H. J. (2014). On our best behaviour. *Artificial Intelligence*, *212*(1), 27–35. https://doi.org/10.1016/J.ARTINT.2014.03.007

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V., & Allen, P. G. (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. https://doi.org/10.48550/arxiv.1907.11692

Logan, P., Harvey, D., & Spencer, D. (2012). Documents are an Essential Part of Model Based Systems Engineering. *INCOSE International Symposium*, *22*(1), 1899–1913. https://doi.org/10.1002/J.2334-5837.2012.TB01445.X

Madni, A. M., & Sievers, M. (2018). Model-based systems engineering: Motivation, current status, and research opportunities. *Systems Engineering*, *21*(3), 172–190. https://doi.org/10.1002/SYS.21438

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., & McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 55–60. https://doi.org/10.3115/v1/P14-5010

MarketAndMarkets. (2021). *Artificial Intelligence in Healthcare Market by Offering (Hardware, Software, Services), Technology (Machine Learning, NLP, Context-aware Computing, Computer Vision), Application, End User and Geography - Global Forecast to 2027*.

MarketsAndMarkets. (2019). *Legal AI Software Market by Component (Solutions and Services), Deployment Mode, Technology, End User (Corporate Legal Departments and Law Firms), Application (Legal Research, Contract Management, and eDiscovery), and Region - Global Forecast to 2024*.

McKinney, S. M., Karthikesalingam, A., Tse, D., Kelly, C. J., Liu, Y., Corrado, G. S., & Shetty, S. (2020). Reply to: Transparency and reproducibility in artificial intelligence. *Nature 2020 586:7829*, *586*(7829), E17–E18. https://doi.org/10.1038/s41586-020-2767-x

McKinney, S. M., Sieniek, M., Godbole, V., Godwin, J., Antropova, N., Ashrafian, H., Back, T., Chesus, M., Corrado, G. C., Darzi, A., Etemadi, M., Garcia-Vicente, F., Gilbert, F. J., Halling-Brown, M., Hassabis, D., Jansen, S., Karthikesalingam, A., Kelly, C. J., King, D., … Shetty, S. (2020). International evaluation of an AI system for breast cancer screening. *Nature 2020 577:7788*, *577*(7788), 89–94. https://doi.org/10.1038/s41586-019-1799-6

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems 26 (NIPS 2013)*.

Minaee, S. (2021). Deep Learning-based Text Classification: A Comprehensive Review. *ACM Comput. Surv*, *54*. https://doi.org/10.1145/3439726

Moitra, A., Siu, K., Crapo, A. W., Durling, M., Li, M., Manolios, P., Meiners, M., & McMillan, C. (2019). Automating requirements analysis and test case generation. *Requirements Engineering*, *24*(3), 341–364. https://doi.org/10.1007/S00766-019-00316-X/FIGURES/14

NASA. (2005). *WISE Mission Operations System (MOS) Requirements Document* (Vol. 7, Issue 1). https://wise2.ipac.caltech.edu/docs/doc_tree/proj/WISE_MOS_Reqts_D30571.pdf

NASA. (2007). *James Webb Space Telescope Mission Requirements Document*. https://spacese.spacegrant.org/uploads/Requirements%20Config/JWST%20Mission%20Requirements%20Document.pdf

NASA. (2019). *Gateway System Requirements*. https://ntrs.nasa.gov/citations/20190029153

NOAA, & NASA. (2019a). *Joint Polar Satellite System (JPSS) Ground Segment Data Product Specification (GSegDPS)*. https://www.nesdis.noaa.gov/s3/2022-03/474-01543_JPSS-GSegDPS_A.pdf

NOAA, & NASA. (2019b). *Joint Polar Satellite System (JPSS) JPSS Level 1 Requirements - J2 Follow-On Final*. https://www.nesdis.noaa.gov/s3/2022-03/L1RD.pdf

NOAA, & NASA. (2019c). *Joint Polar Satellite System (JPSS) Level 1 Requirements Document Supplement (L1RDS) - Final*. https://www.nesdis.noaa.gov/s3/2022-03/L1RDS.pdf

Norheim, J. J., Lim, S. C., Kerbrat, A., & Rebentisch, E. (2022). *Methods for Extracting Structured Data from Engineering Requirements using Natural Language Processing*.

