A Language-model-based Approach for Detecting Incompleteness in Natural-language Requirements

by

Dipeeka Luitel

Thesis submitted to the University of Ottawa

in partial fulfillment of the requirements for the degree of

Master of Computer Science

in

Computer Science Concentration in Applied Artificial Intelligence

Examining Committee

The following served on the Examining Committee for this thesis.

Carleton Member: Majid Komeili

Assistant Professor, School of Computer Science

Carleton University

Internal Member(s): Diana Inkpen

Professor, School of Electrical Engineering and Computer Science

University of Ottawa

Supervisor(s): Mehrdad Sabetzadeh

Associate Professor, School of Electrical Engineering and Computer Science

University of Ottawa

Declaration of Authorship

I hereby certify that this thesis is entirely my own original work except where otherwise indicated. I am aware of the University of Ottawa regulations concerning plagiarism, including those regarding consequent disciplinary actions. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

Abstract

[Context and motivation] Incompleteness in natural-language requirements is a challenging problem. **[Question/Problem]** A common technique for detecting incompleteness in requirements is checking the requirements against external sources. With the emergence of language models such as BERT, an interesting question is whether language models are useful external sources for finding potential incompleteness in requirements. **[Principal ideas/results]** We mask words in requirements and have BERT's masked language model (MLM) generate contextualized predictions for filling the masked slots. We simulate incompleteness by withholding content from requirements and measure BERT's ability to predict terminology that is present in the withheld content but absent in the content disclosed to BERT. **[Contributions]** BERT can be configured to generate multiple predictions per mask. Our first contribution is to determine how many predictions per mask is an optimal trade-off between effectively discovering omissions in requirements and the level of noise in the predictions. Our second contribution is devising a machine learning-based filter that post-processes predictions made by BERT to further reduce noise. We empirically evaluate our solution over 40 requirements specifications drawn from the PURE dataset [30]. Our results indicate that: (1) predictions made by BERT are highly effective at pinpointing terminology that is missing from requirements, and (2) our filter can substantially reduce noise from the predictions, thus making BERT a more compelling aid for improving completeness in requirements.

Acknowledgements

I am deeply grateful to all the people who have supported me throughout my journey of completing this Master's thesis. Without their constant encouragement and assistance, this accomplishment would not have been possible. First and foremost, I would like to express my heartfelt gratitude to my thesis supervisor, Dr. Mehrdad Sabetzadeh, and Shabnam Hassani for their support, invaluable guidance, and patience throughout this research endeavour. I would like to thank my family for their unwavering love and encouragement throughout my academic journey. Their sacrifices and belief in me have been the driving force behind my success. Finally, I would like to thank Luna, who has been with me throughout every step of this adventure.

Table of Contents

List of Tables

List of Figures

Abbreviations

AI: Artificial Intelligence

BERT: Bidirectional Encoder Representations from Transformers

CSL: Cost-Sensitive Learning

DT: Decision Tree

GLoVE: Global Vectors for Word Representations

IG: Information Gain

LM: Language Model

LR: Logestic Regression

LSTM: Long Short-Term Memory

ML: Machine Learning

MLM: Masked Language Modelling

NER: Named Entity Recognition

NLP: Natural Language Processing

NN: Neural Network

NSP: Next Sentence Prediction

PURE: PUblic REquirements dataset

RF: Random Forest

RE: Requirements Engineering

RS: Requirements Specifications

SVD: Singular Value Decomposition

SVM: Support Vector Machine

TF-IDF: Term Frequency-Inverse Document Frequency

Chapter 1

Introduction

Improving the completeness of requirements is an important yet challenging problem in requirements engineering (RE) [62]. The RE literature distinguishes two notions of completeness [62]: (1) *Internal* completeness is concerned with requirements being closed with respect to the functions and qualities that one can infer exclusively from the requirements. (2) *External* completeness is concerned with ensuring that requirements are encompassing of all the information that external sources of knowledge suggest the requirements should cover. These external sources can be either people (stakeholders) or artifacts, e.g., higher-level requirements and existing system descriptions [4]. External completeness is a relative measure, since the external sources may be incomplete themselves or not all the relevant external sources may be known [62]. Although external completeness cannot be defined in absolute terms, relevant external sources, when available, can be useful for detecting missing requirements-related information.

1.1 Motivation

When the requirements and the external sources of knowledge are textual, one can leverage natural language processing (NLP) for computer-assisted checking of external completeness. For example, Ferrari et al. [27] use NLP for checking the completeness of requirements against stakeholder-interview transcripts. And, Dalpiaz et al. [19] use NLP alongside visualization to identify differences among stakeholders' viewpoints; these differences are then investigated as potential incompleteness issues.

With modern pre-trained language models (LMs), a new opportunity arises for NLP-based improvement of external completeness in requirements: Using self-supervised learning, (pre-trained) LMs have been trained on very large corpora of textual data, e.g., millions of Wikipedia articles. This raises the prospect that *a LM can serve as an external source of knowledge for completeness checking*. In this thesis, we explore a specific instantiation of this idea using BERT [20].

BERT has been trained to predict masked tokens by finding words or phrases that most closely match the surrounding context. To illustrate how BERT can help with completeness checking of requirements, consider the example in Fig. 1.1. In this example, we have masked one word, denoted `[MASK]`, in each of requirements R1, R2 and R3. We have then had BERT make five predictions for filling each masked word. For instance, in R1, the masked word is *disruptions*. The predictions made by BERT are: *changes*, *problems*, *delays*, *defects*, and *risks*. As seen from the figure, one of these predictions, namely *delay*, is a word that

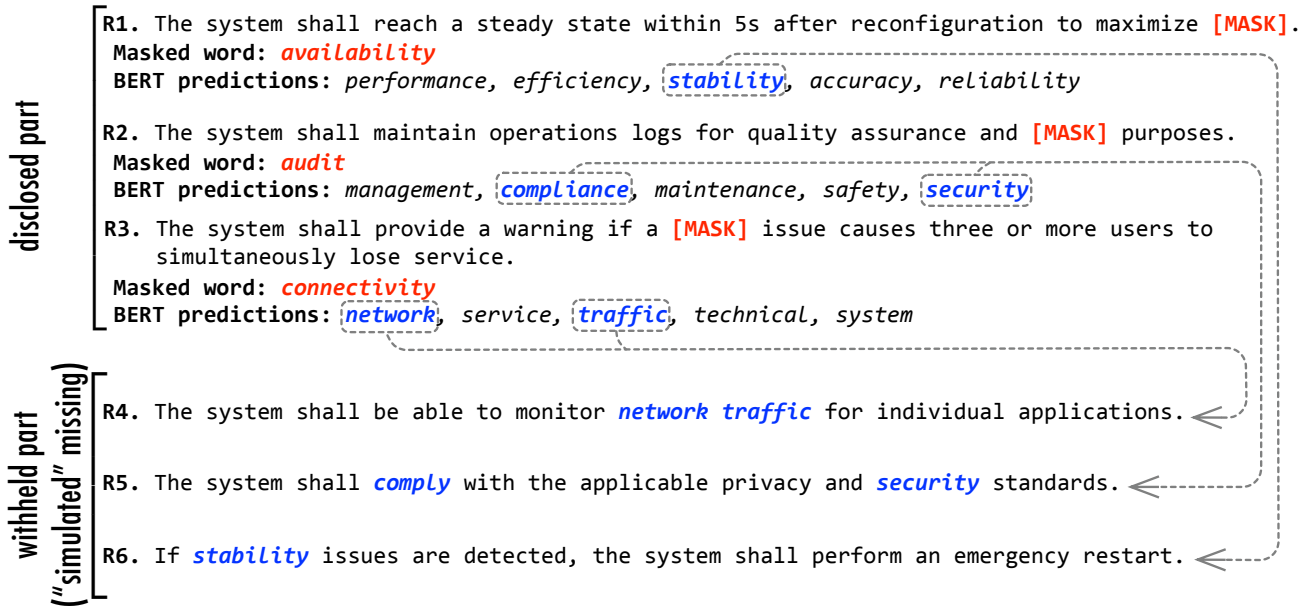

Figure 1.1: Illustrative requirements specification.

appears in R6. Similarly, the predictions that BERT makes for the masked words in R2 and R3 (*transmission* and *audit*) reveal new terminology that is present in R4 and R5 (*comply* and *reliability*). In the above example, ***if requirements R4–R6 were to be missing, BERT's predictions over R1–R3 would provide useful cues about some of the missing terminology.***

1.2 Overall Hypothesis and Findings

Considering the potential highlighted in our motivation for employing LMs, we hypothesize that LMs are a useful source of external knowledge to assess and enhance the completeness

of requirements. To examine this hypothesis, we develop a technical solution based on BERT. In order to systematically explore the hypothesis and measure the usefulness of LMs, we define the following research questions over the solution developed.

- **RQ1.** How accurately can LMs predict missing relevant terminology in an input requirement specification?
- **RQ2.** Does a LM-based approach outperform simpler baselines?
- **RQ3.** How effective are filters at eliminating irrelevant predictions generated by LMs?
- **RQ4.** To what extent are predictions generated by LMs, combined with post-process filtering, effective in detecting incompleteness in requirements?
- **RQ5.** To what extent does the amount of incompleteness in a requirements specification affect the quality of predictions produced by LMs?

With regard to RQ1, we observe that BERT can provide hints for approximately 4 in 10 (simulated) omissions. Approximately 1 out of 8 hints turn out to be relevant.

With regard to RQ2, while baselines sometimes generate proportionally fewer non-relevant hints, a LM-based approach generates significantly more relevant hints than all the baselines considered.

With regard to RQ3, the use of filters is effective in removing unuseful hints. Nevertheless, there are choices to be made about the level of aggression of filtering in order to

strike a balance between relevant hints and non-relevant hints (noise).

With regard to RQ4, we observe that: Using a strict filter, roughly half of the recommendations made by our approach are relevant and hint at approximately 20% of (simulated) missing terminology. Using a lenient filter, approximately one in four recommendations made by our approach are relevant and hint at approximately 35% of (simulated) missing terminology.

With regard to RQ5, our results indicate that a lower incidence of incompleteness does not impact a LM's ability to generate useful hints but does lead to an increase in the number of unuseful hints.

1.3 Contributions

In order to improve the performance of BERT in predicting relevant terminology that is absent from requirements, it is important to develop a strategy that simulates missing information. This can be achieved by randomly withholding a portion of a given requirements specification, while disclosing the remainder to BERT for obtaining masked-word predictions. For example, in Fig. 1.1, requirements R4–R6 would be withheld, while requirements R1–R3 would be disclosed to BERT for prediction. The withheld part *simulates* requirements omissions, which may occur in real-world scenarios. Masking words in the disclosed part and having BERT make predictions for the masks reveals some terms that appear only in the withheld part. BERT can be configured to generate multiple predictions

per mask. Our ***first contribution*** is to determine how many predictions per mask is an optimal trade-off between effectively discovering simulated omissions and the amount of unuseful predictions (noise) that BERT generates.

We observe that a large amount of noise would result if predictions by BERT are to achieve good coverage of requirements omissions. Some of the noise is trivial to filter. For instance, in the example of Fig. 1.1, we can dismiss the prediction of *problems* (made over R1); this word already appears in the disclosed portion (specifically, in R2), thereby providing no cues about missing terminology. Also, several predictions, e.g., *other* and *similar*, can be filtered, since they carry little meaning. After applying these obvious filters, one would still be left with considerable noise. Our ***second contribution*** is a machine learning-based filter that post-processes predictions by BERT to strike a better balance between noise and useful predictions. Our solution development and evaluation is based on 40 requirements specifications from the PURE dataset [30]. These specifications contain over 23,000 sentences combined. To facilitate replication and further research, we make our implementation and evaluation artifacts publicly available [42].

1.4 Publications

We have published the following conference paper based on the thesis research:

[43] Luitel, D., Hassani, S., Sabetzadeh, M. (2023). Using Language Models for Enhancing the Completeness of Natural-Language Requirements. In: Ferrari, A., Penzen-

stadler, B. (eds) Requirements Engineering: Foundation for Software Quality. REFSQ 2023. Lecture Notes in Computer Science, vol 13975. Springer, Cham.

We are currently working on a journal extension as part of our ongoing research.

1.5 Thesis Organization

The remaining chapters of this thesis are organized as follows: Chapter 2 provides background on LMs and other essential concepts relevant to this research. In Chapter 3, we discuss related works within the RE literature, exploring various methods for completeness checking of natural language requirements and examining different NLP techniques employed in the realm of RE. Chapter 4 presents our approach, where each step used in detecting incompleteness is described in detail. The experimental design and analysis are presented in Chapter 5, offering insights into our evaluation methodology and results. Finally, Chapter 6 presents the main conclusions drawn from our research and explores potential areas of future work in this area.

Chapter 2

Background

Below, we review the background for our work, covering the NLP pipeline, language models, word embeddings, machine learning and corpus extraction.

2.1 NLP Pipeline

NLP refers to a branch of computer science and more specifically, artificial intelligence (AI) [45], and is defined as the automatic analysis and representation of natural language, like speech and text, by software [60]. There are many applications of NLP, including conversational agents, chatbots and dialogue systems, machine translation, and web-based question answering [36].

NLP is usually performed using a pipeline of modules [35]. A standard pipeline, as

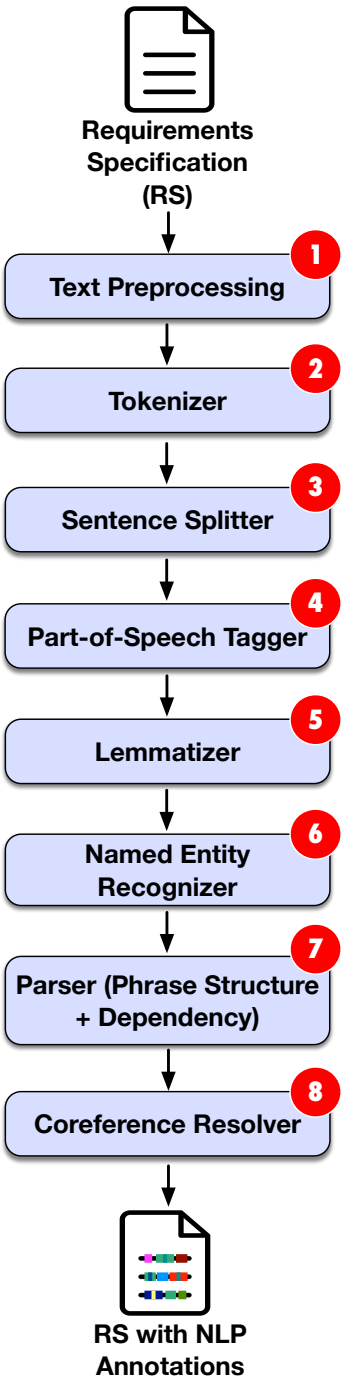

Figure 2.1: NLP Pipeline.

shown in Figure 2.1, may be composed of a variety of modules such as text preprocessing, part-of-speech (POS) tagging, named entity recognition (of organizations, people, and places), parsing sentences to their grammatical structures, and identifying co-references between noun phrases [36]. The exact pipeline can vary depending on the specific NLP task and the tools or models used to perform it. The output of the NLP pipeline is an annotated requirement specification. We use the annotations produced by the NLP pipeline for several purposes, including the identification and lemmatization of terms in requirements documents as well as processing predictions made by BERT in their surrounding context. The steps in an NLP are as follows:

2.1.1 Text preprocessing

In this step, raw text is cleaned and prepared for further analysis. This may involve removing irrelevant characters or symbols, converting text to lowercase, and removing stop words (common words like 'the', 'and', 'is'). This helps to reduce noise and makes it easier for the model to process the text.

2.1.2 Tokenization

Tokenization is the process of breaking the text into individual words or tokens. Tokens are the basic units of text that the model processes. Tokenization can be performed using various techniques such as whitespace-based, rule-based, or statistical methods. For

example, the following sentence “The model shall be implemented in Python.” would be tokenized using whitespace as: ‘the’, ‘model’, ‘shall’, ‘be’, ‘implemented’, ‘in’, ‘Python’ and ‘.’.

2.1.3 Sentence splitting

This step involves breaking the text into individual sentences. This is particularly important for tasks where sentence-level information is required, such as sentiment analysis or text summarization. Sentence splitting typically uses punctuation marks, capitalization, and other cues to identify sentence boundaries. However, the structure of what constitutes a sentence is predefined and may not necessarily be grammatically correct, in part due to the structure of requirement documents.

2.1.4 Part-of-speech (POS) tagging

In this step, each token is assigned a POS label, such as noun, verb, adjective, adverb, etc. This information is useful for understanding the syntactic structure and the role each word plays within a sentence. Continuing with the previous example,“The model shall be implemented in Python.” would have the following POS tag for each word: ‘The’: DT (determiner), ‘model’: NN (singular noun), ‘shall’: MD (modal auxiliary verb), ‘be’: VB (base form verb), ‘implemented’: VBN (past participle verb), ‘in’: IN (preposition), ‘Python’: NNP (singular proper noun), and ‘.’ – PUNCT (punctuation).

2.1.5 Lemmatization

During the lemmatization step, words are reduced to their base form, known as the lemma. This process helps normalize words with different inflections, allowing the pipeline to treat them as the same entity. Lemmatization improves text analysis by reducing vocabulary size and ensuring words with the same root meaning are treated equally. For example, the lemma for both 'running' and 'ran' is 'run'.

2.1.6 Named Entity Recognition (NER)

NER is the process of identifying and classifying real-world objects (entities) mentioned in the text, such as people, organizations, locations, dates, or quantities. This step helps extract structured information from unstructured text and is crucial for tasks like information extraction and question-answering systems.

2.1.7 Parsing

Parsing involves determining the grammatical structure of a sentence. There are two main types of parsing: **Phrase Structure**: Constituency parsing aims to represent the hierarchical structure of a sentence using a parse tree. The tree consists of nodes representing constituents (phrases or groups of words) and their relationships to each other. **Dependency Parsing**: Dependency parsing, on the other hand, focuses on representing the syntactic dependencies between words in a sentence. In a dependency graph, nodes are

words, and directed edges represent the dependencies, such as subject-verb, object-verb, etc. This information can be helpful in tasks like information extraction.

2.1.8 Coreference Resolver

Coreference resolution is the process of identifying and linking mentions of the same entity within a text. Coreference resolution aims to find such links and replace the pronouns or other referring expressions with their respective entities. This step can improve the performance of downstream tasks like text summarization or question-answering systems. For example, in the sentence "The application must be compatible with Windows 10. It should also run smoothly on this operating system." In this example, the pronoun 'It' refers to 'the application' and the term 'this operating system' refers to 'Windows 10'. Coreference resolution aims to find such links and would replace the referring expressions with their respective entities, resulting in: "The application must be compatible with Windows 10. The application should also run smoothly on Windows 10."

2.2 Language Models

Deep learning has been a popular approach for various NLP tasks including sentiment analysis, question answering and natural language inference [47]. Recent NLP approaches heavily rely on deep learning, and in particular, transfer learning [20]. Bidirectional Encoder Representations from Transformers (BERT) is a pre-trained language model using

two unsupervised tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP). BERT uses a bidirectional encoder to generate context-aware word representations. The encoder consists of a stack of Transformer blocks that use self-attention mechanisms to encode the input sequence in both directions [20]. The bidirectional transformer architecture of BERT with a stacked self-attention encoder-decoder structure enables the parallelization of training and makes it different from past LMs such as ELMO [50], which is a long short-term memory (LSTM) neural network [56]. Parallelization of training in BERT refers to the process of distributing the training workload across multiple computing resources, such as multiple CPUs or GPUs. This allows BERT to train more efficiently and reduce the overall training time. In BERT architecture, the self-attention layers increase weight values of relevant words to reduce the influence of irrelevant words in the outcome [20, 56]. BERT has become one of the most widely used NLP models and has inspired several variants. Some of the most notable BERT variants are as follow:

RoBERTa [40] is a variant of BERT that was developed by Facebook AI. RoBERTa is pre-trained on a larger corpus of text data, and fine-tuned using a different approach compared to BERT. RoBERTa has achieved state-of-the-art results on several NLP benchmark datasets. ALBERT [38] is another variant of BERT which is developed with the goal of reducing the number of parameters while preserving performance. ALBERT achieves this by introducing a factorized embedding parameterization and cross-layer parameter sharing.

DistilBERT [52] is a smaller and faster variant of BERT that is developed by Hugging Face. DistilBERT is trained to have a similar performance to BERT while being faster and

having fewer parameters. Efficiently Learning an Encoder that Classifies Token Replacements Accurately (ELECTRA) [17] uses replaced token detection to better understand the relationship between words in a sentence, leading to improved performance on downstream natural language processing tasks. Replaced token detection is performed by corrupting some of input tokens and training the model to predict whether these tokens have been replaced or not. Additionally, ELECTRA is designed to be computationally efficient, making it easier to train on large amounts of data.

MiniLM [57] is based on BERT's architecture and uses a similar pre-training objective of masked language modelling, where some of the words in a sentence are masked and the model is trained to predict the missing words. However, MiniLM uses a smaller vocabulary and fewer layers than BERT, allowing it to be more computationally efficient. It achieves high performance by using a combination of knowledge distillation, through training a smaller model to mimic the behavior of a larger one, and pruning, which involves removing unnecessary model parameters. Despite its small size, MiniLM is a powerful tool in completing a range of NLP tasks, including text classification, named entity recognition, and natural language inference. For the purposes of this thesis, the original BERT model is selected as this work focuses on predictions of individual tokens without domain specificity. Further research is required to assess the performance of BERT variants on our approach.

BERT Base and BERT Large are two types of the BERT model. BERT Large, while generally more accurate, requires more computational resources. BERT Base has 12 encoder layers with a hidden size of 768, 12 self-attention heads, and $\approx$110 million trainable

parameters. Further variations of BERT include cased and uncased models. For BERT uncased, the text has been lower-cased before tokenization, whereas in BERT cased, the tokenized text is the same as the input text. Previous RE research suggests that the cased model is preferred over uncased for analyzing requirements [23, 34]. To mitigate computation costs and follow best practices, the BERT-base-cased model is employed for our experimentation. With BERT's transfer learning capabilities, the pre-trained BERT model can be fine-tuned using labeled data from downstream tasks [20]. During our preliminary investigation, we observed the computational cost of fine-tuning outweighs any perceived improvement in the quality of words predicted. Therefore, this thesis employs the original pre-trained BERT model as discussed in Chapter 4. For the purposes of this thesis, only the MLM task is used to obtain relevant predicted words.

MLM, or the Cloze task, is a procedure of randomly masking a set percentage of tokens from a natural language input. Then, the model attempts to predict the original vocabulary of the masked token [20]. When feeding an input text to BERT, there are three special tokens to take into consideration. '[CLS]' is a classification token appended to the start of every input to demarcate the beginning of the text. '[SEP]' is a separator token to mark the end of one sentence from the beginning of another. Lastly, '[MASK]' is a masking token used only for the MLM task; '[MASK]' replaces the actual word of a sentence and predicts contextualized tokens likely to match the masked token. Continuing with the example from Section 2.1, the sentence "The model shall be implemented in Python." would be modified into "[CLS] The model shall be implemented in Python. [SEP]" before tokenization. If we

were to mask the word 'Python', the sentence would be updated to "[CLS] The model shall be implemented in [MASK]. [SEP]". The MLM task can also be used to predict probable alternative words for a masked element of an input sentence. Our goal is to utilize MLM for identifying closely related alternative words that may be relevant but are currently missing from an input RS.

2.3 Word Embeddings

In our work, we need a semantic notion of similarity that goes beyond lexical equivalence and allows us to further identify closely related terms; examples would be (i) 'key' and 'unlock', and (ii) 'encryption' and "security". For this, we use cosine similarity over *word embeddings*. Only using lexical equivalence to measure relationships between words can have limitations. Many words have multiple meanings, which can lead to ambiguity. Determining lexical equivalence based solely on word forms might incorrectly group words with different meanings, or fail to group words with the same meaning but different forms. The meaning of a word can also change depending on the context in which it is used. Lexical equivalence might not account for this contextual information, leading to an inaccurate understanding of word relationships. For these reasons, word embeddings are useful for capturing not only lexical information but also semantic relationships, contextual information, and other linguistic features. Cosine similarity quantifies the similarity of two words by calculating the distance between their vector representations [44]. The

formula to calculate cosine similarity between two words *d1* and *d2* is as follows: sim(d1, d2) = (V (d1) · V (d2)) / (|V (d1) || V (d2)|) [44]. To obtain a vector representation, we must first transform each word into its own word embedding. Word embeddings are mathematical representations of words as dense numerical vectors capturing syntactic and semantic regularities [46].

Comparing the words 'security' and 'privacy', there is a small distance between their vector representations, which indicates semantic similarity between the two words. 'Security' and 'compliance' are not semantically equivalent and would therefore have a large distance between their corresponding vectors. GloVe (Global Vectors for Word Representation) is an unsupervised machine learning model that creates word embeddings using a co-occurrence matrix [49]. The model constructs the matrix by counting the number of times words appear in each other's context and then derives a weighting factor based on the strength of the association between words. Using singular value decomposition (SVD), GloVe reduces the dimensionality of the word embeddings while preserving important relationships between words. Finally, the model uses an iterative algorithm to optimize the embeddings according to an objective function that prioritizes minimizing the difference between the dot product of two word vectors and the logarithm of their co-occurrence count, weighted by significance. We employ GloVe's pre-trained model to obtain word embeddings [49] and this decision is motivated by striking a trade-off between accuracy and efficiency. There are several other frameworks capable of text representation such as ELMo [50] and OpenAI [48]. BERT also generates word embeddings; however, these em-

beddings are expensive to compute because they take context into consideration. BERT embeddings thus do not scale well when a large number of pairwise term comparisons is required as is the case for this thesis.

2.4 Machine Learning (ML)

ML, a sub-field of computer science and AI, tries to imitate the way that humans learn and automatically improves through experiences. The two main tasks in machine learning are unsupervised learning and supervised learning. Unsupervised learning is a task using unlabeled data, meaning that the data is not pre-classified or labeled. Unsupervised learning uses this data to find patterns or relationships in the data without any prior knowledge of the output labels. Clustering, dimensionality reduction, and anomaly detection are examples of unsupervised learning. Supervised learning is trained on a labeled dataset, meaning that the data is already pre-classified or labeled. This task learns the relationship between the input features and the corresponding output labels. The goal is to use learned relationships to predict the correct output labels for new, unseen data. Supervised learning can be applied to a variety of tasks, such as classification, regression, and ranking, among others. In this thesis we utilize supervised learning for the purpose of classification. The empirical evaluation (see Chapter 5) examines several widely used ML classification algorithms, namely Neural Network (NN), Decision Tree (DT), Logistic Regression (LR), Random Forest (RF), and Support Vector Machine (SVM).

Traditional classification techniques rely on the extraction of relevant features from the input data. These features are then used to train a machine learning model to classify new data points into one of several pre-defined categories. The features can be numerical, categorical, or a combination of both. Numerical features are continuous or discrete values that describe some aspect of the data, such as a word's length or its frequency of appearance. Categorical features are discrete values that describe some categorical aspect of the data, such as a part-of-speech tag, and can be represented as binary, one-hot encoded vectors, or other types of encoding. There are also text and image features, but these fall outside of the scope of those used in our research. Feature selection is the process of selecting a subset of relevant features from the original set of features to improve the accuracy and efficiency of a machine learning model [15, 39]. The goal is to eliminate irrelevant or redundant features that do not contribute significantly to the model's performance, while retaining the most important features. Our features for learning and our process for creating labelled data are discussed in Chapters 4 and 5, respectively.

Classification models have a tendency to predict the more prevalent class(es) [58]. Furthermore, classification algorithms typically give equal treatment to different misclassification types when minimizing misclassification. In many problems, however, the costs associated with different misclassification are not symmetric. In our context, non-relevant terms outnumber relevant ones. We under-sample the majority class (i.e., non-relevant) to counter imbalance the training set and thereby reduce the risk of filtering useful information [11]. We also take Specifically, we want to assign a higher penalty to relevant

terms being filtered than non-relevant terms being classified as relevant. In other words, we prioritize recall over precision which is often necessary in RE tasks [12]. Misclassification costs can be taken into account either (1) during ML training or (2) after training. The former strategy is known as cost-sensitive learning (CSL) and the latter as cost-sensitive classification [26,32]. CSL refers to a specific set of algorithms that are sensitive to different costs associated with certain characteristics of a considered problem. CSL can improve accuracy by assigning different misclassification costs to different classes or types of errors, with the aim of minimizing the prevalence of certain types of misclassified data [26]. In our appraoch, filtering a false negative (i.e., 'relevant' prediction) is more detrimental than filtering a false positive (i.e., 'non-relevant' prediction). In this case, CSL can be used to adjust the model's parameters and bias it towards reducing the prevalence of false negatives, even if it leads to a higher number of false positives, to ensure that useful predictions do not get filtered.

2.5 Domain-corpus Extraction

Domain-specific corpora are useful resources for improving the accuracy of automation in RE [24]. The domain can be any specific subject or field, such as medicine, astronomy, or law. Domain-specific corpus extraction is the process of creating a corpus from existing texts or documents that are relevant to a particular domain. The first step in domain-corpus extraction is to define the scope of the domain and determine the specific sources

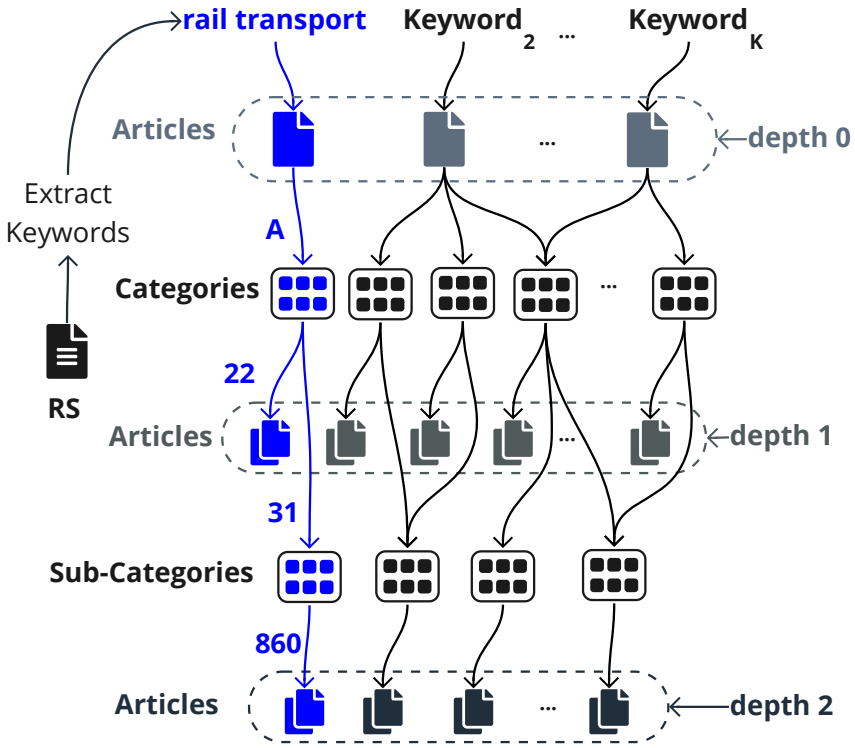

Figure 2.2: WikiDoMiner Structure [25].

that are relevant to the domain. If a domain-specific corpus already exists, there may be no need to generate a new corpus. Existing corpora can be acquired from books, magazines, and news organizations for tasks like handling anaphoric ambiguity [24, 25]. However, if a suitable corpus does not exist, then one can be extracted manually or automatically. The corpus can be constructed using domain documents from sources like Wikipedia which contains a vast repository of information and offers multiple articles for a given domain, allowing it to be a useful source for corpus generation. [18, 24, 25, 28].