Paullada, A., Raji, I. D., Bender, E. M., Denton, E., & Hanna, A. (2021). Data and its (dis)contents: A survey of dataset development and use in machine learning research. *Patterns*, *2*(11), 100336. https://doi.org/10.1016/J.PATTER.2021.100336

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. https://doi.org/10.3115/v1/D14-1162

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep Contextualized Word Representations. *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, *1*, 2227–2237. https://doi.org/10.18653/V1/N18-1202

Rabelo, J., Goebel, · Randy, Kim, M.-Y., Kano, Y., Yoshioka, M., & Satoh, K. (2022). Overview and Discussion of the Competition on Legal Information Extraction/Entailment (COLIEE) 2021. *The Review of Socionetwork Strategies 2022 16:1*, *16*(1), 111–133. https://doi.org/10.1007/S12626-022-00105-Z

Rabelo, J., Kim, M. Y., Goebel, R., Yoshioka, M., Kano, Y., & Satoh, K. (2020). A Summary of the COLIEE 2019 Competition. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12331 LNAI*, 34–49. https://doi.org/10.1007/978-3-030-58790-1_3/TABLES/8

Rabelo, J., Kim, M. Y., Goebel, R., Yoshioka, M., Kano, Y., & Satoh, K. (2021). COLIEE 2020: Methods for Legal Document Retrieval and Entailment. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12758 LNAI*, 196–210. https://doi.org/10.1007/978-3-030-79942-7_13

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). *Language Models are Unsupervised Multitask Learners*. https://github.com/codelucas/newspaper

Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., Rutherford, E., Hennigan, T., Menick, J., Cassirer, A., Powell, R., van den Driessche, G., Hendricks, L. A., Rauh, M., Huang, P.-S., … Irving, G. (2022). *Scaling Language Models: Methods, Analysis & Insights from Training Gopher*.

Riesener, M., Dölle, C., Becker, A., Gorbatcheva, S., Rebentisch, E., & Schuh, G. (2021). Application of natural language processing for systematic requirement management in model-based systems engineering. *INCOSE International Symposium*, *31*(1), 806–815. https://doi.org/10.1002/J.2334-5837.2021.00871.X

RITA. (2011). *Core System Requirements Specification (SyRS)*. https://www.its.dot.gov/meetings/pdf/CoreSystem_SE_SyRS_RevA%20(2011-06-13).pdf

Rogers, A., Kovaleva, O., & Rumshisky, A. (2020). A Primer in BERTology: What We Know About How BERT Works. *Transactions of the Association for Computational Linguistics*, *8*, 842–866. https://doi.org/10.1162/TACL_A_00349

Ryan, K. (1993). The role of natural language in requirements engineering. *Proceedings of the IEEE International Conference on Requirements Engineering*, 240–242. https://doi.org/10.1109/ISRE.1993.324852

Sahlgren, M. (2008). The distributional hypothesis. *Rivista Di Linguistica*, *20*(1). https://www.italian-journal-linguistics.com/app/uploads/2021/05/2_Sahlgren-1.pdf

Sainani, A., Anish, P. R., Joshi, V., & Ghaisas, S. (2020). Extracting and Classifying Requirements from Software Engineering Contracts. *Proceedings of the IEEE International Conference on Requirements Engineering*, *2020-August*, 147–157. https://doi.org/10.1109/RE48521.2020.00026

Sennrich, R., Haddow, B., & Birch, A. (2016). Neural Machine Translation of Rare Words with Subword Units. *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers*, *3*, 1715–1725. https://doi.org/10.18653/V1/P16-1162

Shaghaghian, S., Feng, L. Y., Jafarpour, B., & Pogrebnyakov, N. (2020). Customizing Contextualized Language Models for Legal Document Reviews. *Proceedings - 2020 IEEE International Conference on Big Data, Big Data 2020*, 2139–2148. https://doi.org/10.1109/BIGDATA50022.2020.9378201

Sinha, R., Pang, C., Martinez, G. S., Kuronen, J., & Vyatkin, V. (2015). Requirements-Aided Automatic Test Case Generation for Industrial Cyber-physical Systems. *2015 20th International Conference on Engineering of Complex Computer Systems (ICECCS)*, 198–201. https://doi.org/10.1109/ICECCS.2015.32