Manual corpus extraction involves searching for and selecting relevant texts or docu-

ments by hand based on predetermined criteria. This process can be time-consuming and requires domain knowledge and expertise to ensure that the selected texts are representative of the domain. Automated corpus extraction navigates a collection of text according to pre-specified criteria to extract a corpus. This can be done using techniques such as keyword extraction, topic modelling, and named entity recognition. To avoid requiring any feedback from the user and minimize the cost of using our approach, we rely on automated extraction using the tool *WikiDoMiner* [25], which is specifically designed for domain-specific corpus generation by crawling Wikipedia. The tool first extracts keywords from an input specification and then assembles a set of Wikipedia articles relevant to the terminology, and thus the domain of the specification. WikiDoMiner allows control over corpus expansion through a depth parameter. A depth of zero creates a corpus of articles directly matching the key phrases in the input document, while increasing the depth results in larger corpora that encompass sub-categories of Wikipedia articles. We rely on the domain specific corpus extracted from WikiDoMiner to calculate feature-based statistics, as we discuss in Chapter 3.

Chapter 3

Related Works

As noted in Chapter 1, our work focuses on external completeness of requirements. Requirements completeness is characterized along internal and external dimensions. External completeness ensures all information pertinent to the definition of a proposed system is found within the specification. This includes checking for: (1) non-existent references, (2) missing specification items, and (3) missing functions in the specification checking against an external source. Sources may be either humans (e.g. stakeholders) or development artifacts (e.g. higher-level requirements, models, and existing system documentation) [7]. Several works have been presented in the literature to define and to measure completeness of requirements specifications. In this section, we compare our approach with the most pertinent related works in the RE literature.

3.1 Completeness Checking of NL Requirements

There are various methods for determining the completeness of NL requirements. España et al. [22] measure completeness by comparing use cases against information systems through communication analysis. The authors evaluate the level of completeness achieved by reviewed models against a reference model. Gigante et al. [31] use ontological engineering to determine completeness of requirements. They model requirements in the form of (subject, predicate, object) triplets. These triplets are then compared against an external source to verify completeness.

Eckhardt et al. [21] propose a framework consisting of a unified model for performance requirements and a content model to capture relevant content for addressing incompleteness. This process utilizes sentence patterns derived from the content model to evaluate completeness of requirements. Alrajeh et al. [2] create synthetic obstacles to verify completeness of requirements. The approach uses model checking and iteratively generates domain-specific obstacles, with the process ending after achieving a domain-complete set of obstacles. The above works cross validate requirements against an external model to measure completeness. Our approach seeks to address the same challenge by using a generative language model, BERT, and its vast pre-training data as a knowledge source for making contextualized predictions. Arora et al. [7] conduct a case study to detect external incompleteness of requirements using domain models. The authors simulate requirements omissions and demonstrate that UML class diagrams can display a near-linear sensitivity

to detecting missing and under-specified requirements. Dalpiaz et al. [19] develop a technique based on NLP and visualization to explore commonalities and differences between multiple viewpoints and thereby help stakeholders pinpoint occurrences of ambiguity and incompleteness. Differences may occur when terms appear in a single viewpoint, i.e., the situation where a viewpoint refers to concepts that do not appear in other viewpoints. In the above works, the sources of knowledge used for completeness checking are existing development artifacts. Our approach does not require any user-provided artifacts. We leverage BERT's capacity to predict relevant terminology, independently of supplementary artifacts, to ensure domain independence.

Bhatia et al. [14] address incompleteness in privacy policies by representing data actions as semantic frames. A semantic frame is constructed by identifying relevant questions for the data action, as semantic roles. Semantic roles represent the relationship of different clauses in statements to the main action. They identify the expected semantic roles for a given frame, and consequently determine incompleteness by identifying missing role values. Cejas et al. [3] use NLP and ML for completeness checking of privacy policies. Their approach identifies instances of pre-defined concepts such as "controller" and "legal basis" in a given policy. They create a conceptual model as a hierarchical representation of metadata types referring to different GDPR concepts based on hypothesis coding. It then verifies through rules whether all applicable concepts are covered. The above works deal with privacy policies only and have a predefined conceptual model for textual content. Our BERT-based approach is not restricted to a particular application domain and does not

have a fixed conceptualization of the textual content under analysis. Instead, we utilize BERT's pre-training and attention mechanism to make contextualized recommendations for improving completeness.

3.2 NLP for Requirements Engineering

Natural Language Processing for Requirements Engineering (NLP4RE) is a field that employs techniques from NLP to address challenges faced in the RE domain. Applications of NLP4RE include terminology extraction [33], defect detection [9], requirements similiarity and retrieval [1], requirements tracing [10], user story analysis [41], and legal requirements analysis [54].

The state of the art in NLP4RE has been extensively covered in a recent literature review by Zhao et al. [61]. The authors systematically examine 404 primary studies spanning over 36 years and provide a detailed breakdown of trends observed in the NLP4RE community. A majority of papers presented were solution proposals. However, of the 130 different tools, only 17 are publicly available. To facilitate further research, the artifacts from this thesis have been made publicly available. Furthermore, from the authors' list of reviewed materials, although two papers utilize BERT-based language modelling, neither paper focuses on evaluating completeness of requirements. The first paper by Hey et al. [34] presents a new method for unsupervised representation learning, called NORBERT (Non-iterative, Oblivious Representation of Bidirectional Encoder Representations from

Transformers). NORBERT can be used for a wide range of NLP tasks, particularly when labeled data is limited or unavailable. The second paper by Sainani et al. [51] use BERT for automating the extraction and classification of requirements from software engineering contracts.

Ambiguity and incompleteness of requirements are both common challenges in NL. Ferrari et al. [29] explore the detection and interpretation of ambiguous requirements using graph-based modelling created with domain-relevant documents. Interpretation of requirements may vary as users may not necessarily have equal background knowledge of a domain. Similarly, our work seeks to minimize incompleteness in requirements which may be exacerbated by user's varying levels of domain knowledge or command of the language used in writing requirements.

Shen and Breaux [53] propose an NLP-based approach for extracting domain knowledge from word embeddings and user-authored scenarios.Their approach involves gathering a corpus of scenarios authored by users in four distinct directory-service domains - apartments, hiking trails, restaurants, and health clinics. Then, the authors extract basic domain models from these scenarios by utilizing typed dependencies. The authors use seed question templates that include a domain-specific noun, seed verb and mask, and utilize MLM to predict substitute tokens for the mask. While this approach is not concerned with checking the completeness of requirements, it uses BERT's MLM for generating alternative entities by masking words in requirements statements. Our approach uses BERT's MLM in a similar manner. In contrast to the above work, we take steps to address the challenge

arising from such use of BERT over requirements, namely the large number of non-relevant alternatives (false positives) generated. We propose a ML-based filter that uses a combination of NLP and statistics extracted from a domain-specific corpus to reduce the incidence of false positives.

Chapter 4

Approach

Figure 4.1 presents an outline of our approach for recommending relevant terms that are missing from a given textual requirement specification (RS). The RS serves as the input to the approach, which consists of six main steps.

In the first step, the RS is parsed to extract necessary information. This parsing allows us to identify key entities, relationships, and concepts that are relevant to the RS. In the second step, we use BERT, a state-of-the-art language model, to generate predictions for masked words in the RS. These predictions are generated by masking specific words in the RS and predicting the most likely replacement for the masked word based on the context. In the third step, we filter out predictions that provide little or no additional information. For example, if a masked word is predicted to be a stop word or a word that is already present in the RS, we exclude it from our final recommendations. In the fourth step, we

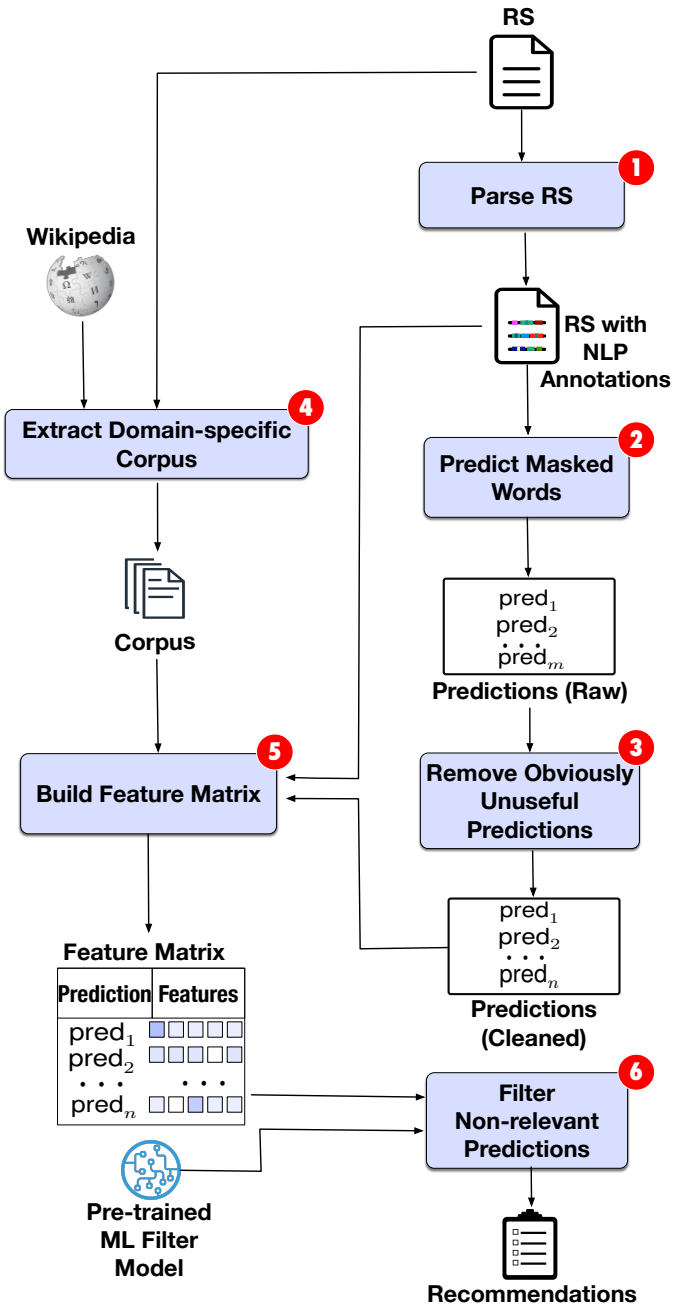

Figure 4.1: Approach Overview.

construct a domain-specific corpus for the given RS. This corpus comprises documents and resources related to the RS domain. Using this corpus and the results from Step 1, we identify the most relevant concepts and relationships that should be considered for the RS. In the fifth step, we build a feature matrix for ML-based filtering of non-relevant terms from predictions made by BERT. This feature matrix is constructed based on the identified concepts and relationships from Step 4 and the predictions generated by BERT in Step 2. In the sixth and final step, we feed the computed feature matrix to a pre-trained classifier. The classifier uses the features to remove noise, i.e., non-relevant words, from the predictions. The output of our approach is a list of recommended terms that are likely relevant to the RS but are currently absent from it. These recommended terms can help to improve the completeness and clarity of the RS.

4.1 Parsing RS using NLP

To recommend relevant terms that may be missing from a given RS, we first apply to the RS a reduced version of the NLP pipeline presented in Chapter 2.1.Figure 4.2 shows the reduced pipeline. Once the document is fully annotated, the document serves as the input for the MLM task performed by BERT.

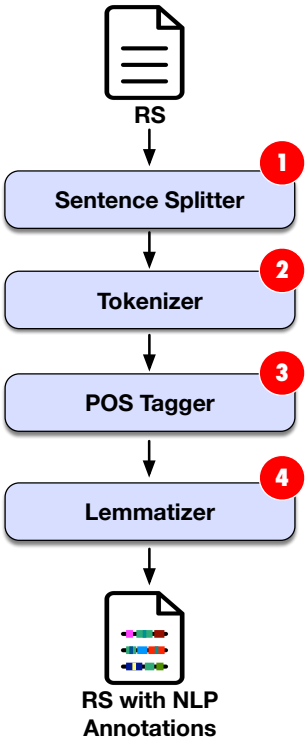

Figure 4.2: NLP Pipleline used in our Approach.

4.2 Obtaining Predictions from BERT

The core components of requirements are *nouns* and *verbs*, which convey the primary meaning of a statement [7, 37]. Our methodology involves iterating through each sentence in the annotated RS derived from Step 1. We selectively mask one noun or verb at a time by identifying its POS tag, allowing us to create a sentence with a single concealed word. Consider the sentence: "The system shall generate reports on inventory levels, product movement, and sales history." The POS tags for each word in the given

sentence are as follows: 'The' (DT); 'system' (NN); 'shall' (MD); 'generate' (VB); 'reports' (NNS); 'on' (IN); 'inventory' (NN); 'levels' (NNS); ',' (COMMA); 'product' (NN); 'movement' (NN); ',' (COMMA); 'and' (CC); 'sales' (NNS); 'history' (NN); '.' (PERIOD). We generate predictions by iteratively masking nouns and verbs as follows: "The system shall [MASK] reports on inventory levels, product movement, and sales history", "The system shall generate [MASK] on inventory levels, product movement, and sales history", and so on.

This modified sentence is inputted into BERT, which generates a configurable number of predictions for the masked word. For example, in our illustration shown in Fig.1.1, we use BERT to generate five predictions per masked word. Based on our empirical evaluation discussed in Chapter 5, we recommend using 15 predictions per masked word. BERT generates a probability score for each prediction, indicating its level of confidence. We retain the probability scores for use in Step 5 of our approach. Although we experimented with increasing the number of common words from 250, it had a minimal impact on recall as these words were infrequently predicted.

4.3 Removing Obviously Unuseful Predictions

In order to improve the accuracy of our predictive model and make it more efficient, we implement a strategy to eliminate irrelevant predictions that provide little or no additional information. The first criterion we use for filtering involves discarding predictions that are

already present in the RS. Consider the following sentence: "The system shall provide a programmable interface to support system integration." If the word 'integration' is masked and BERT correctly predicts the term, then it is eliminated from the prediction list. These predictions do not offer any new insights and therefore do not add value to the output. In addition to this, we have identified two further categories of predictions that are unlikely to contribute meaningfully to the final output. The first of these categories consists of the top 250 most commonly used words in the English language. These words are generic and their inclusion would not provide any useful information. The second category of predictions that we discard is a combination of vague words and stop words that we have compiled from two separate sources: requirements compiled by Berry et al. [13], and Arora et al.'s [5] [6]. By filtering out these unhelpful terms, we can effectively refine our list of predictions and eliminate any irrelevant terms. The end result of this step is a more precise and focused list of predictions that is cleared of unhelpful terms. By implementing these filtering criteria, we can ensure that our predictive model is optimized for accuracy and relevance to the specific domain of the RS.

4.4 Generating Domain-specific Corpus for RS

In Step 3 of our approach, we utilize WikiDoMiner (introduced in Chapter 2.5) to automatically extract a domain-specific corpus for an input RS [25]. For example, if an aerospace engineering RS is fed into WikiDominer, the resulting corpus may contain articles on air-

crafts, aviation, and fluid dynamics. Recall that as the depth parameter of WikiDominer increases, the corpus grows larger, encompassing more sub-categories of Wikipedia articles at each level. However, in this thesis, we limit our search to direct article matches only (i.e., $depth = 0$). This decision has two distinct advantages. First, by limiting the depth to zero, we significantly reduce the the volume of text being processed, enabling faster corpus generation. Second, focusing on direct article matches helps scope the expansion of terminology to the content that is immediately relevant to the domain of the input RS. During our exploratory investigation, we discovered that increasing the depth value results in a considerable expansion of the corpus size. However, a larger corpus size is not only more expensive to compute, but also would result in diluting domain-specificity. Our decision to use a depth value of zero strikes a balance between corpus size and domain coverage, yielding better-suited corpora for our purposes. Step 5 utilizes the domain-specific corpus to compute features, which are subsequently employed for filtering non-relevant predictions.

4.5 Building Feature Matrix for Filtering

In Step 5, we generate a feature vector for each prediction from Step 3. These feature vectors serve as input for a ML-based classifier, which determines whether the prediction is 'relevant' or 'non-relevant' to the input RS. Our feature design is based on the principle of keeping the features generic and normalized. This is essential because we do not want the features to rely on any specific domain or terminology. By keeping them generic, we

ensure that our approach can be applied to various domains, whereas normalization enables us to combine labelled data from multiple sources to train ML models that can be used for unseen data.

Features 10 and 11 are based on quantile bucketing. To categorize the frequency of predictions into discrete intervals or 'buckets', we can divide the terms into equal-sized groups based on percentiles. Suppose we have a bag of 1000 predictions generated by BERT, and wish to create 10 quantile buckets based on the frequency of these predictions. To accomplish this, we first count the number of occurrences of each predicted word and sort them in descending order according to their frequency. We then divide the words into 10 equally-sized groups based on their rank to create the quantile buckets. We assign the most frequently predicted words to bucket 0, and the least frequently predicted words to bucket 9.

Features 12 and 13 are based on Term Frequency-Inverse Document Frequency (TF-IDF) which is a technique used to measure the importance of words within a collection of documents (i.e., corpus). It consists of two components: Term Frequency (TF) and Inverse Document Frequency (IDF). TF represents how often a term occurs in a document, while IDF measures the significance of the term across the entire document collection. For example, if we have a dataset containing three articles, and we want to rank the importance of words within these articles using TF-IDF, consider the following collection of documents: Document 1: "The software shall allow users to upload files." Document 2: "The system shall store uploaded files securely." Document 3: "Administrators shall be able to manage

Table 4.1: Features for Learning Relevance and Non-relevance of Predictions Made by BERT.

ID	Type (T), Definition (D) and Intuition (I)
F1	**(T)** Nominal **(D)** POS tag of the masked word (noun or verb). **(I)** This feature is helpful if nouns and verbs happen to influence relevance in different ways.
F2	**(T)** Nominal **(D)** POS tag of the prediction; this is obtained by replacing the masked word with the predicted word and running the NLP pipeline on the resulting sentence. **(I)** The intuition is similar to F1, except that predictions are not necessarily nouns or verbs and can, e.g., be adjectives or adverbs.
F3	**(T)** Nominal (Boolean) **(D)** True if F1 and F2 match; otherwise, False. **(I)** A mismatch between F1 and F2 could be an indication that the prediction is non-relevant.
F4	**(T)** Numeric **(D)** Length (in characters) of the masked word. **(I)** Words that are too short may give little information. As such, predictions resulting from masking short words could be non-relevant.
F5	**(T)** Numeric **(D)** Length (in characters) of the prediction. **(I)** Predictions that are too short could be non-relevant.
F6	**(T)** Numeric **(D)** min (F4, F5)/ max (F4, F5). **(I)** A small ratio (i.e., a large difference in length between the prediction and the masked word) could indicate non-relevance.
F7	**(T)** Numeric **(D)** The confidence score that BERT provides alongside the prediction. **(I)** A prediction with a high confidence score could have an increased likelihood of being relevant.
F8	**(T)** Numeric **(D)** Levenshtein distance between the prediction and the masked word. **(I)** A small Levenshtein distance between the prediction and the masked word could indicate relevance.
F9	**(T)** Numeric **(D)** Semantic similarity computed as cosine similarity over word embeddings. **(I)** A prediction that is close in meaning to the masked word could have a higher likelihood of being relevant.
F10*	**(T)** Ordinal **(D)** A value between zero and nine, indicating how frequently the prediction (in lemmatized form) appears across *all BERT-generated predictions* over a given RS. **(I)** A smaller value could indicate a higher likelihood of relevance.
F11* †	**(T)** Ordinal **(D)** A value between zero and nine, indicating how frequently the prediction (in lemmatized form) appears in the *domain-specific corpus*. **(I)** A smaller value could indicate a higher likelihood of relevance.
F12† ‡	**(T)** Numeric **(D)** Average TF-IDF rank of the prediction across all articles in the domain-specific corpus. **(I)** A higher rank could indicate a higher likelihood of relevance.
F13† ‡	**(T)** Numeric **(D)** Maximum TF-IDF rank of the prediction across all articles in the domain-specific corpus. **(I)** Same intuition as that for F12.

*Zero is most frequent (top ten percentile) and nine is least frequent (bottom ten percentile). †Feature uses domain-specific corpus. ‡TF-IDF values are normalized by Euclidean norm.

users and their files."

In this example, the term 'files' has a low TF-IDF value since it appears in every document within the corpus. This low value indicates that it is not particularly important within the corpus due to its commonness. However, other terms like 'upload', 'store', and 'manage' might have higher TF-IDF values, indicating greater importance in the context of the corpus. The computation of TF-IDF is simple and well known. To make our example self-contained, the steps for the calculations are discussed below:

Calculate TF: For each article, count the number of times each word appears and divide it by the total number of words in that article. This results in a normalized term frequency for each word within the article.

Calculate IDF: For each word, calculate the logarithm of the ratio of the total number of articles in the dataset to the number of articles containing that word. This gives higher importance to words that appear less frequently across the corpus.

Calculate TF-IDF: Multiply TF by IDF for each word within each article. This results in a TF-IDF score that represents the importance of each word within a specific article and across the corpus. That is, $\text{tf} - \text{idf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D)$. Where: t is a term or word; d is a document containing the term t; D is a corpus or collection of documents; $\text{tf}(t, d)$ is the term frequency of term t in document d; and $\text{idf}(t, D)$ is the inverse document frequency of term t in the corpus D.

This step of our approach allows us to generate a feature matrix, where each row

represents a prediction from Step 3, and each column corresponds to a feature described in Table 4.1. The resulting feature matrix provides valuable insights into the relevance of each prediction to the input RS. This approach allows us to filter out non-relevant predictions effectively. Additionally, by using ML-based classifiers, we can improve the accuracy of the filtering process.

4.6 Filtering Noise from Predictions

In Step 5 of our approach, we implement a pre-trained ML-based filter to minimize the noise in the predictions obtained from Step 3. Noise can have a impact on the accuracy of predicted words as it significantly contributes to the number of unuseful predictions generated by BERT. Filtering is, therefore, essential for our classification task because it helps to reduce irrelevant predictions. To determine the most suitable ML algorithm for this task, we carry out empirical experiments, as detailed in the RQ2 section of Chapter 5. After identifying the best algorithm, we train it on the development and training portions of our dataset, denoted as P_1 in Table 5.3, as discussed in Chapter 5.

The generic and normalized features described in Table 4.1 allow us to apply the resulting ML model to unseen documents without the need for retraining, as demonstrated in the RQ3 section of Chapter 5. This ability is crucial in ensuring that our approach remains adaptable and efficient when dealing with new data. Following the classification of predictions using the pre-trained model, we proceed to filter out all predictions designated

as ‘non-relevant’. As a result, the output of this step comprises a list of BERT predictions that our filter deems ‘relevant’. Duplicates are naturally removed as they are redundant and does not affect the accuracy of our results. We execute phases one through five of our approach on each dataset in P_1. We then merge the resulting feature matrices into a single, comprehensive training set for the classifier. This process ensures that the ML-based filter is trained on a diverse range of data, allowing it to effectively filter out non-relevant predictions and yield more accurate results.

Chapter 5

Evaluation

In this Chapter, we empirically evaluate our approach. During the process, we also construct the pre-trained ML model required by Step 6 of our approach (Fig 4.1).

5.1 Implementation and Availability

Figure 5.1 provides an overview of the various technologies used in the implementation of our approach. The approach implemented in this thesis is written in Python, primarily due to the numerous third-party libraries available for machine learning tasks. The NLP pipeline used in this thesis is implemented using the latest version of SpaCy, version 3.2.2. SpaCy is a Python library used for natural language processing tasks such as tokenization, POS tagging, and dependency parsing. It provides fast and efficient implementations of

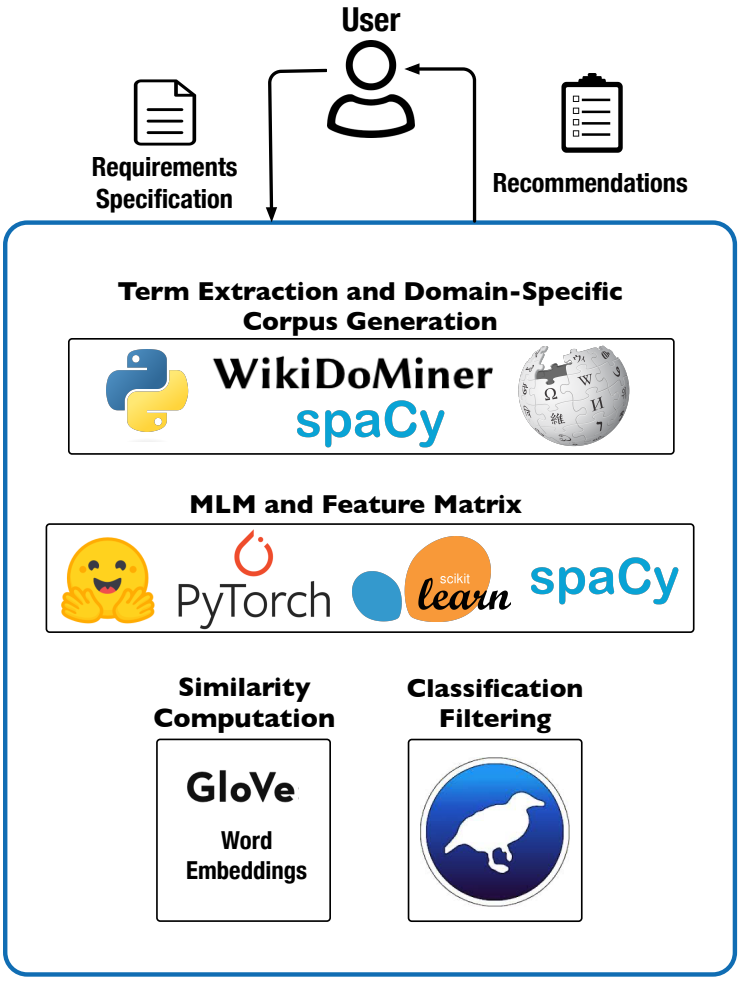

Figure 5.1: Tool Overview.

NLP tasks, making it suitable for processing large datasets. To extract word embeddings, we utilized the GloVe algorithm [49], which is an unsupervised learning algorithm for obtaining vector representations of words. Although there are several effective variants of BERT available (see Section 2.2), we choose to use the original version of BERT. The primary reason for this decision is the high computational cost associated with training and

evaluating BERT and variants thereof. Systematically replicating our experiments with multiple variants would be impractical within the scope of this thesis. BERT is already a computationally expensive model, requiring a significant amount of computing power. Each variant of BERT has its own specific architecture and hyperparameters, which can significantly increase the computational cost of training and evaluating the model. Given these constraints, we opt to use the original version of BERT as it has been extensively tested and validated in the literature. While the variants may offer some advantages in specific contexts or for specific tasks, we feel that the trade-off between increased computational costs and potential gains in performance does not justify the additional experimentation required. For obtaining BERT's masked language model predictions, we used the Transformers library by Hugging Face (https://huggingface.co/) version 4.16.2, which operates within PyTorch version 1.10.2+cu113. The Transformers library provides an easy-to-use interface for working with BERT models, enabling us to generate high-quality predictions for missing terminology in the input RS. Our ML-based filters are implemented using WEKA 3-8-5 [59], which is a machine learning software used for data mining and knowledge discovery. We utilized standard cosine similarity calculations (over word embeddings) and Levenshtein distance to implement the ML features listed in Table 4.1. The TFIDF-based features in the table (F12-13) are calculated using the TfidfVectorizer from scikit-learn version 1.0.2. We utilize WikiDoMiner in corpus generation for our TFIDF-based features. Additionally, WordNet (https://wordnet.princeton.edu/) was utilized for discovering synonyms in Baseline 3. The implementation of the approach is concise,

comprising only 480 lines of code. In addition, the implementation of our experimental procedure are 151 and 159 lines of code for RQ2 and RQ4 respectively. All of the implementation and evaluation artifacts are publicly available, ensuring reproducibility and transparency of the evaluation process [42].

5.2 Dataset

Our evaluation of this thesis is based on the PURE dataset [30]. The PURE (Public Requirements Dataset) is a collection of 79 publicly available software requirements documents, containing over 34,000 sentences, that was created to facilitate research on natural-language requirements. Since its release, the PURE dataset has become a widely used benchmark in a variety of RE tasks such as ambiguity detection and completeness checking. We select 40 documents from the PURE dataset to strike a balance between computation effort and dataset size. We want to establish a dataset that is large enough for statistical significance testing and training machine learning-based filters, while also mitigating the effects of random variation. After careful consideration, we find 40 documents to be a suitable size for our evaluation. These 40 documents represent 15 domains and are selected after a manual cleanup process. We remove unnecessary elements such as table of contents, headers, and section markers to ensure that the documents are uniform and consistent. Establishing a dataset that is diverse and representative allows us to be more confident in the generalizability of the proposed approach. We list these documents

in Table 5.3, along with domain information and summary statistics for each document. We partition the 40 documents into two non-overlapping subsets, denoted as P_1 and P_2. During approach development and tuning, P_1 is used to answer RQ1-RQ3, while P_2 is used to answer RQ4-RQ5. Our procedure for assigning documents to P_1 or P_2 is as follows: We first randomly select one document per domain and put it into P_2; this is to maximize domain representation in RQ3. From the rest, we randomly select 20 documents for inclusion in P_1, while attempting to have P_1 represent half of the data in terms of token count. Any remaining document after this process is assigned to P_2, thus giving us 20 documents in P_2 as well. Table 5.3 provides domain information and summary statistics for documents in P_1 and P_2 after cleanup.