SKA Organisation. (2015a). SKA Phase 1 System (Level 1) Requirements Specification. In *2015*. https://www.skatelescope.org/wp-content/uploads/2014/03/SKA-TEL-SKO-0000008-AG-REQ-SRS-Rev06-SKA1_Level_1_System_Requirement_Specification-P1-signed.pdf

SKA Organisation. (2015b). *SKA1 Level 0 Science Requirements*. https://www.skatelescope.org/wp-content/uploads/2014/03/SKA-TEL-SKO-0000007_SKA1_Level_0_Science_RequirementsRev02-part-1-signed.pdf

Song, Y., Kołcz, A., & Gilez, C. L. (2009). Better Naive Bayes classification for high-precision spam detection. *Software: Practice and Experience*, *39*(11), 1003–1024. https://doi.org/10.1002/SPE.925

Spike, M. (2018). The evolution of linguistic rules. *Biology & Philosophy 2018 32:6*, *32*(6), 887–904. https://doi.org/10.1007/S10539-018-9610-X

Tagarelli, A., & Simeri, A. (2021). Unsupervised law article mining based on deep pre-trained language representation models with application to the Italian civil code. *Artificial Intelligence and Law*, 1–57. https://doi.org/10.1007/S10506-021-09301-8/TABLES/19

Tai, W., Kung, H. T., Dong, X., Comiter, M., & Kuo, C.-F. (2020). exBERT: Extending Pre-trained Models with Domain-specific Vocabulary Under Constrained Training Resources. *Findings of the Association for Computational Linguistics: EMNLP 2020*, 1433–1439. https://doi.org/10.18653/v1/2020.findings-emnlp.129

ThalesAleniaSpace. (2009). *GALILEO GALILEI (GG) Mission Requirements Document*. http://eotvos.dm.unipi.it/temp/OLD/GG%20PRR%20DataPack/01%20-%20SD-TN-AI-1167_2_GG%20Mission%20Requirements%20Document.pdf

TMT. (2021a). *Thirty Meter Telescope Observatory Architecture Document*. https://www.tmt.org/download/Document/215/original

TMT. (2021b). *Thirty Meter Telescope Operations Requirements Document*. https://www.tmt.org/download/Document/15/original

TMT. (2021c). *Thirty Meter Telescope Science Requirements Document*. https://www.tmt.org/download/Document/11/original

UN. (n.d.). *Technical Requirements Document Mobile Surveillance System*. Retrieved June 4, 2022, from https://www.iom.int/sites/g/files/tmzbdl486/files/procurement/Technical%20Specifications_Mobile%20Surveillance%20System_Item%203.pdf

University of Ottawa. (n.d.). *PROMISE Software Engineering Repository*. Retrieved April 29, 2022, from http://promise.site.uottawa.ca/SERepository/

University of Texas. (1998). *Gravity Recovery and Climate Experiment (GRACE) Science & Mission Requirements Document*. https://geodesy.geology.ohio-state.edu/course/refpapers/Grace_smrd.pdf

US DOC, NOAA, & NESDIS. (2018a). *Joint Polar Satellite System (JPSS) National Environmental Satellite, Data, and Information Service (NESDIS) Environmental Satellite Processing Center (ESPC) Requirements Document (JERD) Volume 1*.

US DOC, NOAA, & NESDIS. (2018b). *Joint Polar Satellite System (JPSS) National Environmental Satellite, Data, and Information Service (NESDIS) Environmental Satellite Processing Center (ESPC) Requirements Document (JERD) Volume 2: Science Requirements*. https://www.nesdis.noaa.gov/s3/2022-03/JERDV2_Version_3_Updated_11292019-mcl-FinalDRAFT-mcl.pdf

Uszkoreit, J. (2017, August 31). *Google AI Blog: Transformer: A Novel Neural Network Architecture for Language Understanding*. https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Varenov, V., & Gabdrahmanov, A. (2021). Security Requirements Classification into Groups Using NLP Transformers. *Proceedings of the IEEE International Conference on Requirements Engineering*, 444–450. https://doi.org/10.1109/REW53955.2021.9714713

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, 6000–6010.