5.3 Metrics

We define separate metrics for measuring (1) the quality of term predictions and (2) the performance of filtering. The first set of metrics is used in RQ1, RQ2, RQ4, and RQ5 and the second set is used in RQ3 and RQ4.

To clearly define our metrics, it is necessary to introduce some notation that we will use throughout our evaluation. Firstly, we employ a function $\mathsf{Lem} : \mathsf{bag} \rightarrow \mathsf{bag}$, which lemmatizes every element in a bag of words, effectively simplifying and standardizing the words in the bag. We use a function $\mathsf{U} : \mathsf{bag} \rightarrow \mathsf{set}$, which removes any duplicate words from a bag and returns a set. We also define a set C, which contains common words and

Table 5.1: Our Dataset (Subset of PURE [30]).

Domain	Document	Summary	Dev & Training	Testing	# of Sentences	# of Tokens
Security	sprat	Privacy and security for web-based systems		x	424	6284
	cctns	Broad crime investigation area		x	258	5458
	dii	XML services of the Defence Information Infrastructure		x	300	4526
Finance	gamma	GAMMA-J's web store	x		555	6510
	jse	Issuing of money market instruments ISIN codes by the JSE	x		468	6732
	e-procurement	eProcurement interoperability in the UK public sector		x	458	7277
Administration	tachonet	Issuing driver's licences	x		518	8681
	nasa x38	Scheduling services, communication services, time services, etc. for NASA's X-38 Crew Return Vehicle	x		1175	19132
	nenios	Childcare centre management	x		135	2428
	libra	Scheduler for queuing and resource managing	x		229	4476
	inventory	Web interface for accessing and managing integrated inventory		x	267	2513
Astronomy	evla back	Real-time astronomical data processing of the EVLA project	x		296	5236
	gemini	Control and data acquisition systems of Gemini telescopes	x		1458	24279
	esa	Data processing algorithms and support for APAF's Mars Express mission		x	154	2627
	telescope	Swift X-Ray Telescope, control processor and flight software		x	349	5996

Table 5.2: Our Dataset (cont. part 2).

Domain	Document	Summary	Dev & Training	Testing	# of Sentences	# of Tokens
Energy	pnnl	Detection of faults in HVAC system components	x		496	8985
	themas	Technical description for The Energy Management System (THEMAS)		x	340	4503
	elsfork	Supporting functions for remote operation and supervision of wind power plants		x	455	7131
Communications	philips	Instant messanger application	x		74	893
	ctc network	Center-to-Center Communications (C2C) project	x		193	2534
	agentmom	Broadcasting, multicasting, secured communication to agentMom in Multi-Agent Systems		x	108	1523
	tcs	Provide military with services to communicate with tactical unmanned aerial vehicles		x	1748	31959
Hardware Design	beyond	Domain-oriented approach for support of electronic systems	x		545	10391
	evlacorr	WIDAR correlator hardware and EVLA monitor & control system		x	180	3836
Medicine	microcare	Resource to address selected health problems		x	393	6250
Databases	npac	Database of information required to effect the porting of telephone numbers		x	3754	73326

Table 5.3: Our Dataset (cont. part 3).

Domain	Document	Summary	Dev & Training	Testing	# of Sentences	# of Tokens
Games	spacefractions	Web-based game to help improve fraction-solving skills for sixth-grade students	x		168	2796
	multi-mahjong	Single and multi-user mahjong game	x		374	5687
	qheadache	Computerized game that displays an interface used to solve a specific headache		x	151	1749
Art	colorcast	Art paint numbering scheme		x	224	3625
Weather	claruslow	Surface transportation-based weather observations	x		903	15372
	gridbgc	Generating gridded surface weather datasets from observation data records	x		505	4871
	clarushigh	Surface transportation-based weather observations		x	520	8869
Legal	iijis	Integrated Justice Information System (IJIS) for more effective and efficient administration of justice		x	266	3853
Transport	rlcs	Open and close the reversible lanes for peak traffic hours and special events		x	587	10808
UX/Visualization	watcomgui	Open Watcom low level GUI library research	x		652	7786
	sce api	Standard Co-Emulation API: Modeling Interface for verification requirements	x		1049	21543
	hats	GUI for the High Assurance Transformation System (HATS)	x		1320	19739
	watcom	Code support to the Open Watcom compiler	x		599	7481
	grid 3D	Visualisation of biological data		x	130	1813
# of Sentences	-	-	11712	12694	-	-
# of Tokens	-	-	185552	193926	-	-

stopwords as detailed in Step 3 of Chapter 4. Given a document p, we divide it into two separate partitions: a disclosed partition h_1 and a withheld partition h_2. The terminology contained in h_1 is defined as set $X = \mathsf{U}(\mathsf{Lem}(h_1))$, while the terminological content of h_2 is defined as set $Y = \mathsf{U}(\mathsf{Lem}(h_2))$. We define novel terminology we aim to predict as $N = (Y - X) - C$. This set contains all terms that appear exclusively in h_2 and are not considered common words or stopwords. In order to predict the novel terminology in set N, we utilize a bag V, which is the output of Step 3 in Figure 4.1 when applied specifically to h_1. It is important to note that bag V is already stripped of any terminology that appears in h_1, as well as all common words and stopwords. Our ultimate objective is to employ BERT to predict as many terms in set N as possible, thereby demonstrating the model's ability to effectively hint at potential incompleteness in the RS.

5.3.1 Quality of Term Predictions

The quality of term predictions is determined by the accuracy and coverage of the set D, which is composed of the (duplicate-free) lemmatized predictions that have the potential to suggest new terminologies in the withheld half of a given document. Assessing the quality of D allows us to evaluate BERT's capability for predicting novel terminologies.

Accuracy is an important metric that quantifies the proportion of terms in D that correspond to some term in the set N. The set N represents the novel terminology present in the withheld half of the document. By calculating the ratio of accurate predictions in

D to the total number of terms in D, we can measure how many of the predicted terms appear in the withheld portion. In order to determine whether a term in D matches a term in N, we employ word embeddings to calculate the cosine similarity between the two terms. To establish our ground truth, we consider a term to be a match if it achieves an 85% cosine similarity threshold, calculated over non-contextualized word embeddings. Although there are other methods available, such as using contextualized word embeddings or lemma matching, we opt for non-contextualized word embeddings since contextualized embeddings generated by BERT would require computing a large number of pairwise comparisons, which would be a costly and not scale well. On the other hand, using lemmas for matching predictions would be too rigid, and this method may fail to match synonyms or encounter difficulties with homonyms. Given these limitations, we utilize GLoVE's non-contextualized word embeddings as a proxy for evaluation. As we do not have access to domain experts or the resources to employ contextualized embeddings for this thesis, non-contextualized word embeddings serve as an effective solution for our purposes. While this approach may not be ideal, it is the best available option for evaluating the performance of our model under the given constraints.

The 85% cosine similarity threshold ensures that the matching process is not overly strict and allows for variations in word forms while still enforcing a meaningful semantic similarity between the terms. To summarize the accuracy calculation, $Accuracy = |t \in D \mid t \text{ matches some } t' \in N|/|D|$. A term t is considered to match another term t' if its word embeddings meet or exceed the specified cosine similarity threshold. It is essential

to measure not only the accuracy of the predicted terms but also the extent to which the model is capable of predicting a broad range of novel terms.

Coverage is a metric that calculates the proportion of terms in N that have a matching term in D. Coverage is calculated as follows: $\textit{Coverage} = |\{t \in N \mid t \text{ matches some } t' \in D\}|/|N|$. The intuition for Accuracy and Coverage is the same as that for the standard Precision and Recall metrics, respectively. Nevertheless, since our matching is inexact and based on a similarity threshold, it is possible for more than one term in D to match an individual term in N. Coverage, as we define it, excludes multiple matches, providing a measure of how much of the novel terminology in the withheld half is hinted at by BERT. Overall, these metrics provide an assessment of the quality of BERT's predictions through the MLM task, and they help to evaluate its effectiveness in identifying and suggesting new terms. The accuracy and coverage metrics provide complementary information about BERT's performance, and both are necessary to gain a complete understanding of the model's effectiveness in predicting novel terminologies.

5.3.2 Quality of Filtering

Our filter is a binary classifier that distinguishes between relevant and non-relevant terms in the outputs generated by BERT. The filter's purpose is to reduce the number of false positives, i.e., irrelevant terms that BERT has predicted as relevant, so that only the most relevant terms are retained for further analysis. To measure the performance of

our filter, we use three standard metrics: Classification Accuracy, Precision, and Recall. These metrics provide a comprehensive evaluation of the filter's effectiveness in correctly identifying relevant terms and excluding non-relevant terms.

We define true positive (TP), false positive (FP), true negative (TN), and false negative (FN) as follows: A TP is a classification of 'relevant' for a term that has a match in the set N, which represents the novel terminology in the withheld half of the document. An FP is a classification of 'relevant' for a term that does not have a match in N. A TN is a classification of 'non-relevant' for a term that does not have a match in N. A FN is a classification of 'non-relevant' for a term that does have a match in N.

Classification Accuracy is calculated as $(TP + TN)/(TP + TN + FP + FN)$. This metric measures the proportion of correct classifications made by the filter, taking into account both correct and incorrect classifications, making it a good overall indicator of the filter's performance. *Precision* is calculated as $TP/(TP + FP)$, which measures the proportion of relevant terms among the terms classified as relevant. Precision indicates the filter's effectiveness in avoiding false positives, ensuring that only truly relevant terms are classified as such. *Recall* is calculated as $TP/(TP + FN)$, which measures the proportion of relevant terms among all the relevant terms present in the set N. In other words, recall demonstrates how well the filter captures all the relevant terms, providing insights into its sensitivity and the extent to which it can identify important terminologies.

5.4 Research Questions

In this section we outline the procedures and results of our research questions. We reformulate the general research questions from the introduction and instantiate them around BERT.

5.4.1 RQ1. How accurately can BERT predict relevant but missing terminology for an input RS?

Research Question 1 (RQ1) aims to determine BERT's accuracy in predicting relevant but missing terminology in an input RS. This question establishes whether BERT can be a viable solution for detecting incompleteness in natural-language requirements. The number of predictions generated by BERT per mask is a configurable parameter, and RQ1 identifies the optimal value offering the best balance for producing useful recommendations.

This investigation will focus on the optimal number of predictions per mask between the range of 5 and 20. The optimal number of predictions is essential in ensuring that our approach can generate accurate recommendations for missing terminology in the RS. If the number of predictions is too low, the approach may miss out on relevant recommendations, reducing the overall accuracy of the approach. Conversely, if the number of predictions is too high, the approach may generate too many recommendations, making it difficult for users to identify the most relevant recommendations for their specific needs.

5.4.1.1 Experimental Procedure for RQ1

In this experiment, we aim to answer RQ1 and find a suitable balance between number of predictions made by BERT and the quality of these predictions. The following approach is in line with the arguments presented in Chapter 1. For every document $p \in P_1$, we randomly partition the set of sentences in p into two subsets of (almost) equal sizes. One of these subsets is *disclosed* to BERT, while the other is *withheld*. Injecting incompleteness in p by withholding allows us to evaluate BERT's effectiveness in predicting relevant terms to an input RS. We apply Steps 1, 2 and 3 of our approach, as shown in Fig. 4.1, to the disclosed subset, treating it as if it were the entire input document. Note that the corpus extraction step of our approach exclusively uses the disclosed portion without any knowledge of the withheld portion of our dataset. This ensures that the performance of the corpus-based features (F12–F13) in identifying relevant terms from the disclosed subset is evaluated independently of the withheld subset.

To evaluate the performance of BERT in identifying relevant terms from the disclosed subset, we vary the number of predictions that BERT generates per mask. We run Step 2 of our approach with four different numbers of predictions per mask: 5, 10, 15, and 20. For every document $p \in P_1$, we compute two metrics, Accuracy and Coverage, which are defined in Section 5.3. As the number of predictions per mask increases from 5 to 20, BERT generates more predictions which reveal terms that are relevant to the withheld subset. However, beyond 15 predictions per mask, the benefits of increasing the number

of predictions per mask diminish. As we will see in Chapter **??**, this additional coverage comes at the cost of introducing noise, which occurs when BERT generates (far too many) non-relevant predictions for every useful prediction.

The goal of RQ1 is to find a suitable compromise in terms of the number of predictions made by BERT to achieve good coverage while ensuring that the rate of introducing noise is not higher than the rate of useful predictions. We analyze the performance of BERT for different numbers of predictions per mask to determine the optimal number of predictions for finding relevant matches in the withheld portion. To verify that the trends we observe in our experiments are not due to random variation, we shuffle the documents p differently for each number of predictions per mask (i.e., the disclosed and withheld subsets for each document p are different random subsets when experimenting with 5 predictions per mask than when experimenting with 10 predictions per mask, and so on). This approach is designed to eliminate any potential biases in our results that may be caused by a specific shuffle of documents that we use. Based on our experiments and results, we find that the optimal number of predictions per mask is 15 which we use for the remainder of our analysis.

5.4.1.2 RQ1 Results

In order to determine the optimal number of predictions per mask for detecting omissions in an RS, we focus on BERT's performance with varying numbers of predictions generated for each masked token, ranging from 5 to 20 in increments of 5. Figures 5.2 and 5.3 provide box

plots for Accuracy and Coverage, respectively. We use 20 documents in P_1 as our dataset, with each datapoint representing one document. We perform statistical significance tests on the obtained metrics using the non-parametric pairwise Wilcoxon's rank sum test [16] and Vargha-Delaney's effect size [55]. The Vargha-Delaney (VD) statistical analysis is a non-parametric measure of the effect size in two-sample comparison tests. It takes into account both the differences in performance and the relative sizes of the datasets being compared. The VD statistic values ranges from 0 to 1, with 0.5 indicating no difference in performance between the two models, and values closer to 1 indicating a larger difference between the two models being compared. We present the results of these tests in Table 5.4, which compares the metrics across a different number of predictions per mask. Each column in the table compares Accuracy and Coverage across two levels of predictions per mask. For example, the *5 vs. 10* column compares the metrics for when BERT generates 5 predictions per mask versus when it generates 10.

For Accuracy, Fig. 5.2 shows that as the number of predictions per mask increases, there is a downward trend in Accuracy. The difference in Accuracy is statistically significant for all pairs of the form: *5 vs.* $y > 5$ (large effect size) and *10 vs.* $y > 10$ (medium to large effect size). In terms of Accuracy, 5 predictions per mask lead to statistically significantly better results as compared to the other four alternatives. Based on Table 5.4, the decline in Accuracy is statistically significant with each increase in the number of predictions, the exception being the increase from 15 to 20, where the decline is not statistically significant. We note that statistical non-significance could be due to the small population size (20

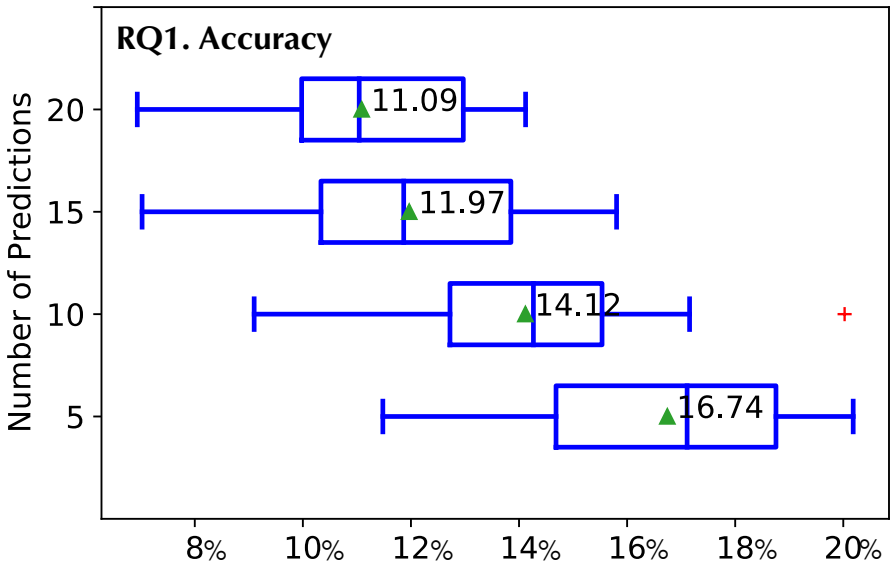

Figure 5.2: Accuracy for Different Numbers of Predictions per Mask.

datapoints for each level of predictions per mask).

For Coverage, Fig. 5.3 shows an upward but saturating trend. The difference in Coverage is statistically significant for pairs of the form:*5 vs.* $y > 5$ (large effect size). Results indicate an average gain of 3.2% in coverage as we move from 10 to 15 predictions, despite the difference not being statistically significant. Increasing from 15 to 20 yields no gain. The slight decrease is due to datasets utilized for 15 and 20 being different random shuffles of each document (this is theoretically possible because the experiments at a different number of predictions use different random shuffles of the RS in P_1). Therefore, we can conclude that five predictions per mask would be too few: all other levels are significantly better. Twenty would be too many, notably because of the lack of a significant difference

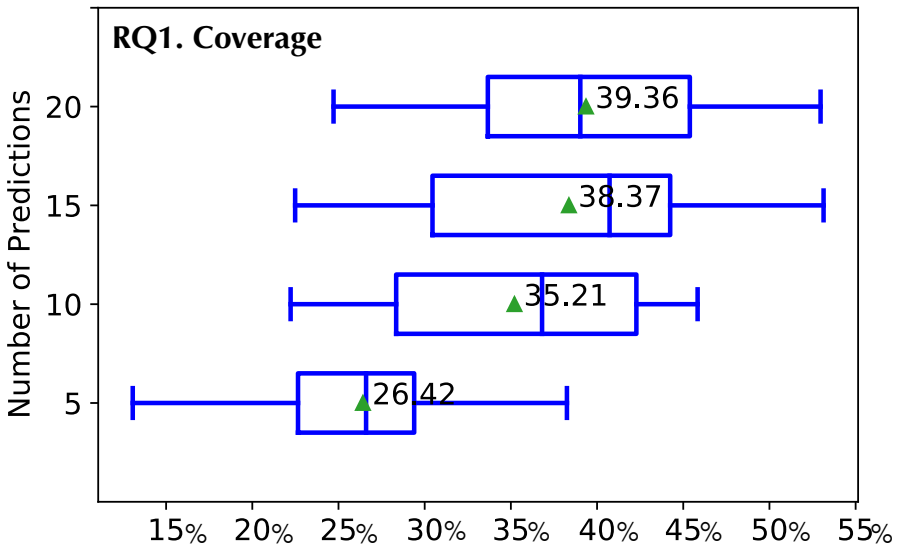

Figure 5.3: Coverage for Different Numbers of Predictions per Mask.

Table 5.4: Statistical Significance of RQ1 over Train Set.

		Accuracy	**Coverage**
5 vs. 10	p-value	0.005131	0.00148
	Â12	0.245 (L)	0.795(L)
5 vs. 15	p-value	4.909E-06	0.00005208
	Â12	0.1075 (L)	0.855(L)
5 vs. 20	p-value	2.317E-08	0.00002138
	Â12	0.045 (L)	0.87(L)
10 vs. 15	p-value	0.0143	0.2184
	Â12	0.725 (M)	0.385(S)
10 vs. 20	p-value	0.0002277	0.2084
	Â12	0.8275 (L)	0.3825(S)
15 vs. 20	p-value	0.2423	0.8831
	Â12	0.61 (S)	0.485 (N)
Effect size: Large (L), Medium (M), Small (S), Negligible (N)			

for Coverage in the *10 vs. 20* column of Table 5.4. The choice is thus between 10 and 15. We select 15 predictions per mask since this level yields an average increase of 3.2% in Coverage compared to 10 predictions per mask. This increase is not statistically significant. Nevertheless, the price to pay is an average decrease of $(14.12 - 11.97 =)$ 2.15% in Accuracy. Given the importance of Coverage, we deem 15 to be a better compromise than 10.

Overall, our findings suggest that when requirements omissions are simulated by withholding, having BERT make 15 predictions per mask is the best approach for detecting missing terminology. On average, BERT predicted terms that hinted at ≈4 out of 10 omissions (Coverage ≈38%). However, only approximately 1 in 8 predictions were relevant (Accuracy ≈12%).

5.4.2 RQ2. How does our approach compare to baselines?

Research Question 2 (RQ2) assesses the performance of predictions generated using our approach in comparison to baselines. Specifically, RQ2 will compare the results obtained from RQ1 with their corresponding baseline values to determine the advantage of utilizing BERT to make contextualized predictions prior to filtering.

To achieve this goal, we will compare the accuracy and coverage values obtained from our approach with those obtained from other established baseline metrics. BERT's ability to make contextualized predictions is a crucial advantage, as it allows our approach to make

predictions without utilizing an external source of knowledge. By comparing the results obtained from our approach with those obtained from other baselines, we can determine the extent to which this advantage translates into improved performance. Furthermore, we can identify scenarios where our approach may not perform as well as other established methods.

5.4.2.1 Experimental Procedure for RQ2

To evaluate the effectiveness of our approach, we need to compare it to existing baselines. However, since our approach relies on using a language model as the primary source of knowledge, we are unable to compare the results to other research in this field as they utilize techniques requiring external artifacts. To address this challenge, we define three common-sense baselines that do not use language models, and are therefore suitable for comparison.

The first baseline is a list of the 250-to-1000 most common words in the English language. This baseline serves as a straightforward benchmark in evaluating the effectiveness of context-based predictions generated by our approach. By comparing our approach's predictions to this baseline, we can evaluate BERT's capacity in predicting relevant terminology versus utilizing a list of generic words for detecting omissions. The second baseline uses TF-IDF values, which capture the relevance of a term to a particular document based on its frequency and rarity of appearance in the corpus. This baseline captures only the most relevant and commonly used terminology in the domain and determines if domain-

```
Baseline 1:
INITIALIZE novel_common_words to empty list
INITIALIZE all_common_words to list of 250-1000 common English words
FOR word in all_common_words
    IF word is NOT in disclosed_portion
        IF word is NOT in stop_word
            IF word in withheld_portion
                APPEND word to novel_common_words
            END IF
        END IF
    END IF
END FOR
RETURN novel_common_words
```

Figure 5.4: Baseline 1 Pseudocode

specific predictions outperform BERT's context-specific predictions in identifying relevant but missing terminology. The third baseline produces synonyms for words appearing in the disclosed half to provide an alternative means of predicting relevant terms that relies on semantic similarity. This baseline aims to evaluate the performance of using a simple lexical method compared to the more computationally expensive language model used in our approach. To make a direct comparison to the performance of EXPI, these baselines are applied to the training and development documents in disclosed portion before filtering. In EXPII, we compare the performance of our approach using the metrics defined in the previous experiment (i.e., Accuracy and Coverage from EXPI) against that of the three baselines. This allows us to assess how our approach fares against these straightforward baselines and determine if the complexity of our approach, which relies on BERT and ML-based filters, is justified.

```
Let T be a data frame of terms and their corresponding TF-IDF score
INITIALIZE novel_tfidf_terms to empty list
INITIALIZE tfidf as terms with top k TF-IDF score in T
FOR term in tfidf
    IF term is NOT in disclosed_portion
        IF term is NOT in stop_word
            IF word in withheld_portion
                APPEND term to novel_tfidf_terms
            END IF
        END IF
    END IF
END FOR
RETURN novel_tfidf_terms
```

Figure 5.5: Baseline 2 Pseudocode

```
INITIALIZE novel_synonyms to empty list
FOR term in disclosed_portion
    INITIALIZE lst_synonyms to synonyms of term
    FOR synonym in lst_synonyms:
        IF synonym is NOT in disclosed_portion
            IF synonym is NOT in stop_word
                IF word in withheld_portion
                    APPEND synonym to novel_synonyms
                END IF
            END IF
        END IF
    END FOR
END FOR
RETURN novel_synonyms
```

Figure 5.6: Baseline 3 Pseudocode

5.4.2.2 RQ2 Results

To gain a better understanding of the effectiveness of BERT-generated predictions in detecting missing terminology in requirements documents, we conduct an investigation where we compare the unfiltered results obtained from RQ1 against three different baseline models. The primary objective of this investigation is to determine which model provides the greatest Coverage for the withheld portion before any noise reduction, which is essential since RQ3 and RQ4 aim to reduce noise for greater Accuracy, leading to a decline in Coverage.

Our results, presented in Figure 5.9, shows that all three baseline models have significantly worse Coverage compared to the BERT-generated predictions. Recall from 5.4 Baseline 1 involves selecting the top 250-1000 most common words in the English language, while Baseline 2 uses TF-IDF values obtained from the domain-specific corpus to build our feature matrix. Baseline 3, on the other hand, uses synonyms for terms appearing in the disclosed portion to hint at words appearing in the withheld portion. Our investigation shows that these simple approaches are ineffective in predicting relevant terminology.

Baselines 1 and 2 have Accuracy values which do not have statistical significance compared to the unfiltered BERT output, and in fact, have a slightly better performance as seen in Figure 5.10, Baseline 3 performs significantly worse with a p-value of 2.20E-16 (as observed in Table 5.5). These findings suggest that the unfiltered BERT predictions are superior to the outputs produced by the baseline models. Our approach of masking

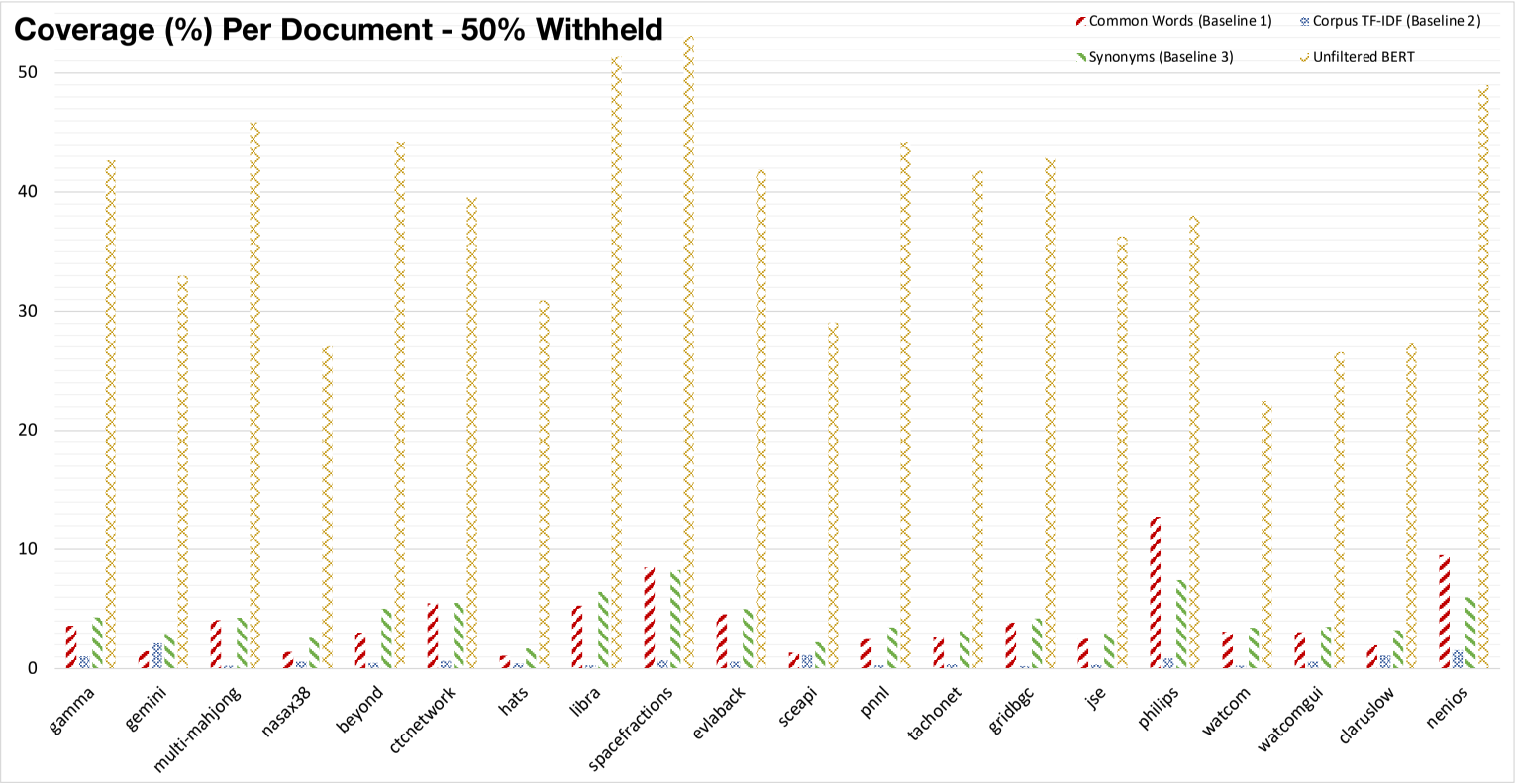

Figure 5.7: Comparing the Coverage of Our Approach Against Baselines for $p \in P_1$.