Walls, B., Bouchez, A., Goodrich, B., Cox, M., Milan-Gabet, R., Molgo, J., Peng, C., Santana, A., Sitarski, B., Souza, A., & Angeli, G. (2021). *GMT Observatory Requirements Document: GMT Requirements Document*. https://giantmagellan.org/wp-content/uploads/2022/04/GMT-DOC-03214_Observatory-Requirements-Rev.-E.pdf

Walls, B., Sitarski, B., Santana, A., Angeli, G., Bouchez, A., Goodrich, R., Bigelow, B., Millan-Gabet, R., Mcirwin, O., Souza, J., Souza, A., & Cox, M. (2021). *GMT Observatory Architecture Document: GMT Requirements Document*. https://giantmagellan.org/wp-content/uploads/2022/04/GMT-REQ-03215_Observatory-Architecture-Rev-G..pdf

Wang, L., & Ling, W. (2016). Neural Network-Based Abstract Generation for Opinions and Arguments. *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference*, 47–57. https://doi.org/10.18653/V1/N16-1007

Wang, Y., Liu, S., Afzal, N., Rastegar-Mojarad, M., Wang, L., Shen, F., Kingsbury, P., & Liu, H. (2018). A comparison of word embeddings for the biomedical natural language processing. *Journal of Biomedical Informatics*, *87*, 12–20. https://doi.org/10.1016/J.JBI.2018.09.008

Wang, Y., Shi, L., Li, M., Wang, Q., & Yang, Y. (2022). Detecting coreferent entities in natural language requirements. *Requirements Engineering*, *1*, 1–23. https://doi.org/10.1007/S00766-022-00374-8/TABLES/5

Webersinke, N., Kraus, M., Bingler, J. A., & Leippold, M. (2021). *ClimateBert: A Pretrained Language Model for Climate-Related Text*. http://arxiv.org/abs/2110.12010

Wettig, A., Gao, T., Zhong, Z., & Chen, D. (2022). *Should You Mask 15% in Masked Language Modeling?* https://doi.org/10.48550/arxiv.2202.08005

Wilks, Y. (1993). *Providing Machine Tractable Dictionary Tools*. 341–401. https://doi.org/10.1007/978-94-011-1972-6_16

Wu, Y., Schuster, M., Chen, Z., Le, Q. v., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., … Dean, J. (2016). *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. https://doi.org/10.48550/arxiv.1609.08144

Yang, L., Cormican, K., & Yu, M. (2019). Ontology-based systems engineering: A state-of-the-art review. *Computers in Industry*, *111*, 148–171. https://doi.org/10.1016/J.COMPIND.2019.05.003

Zhang, X., & LeCun, Y. (2015). *Text Understanding from Scratch*. https://doi.org/10.48550/arxiv.1502.01710

Zhao, L., Alhoshan, W., Ferrari, A., Letsholo, J., Ajagbe, M. A., Ajagbe, M. A., Batista-Navarro, R. T., Letsholo, K. J., & Chioasca, E.-V. (2021). Natural Language Processing for Requirements Engineering. *ACM Computing Surveys (CSUR)*, *54*(3). https://doi.org/10.1145/3444689

Zhao, L., Ferrari, A., Faedo, " A, Pisa, ", Keletso, I., Letsholo, J., Ajagbe, M. A., Ajagbe, M. A., Batista-Navarro, R. T., Alhoshan, W., Letsholo, K. J., Chioasca, E.-V., & Batista, R. T. (2021). Natural Language Processing for Requirements Engineering: A Systematic Mapping Study. *ACM Computing Surveys*, *54*(3). https://doi.org/10.1145/3444689

Zheng, L., Guha, N., Anderson, B. R., Henderson, P., & Ho, D. E. (2021). When does pretraining help? Assessing Self-Supervised Learning for Law and the CaseHOLD Dataset of 53,000+ Legal Holdings. *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*, *21*, 159–168. https://doi.org/10.1145/3462757.3466088

Zhong, H., Xiao, C., Tu, C., Zhang, T., Liu, Z., & Sun, M. (2020). *How Does NLP Benefit Legal System: A Summary of Legal Artificial Intelligence*. 5218–5230. https://doi.org/10.18653/V1/2020.ACL-MAIN.466