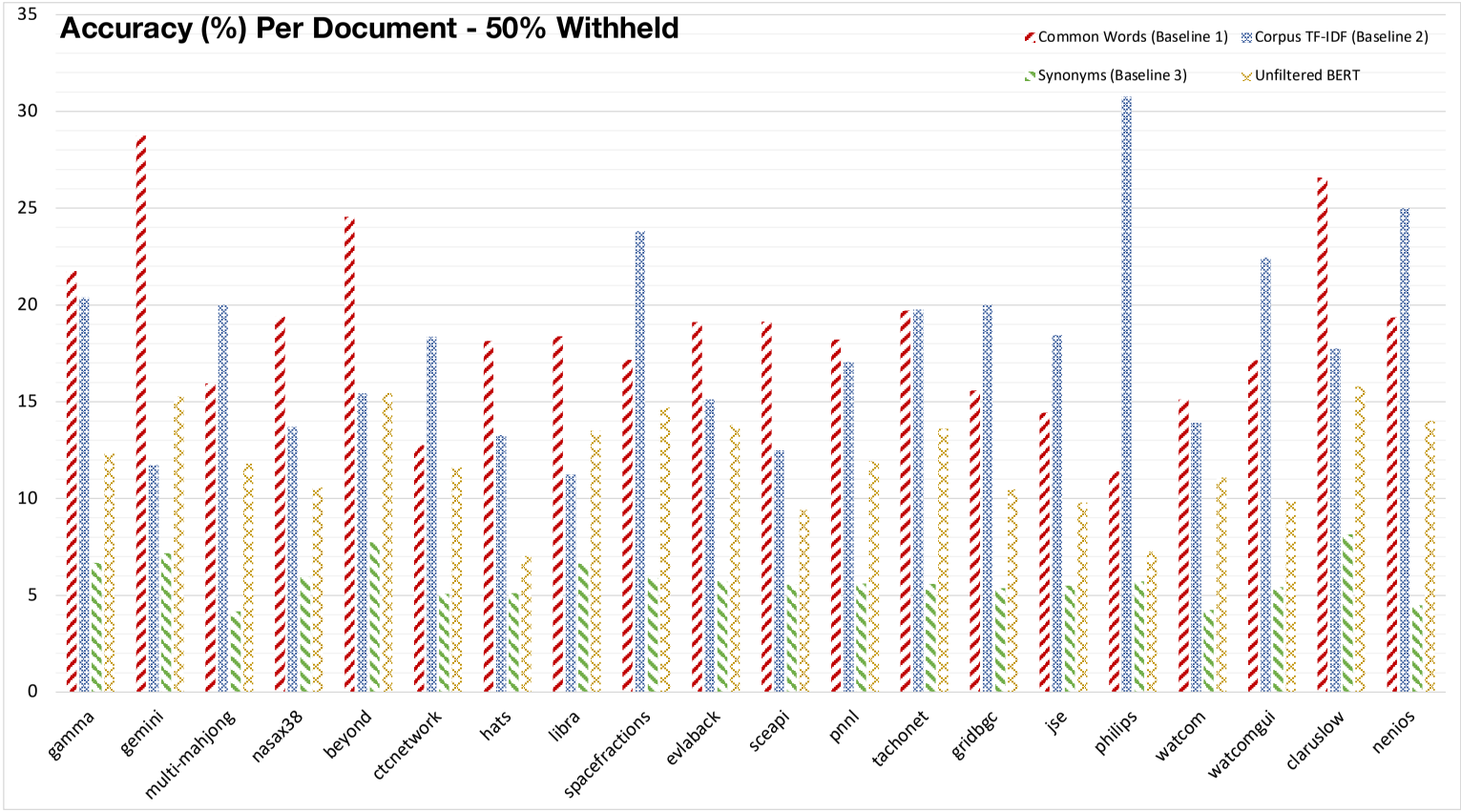

Figure 5.8: Comparing the Accuracy of Our Approach Against Baselines for $p \in P_1$.

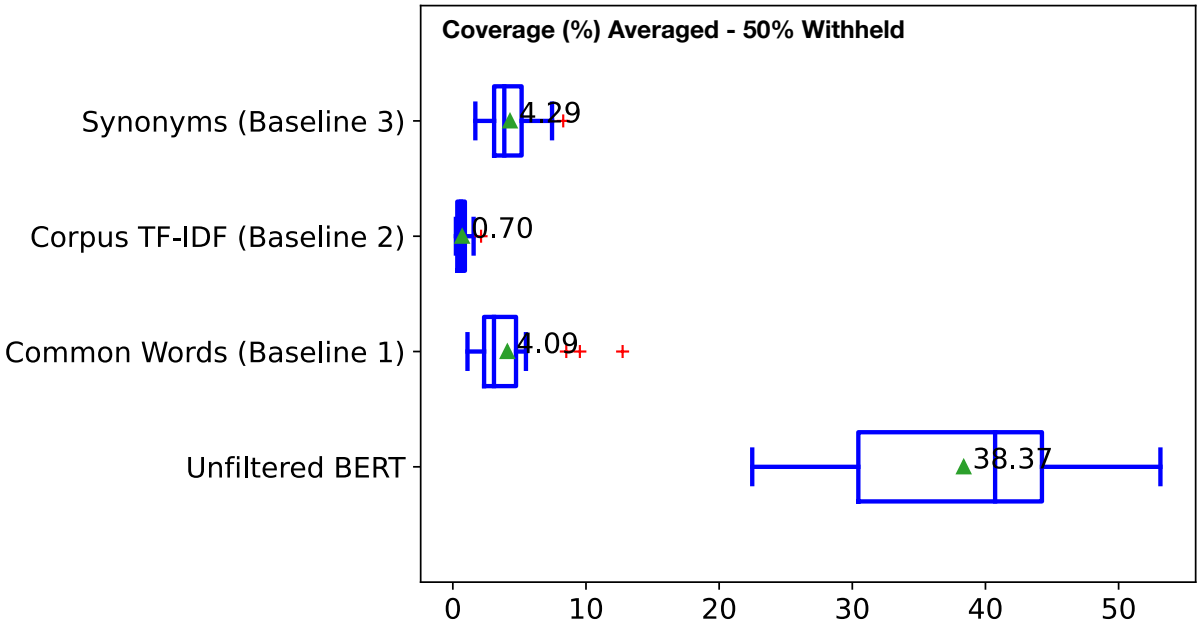

Figure 5.9: Baseline Coverages Averaged over Train Set (P_1).

		Accuracy	**Coverage**
Unfiltered vs. Baseline 1	p-value	5.57E-01	2.20E-16
	Â12	0.4759 (N)	0.0059 (L)
Unfiltered vs. Baseline 2	p-value	7.31E-02	2.20E-16
	Â12	0.4266 (N)	0.1175 (L)
Unfiltered vs. Baseline 3	p-value	0.2.2e-16	2.20E-16
	Â12	0.0835 (L)	0.033 (L)
Effect size: Large (L), Medium (M), Small (S), Negligible (N)			

Table 5.5: Statistical Significance of RQ2 Baselines over Train Set.

certain words in the input RS and predicting relevant context-based tokens results in more matches to the terminology in the withheld portion. We conclude that BERT's ability to generate prediction results with the highest Coverage of 38.37% (as seen in Figure 5.9) is

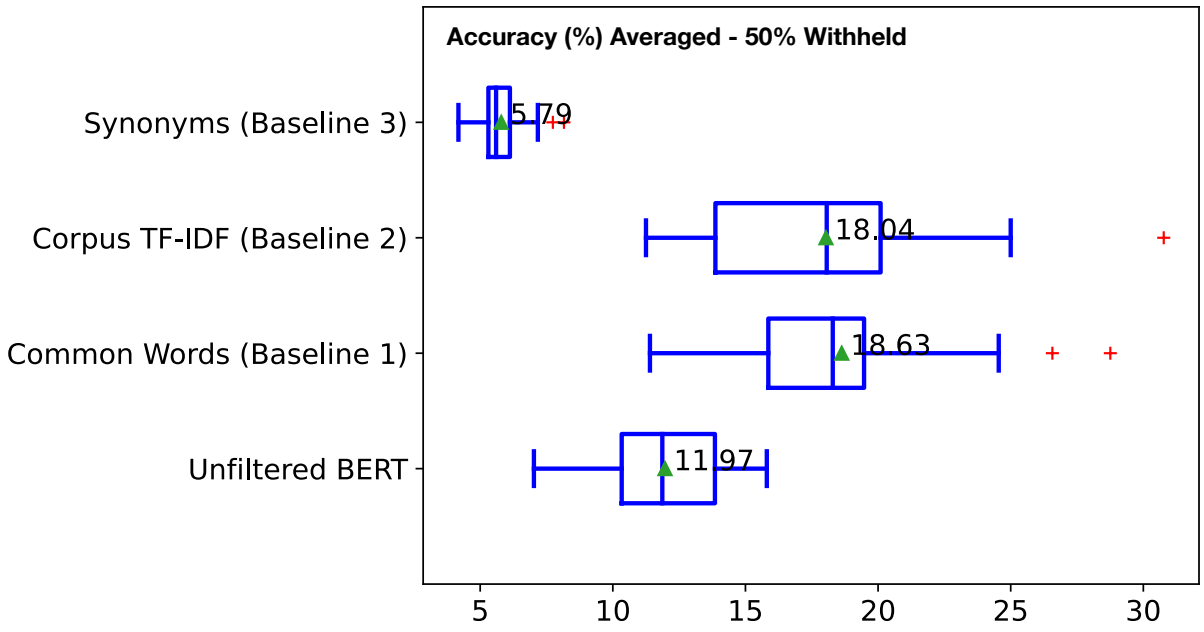

Figure 5.10: Baseline Accuracies Averaged over Train Set (P_1).

an improvement of over 5000% compared to Baseline 2. This significant improvement in Coverage justifies our approach taken in using the MLM task to generate predictions for potentially relevant terminology appearing in the withheld portion. Notably, BERT is able to outperform the baseline models even before any filtering or domain-specific fine-tuning was performed. This highlights BERT's strength in processing an input RS by capturing underlying language patterns to generate context-specific predictions. Our results suggest that BERT-generated predictions are highly effective in detecting missing terminology in requirements documents, even without any additional domain-specific knowledge. This finding has significant implications for the development of tools and techniques that can leverage BERT's capabilities for improving the accuracy and completeness of requirements.

5.4.3 RQ3. Which ML classification algorithm most accurately filters unuseful predictions made by BERT?

Research Question 3 (RQ3) seeks to establish the most accurate ML classification algorithm for filtering out unuseful predictions made by BERT. Although BERT generates useful recommendations, there is also a significant amount of noise present. RQ3 investigates different ML algorithms to filter this noise and explores the effect of data balancing and cost-sensitive learning to prevent over-filtering. The goal of RQ3 is to improve the accuracy of predictions generated through our approach, making it more effective in detecting incomplete requirements.

5.4.3.1 Experimental Procedure for RQ3

This experiment answers RQ3 and further constructs the training set for the ML classifier in Step 6 of our approach (Fig. 4.1). We recall the disclosed and withheld halves as defined in *EXPI*. For every document $p \in P_1$, we label the predictions made by BERT as 'relevant' or 'non-relevant' using the following procedure: Any prediction that matches some term in the withheld half is labelled 'relevant'. The criterion for deciding whether two terms match is a cosine similarity of 85% over GloVe word embeddings (introduced in Chapter 2). All other predictions are labelled 'non-relevant'. The threshold of 85% is a conservative one, letting only terms with the same lemma or terms with very high semantic similarity to be matched. Cosine similarity with this high threshold performs better than lexical matching.

For each prediction, a set of features is calculated as detailed in Step 5 of our approach. It is paramount to note that Step 4, which is a prerequisite to Step 5, *exclusively* uses the content of the disclosed half without any knowledge of the withheld half. The above process produces labelled data for each $p \in P_1$. We aggregate the labelled data from the documents in P_1 into a single *training set*. This is possible because our features (listed in Table 4.1) are generic and normalized.

The principle to create the training set (T) for ML-algorithms is by partitioning a textual RS into disclosed and withheld parts. T includes the labelled predictions of the disclosed half, aggregated from all p in P_1 and their features. As the majority of predictions are labelled 'non-relevant', we apply under-sampling on T to balance the number of instances of predictions. Then, the most influential features are selected to train the model. We consider only widely used and recommended ML algorithms for our filter. These ML classifiers attempt to increase Recall by preserving 'relevant' predictions while filtering as many 'non-relevant' predictions as possible to improve Precision. We then feed the training set to five ML algorithms: NN, DT, LR, RF and SVM to determine the performance of each algorithm. All algorithms are tuned with optimal hyperparameters that maximize classification accuracy over the training set. For tuning, we apply multisearch hyperparameter optimization using random search [8]. The basis for tuning and comparing algorithms is ten-fold cross-validation. We experiment with under-sampling the 'non-relevant' class with and without CSL; the motivation is reducing false negatives (i.e., relevant terms incorrectly classified as 'non-relevant').

We run the ten-fold cross-validation procedure above, once with under-sampling only and once with under-sampling combined with CSL. We under-sample the 'non-relevant' class to balance the number of 'non-relevant' and 'relevant' labels. The full training set exhibits an imbalance ratio of 18421:43575. For CSL, we assign double the cost (penalty) to false negatives compared to false positives (i.e., noise) to give more weight to recall than precision. We further assess the importance of our features using information gain (IG) [58]. In our context, IG measures how efficient a given feature is in discriminating 'non-relevant' from 'relevant' predictions. A higher IG value implies a higher discriminative power. In ten-fold cross-validation, a given dataset is randomly split into ten equal subsets. The following procedure is repeated ten times, with nine subsets used for training and the last subset reserved for evaluation. The ML algorithms that yield the best average Accuracy and Recall are 'SVM' and 'RF'.

5.4.3.2 RQ3 Results

Table 5.6 presents the results of ML-algorithm selection for filtering unuseful predictions using three options: (1) full training set with 61,996 datapoints, (2) under-sampled training set with 36,842 datapoints, and (3) under-sampled training set with CSL. For each option, we calculate Classification Accuracy, Precision, and Recall using ten-fold cross-validation. The best result for each metric is highlighted in bold in the table.

When one uses the full training set (*option 1*) or the under-sampled training set without CSL (*option 2*), RF turns out to be the best alternative. This may be justified as there are

Table 5.6: ML Algorithm Selection (Q3) with Tuned Hyperparameters.

		Full Training Set*	**Under-sampled†**	**Under-sampled + CSL‡**
Neural Network	Classification Accuracy (%)	81.1	78.9	77.6
	Precision (%)	69.5	77.6	72
	Recall (%)	65.1	81.2	90.4
Decision Tree	Classification Accuracy (%)	82.3	78.9	77.3
	Precision (%)	74.6	78.3	71.7
	Recall (%)	61.5	79.9	90.3
Logistic Regression	Classification Accuracy (%)	81.3	79	77.1
	Precision (%)	73.5	**79.1**	71.7
	Recall (%)	58.1	78.7	91.7
Random Forest	Classification Accuracy (%)	**84.1**	**80.3**	**79.1**
	Precision (%)	**76.4**	78.8	**74.5**
	Recall (%)	**67**	**83**	88.6
Support Vector Machine	Classification Accuracy (%)	81.4	79.2	76.8
	Precision (%)	74.1	77.9	70
	Recall (%)	57.4	81.7	**92.4**

Percision (%) = Precision of the "relevant" class, Recall (%) = Recall of the "relevant" class; all values are percentages.
**Strict filtering (option 1)*, †*Moderate filtering (option 2)*, ‡*Lenient filtering (option 3)*

more 'non-relevant' labels, which results in a larger number of TN causing higher Accuracy. When the under-sampled training set is combined with CSL (*option 3*), RF still has the best Accuracy and Precision. However, SVM presents a moderate advantage in terms of Recall. Since option 3 is meant at further improving the filter's Recall, we pick SVM as the best alternative for this particular option. To evaluate the quality of word predictions,

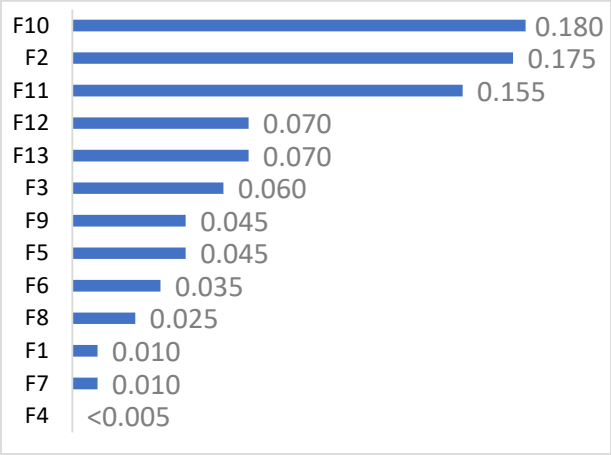

Figure 5.11: Feature Importance (Avg).

we prefer to choose classifiers with high Recall since it allows for more novel predictions. Compared to option 1, options 2 and 3 get progressively more lax by filtering.

Figure 5.11 illustrates the relative importance of the features used in our approach for unuseful prediction filtering. Specifically, the figure lists the features in descending order of their IG, which represents the amount of information that a feature provides for classification. The IG values are averaged across all three options considered in our study (options 1, 2, and 3), which correspond to different levels of strictness as specified above. We observe that the corpus-based features F12–F13, which are based on domain-specific terminology in the input text, are among the most influential features for all three options. This finding justifies the use of a domain-specific corpus extractor in our approach, as it allows us to leverage the unique characteristics of the requirements domain to improve the filtering of irrelevant predictions. Moreover, we note that some of the other features,

such as the masked token POS tag (F2) and the frequency of prediction (F10), also have relatively high IG values. These features capture the syntactic and semantic meaning of the input text and may indicate linguistic factors that are more likely to result in relevant predictions.

5.4.4 RQ4. How accurate are the recommendations generated by our approach over unseen documents?

Research Question 4 (RQ4) focuses on determining the accuracy of the recommendations generated by our approach over the testing portion of our dataset to answer RQ4. In RQ4, we integrate the optimal BERT configuration from RQ1 with the best-performing filter models from RQ3 and evaluate the accuracy and coverage of this combined approach over new data. The goal of RQ4 is to provide insights into the generalizability of our approach on previously unseen documents.

5.4.4.1 Experimental Procedure for RQ4

This experiment aims to answer RQ4 by applying our end-to-end approach to previously unseen requirements documents, specifically the set of unseen testing requirement documents referred to as P_2. To accomplish this, we use a pre-trained classifier for Step 6 of our approach (as shown in Figure 4.1). Importantly, this classifier has to be completely independent of P_2. To build the necessary classifier, we use the training set derived from

P_1, which we discuss in our previous experiment, EXPIII. To carry out the experiment, we use the same strategy as in EXPI. We randomly withhold half of each document p in P_2 and attempt to predict novel terms contained solely in the withheld half. However, unlike in EXPI, we post-process the predictions made by BERT with a classification filter aimed at reducing noise. This allows us to obtain a better understanding of BERT's capacity to act as source of knowledge for completeness checking.

To obtain more accurate and reliable results, we repeat EXPIV five times for each document p in P_2. This is performed to mitigate the effects of random variation resulting from the selection of the disclosed and withheld halves of each document. Overall, we conduct 300 runs of our approach in EXPIV, given that there are 20 documents in P_2 and we test three levels of filtering. It is important to note that filtering of BERT's output is transparent to users, meaning that we report the results of EXPIV in the same way as in EXPI. Additionally, as an extra measure of validation for EXPIII, we report the quality of filtering using the same procedure as we discussed in EXPIII. However, this time we apply it to the validation set, E, which is created from the previously unseen documents of P_2. The results of EXPIV provide insights into the capacity of BERT to act as a source of knowledge for completeness checking, and the effectiveness of our proposed approach in real-world applications.

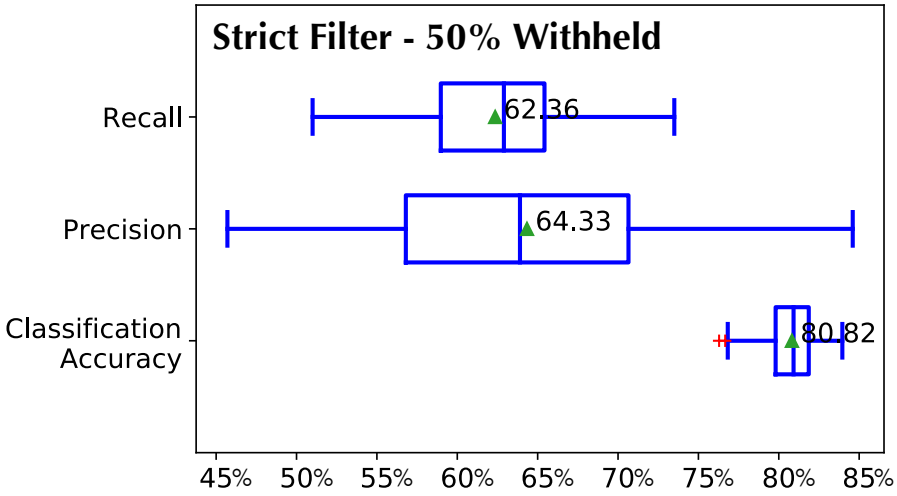

Figure 5.12: Strict Filtering Classification Accuracy, Precision and Recall over P_2.

5.4.4.2 RQ4 Results

We present the performance of our three filters, namely strict, moderate, and lenient, as defined in RQ3. Next, we present end-to-end results by reporting the Accuracy and Coverage of word predictions (over P_2) in the same manner as in RQ1, but with filters applied. The goal of this Q is to measure the overall Accuracy and Coverage of our end-to-end approach including the filtering phase. These values are used to establish whether RQ3 results will hold for the unseen data. Without filters and over our test set (P_2 in Table 5.3), the predictions made by BERT at 15 predictions per mask have an average Accuracy of 12.11% and average Coverage of 40.04%. Box plots are provided in Fig. 5.15. We recall from Section 5.4 that five different random shuffles are performed for each $p \in P_2$. The plots in Fig. 5.15 are based on $5 \times 20 = 100$ runs.

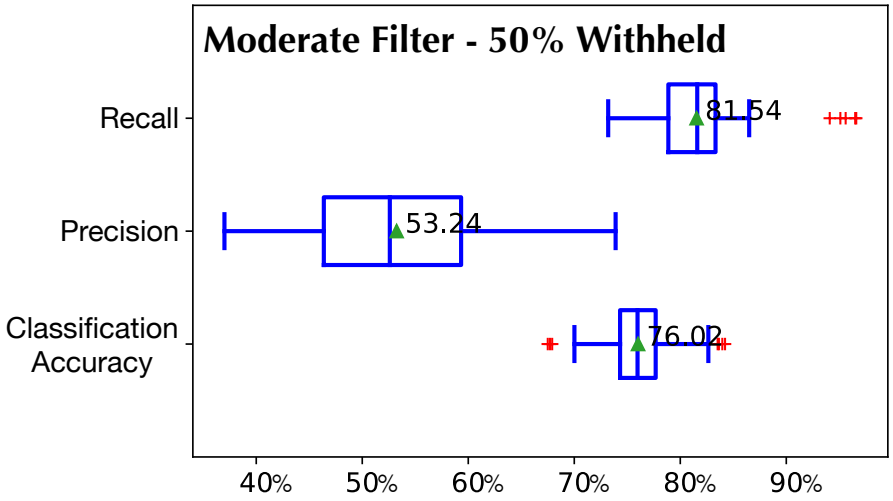

Figure 5.13: Moderate Filtering Classification Accuracy, Precision and Recall over P_2.

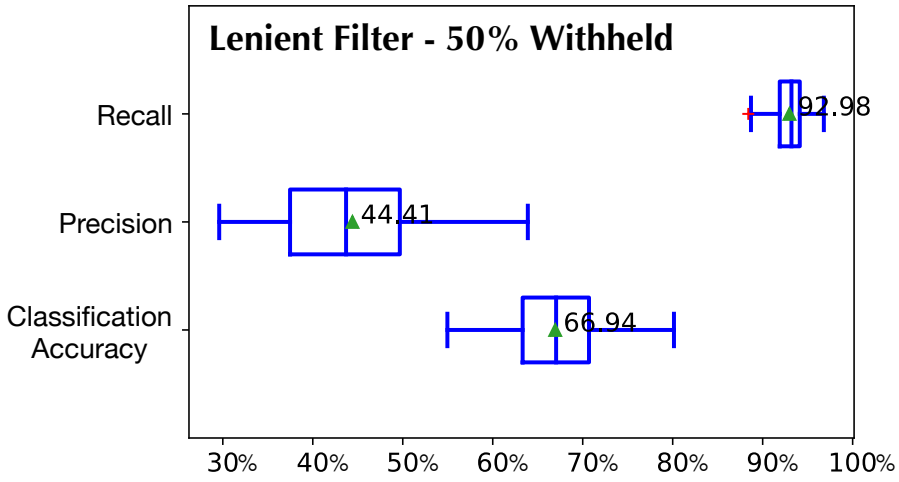

Figure 5.14: Lenient Filtering Classification Accuracy, Precision and Recall over P_2.

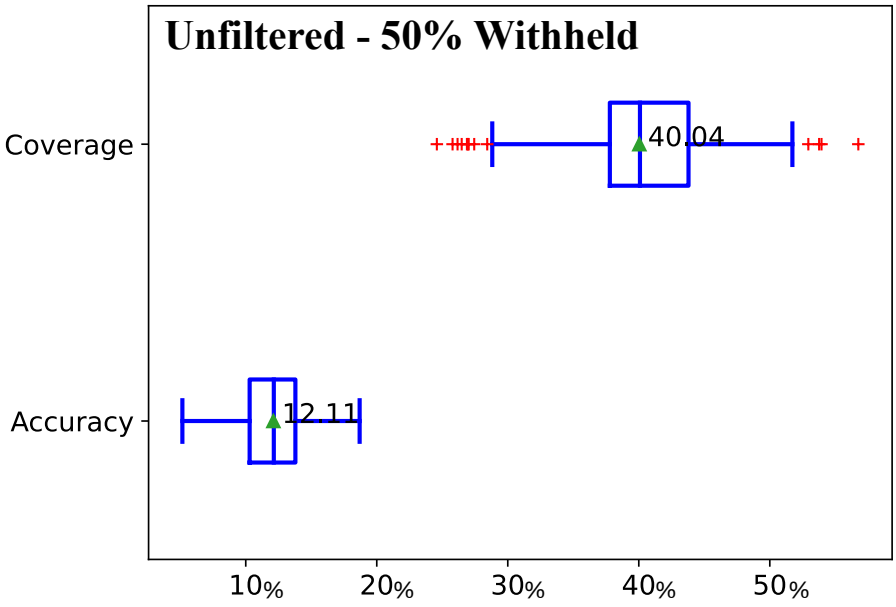

Figure 5.15: 50% Withheld - Accuracy and Coverage over P_2 without filtering.

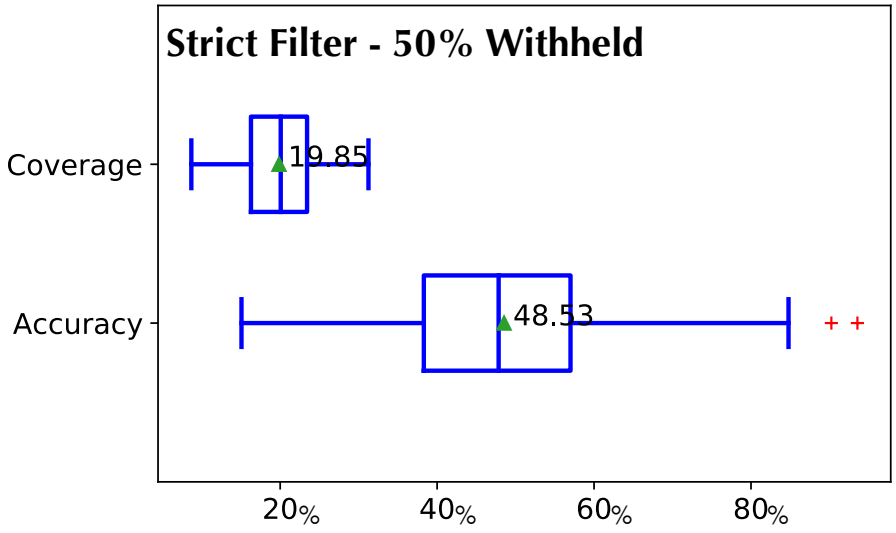

Figure 5.16: 50% Withheld - Accuracy and Coverage over P_2 with Strict Filter.

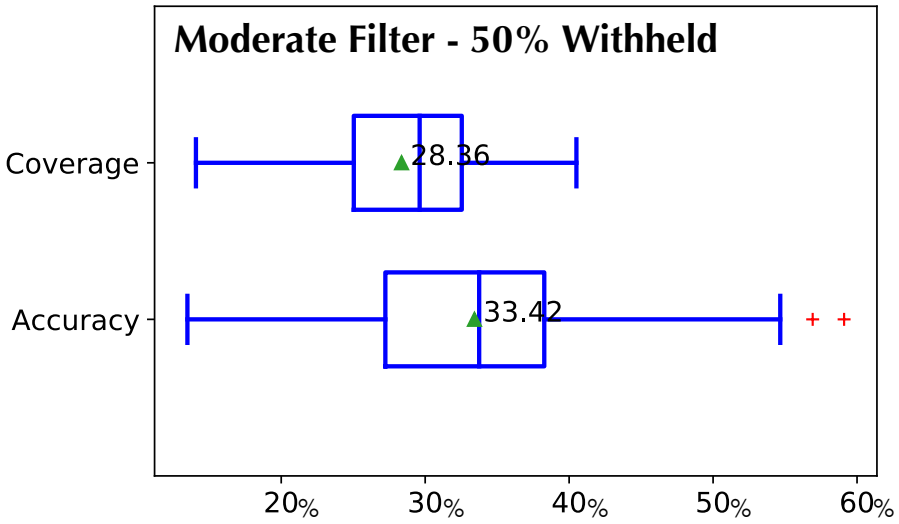

Figure 5.17: 50% Withheld - Accuracy and Coverage over P_2 with Moderate Filter.

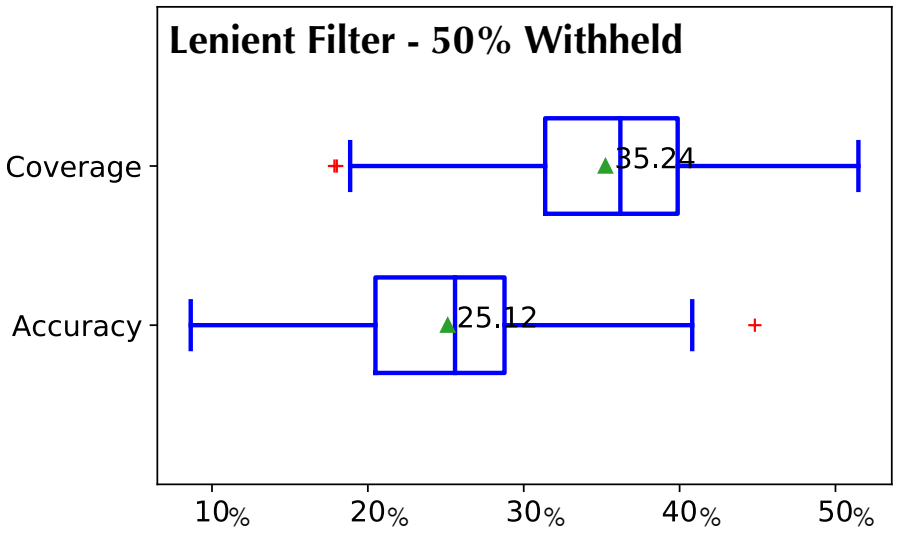

Figure 5.18: 50% Withheld - Accuracy and Coverage over P_2 with Lenient Filter.

Figs. 5.12, 5.13 and 5.14 shows the performance of our three filters, namely strict, moderate, and lenient, over P_2. We observe that for all three filters, Precision levels over unseen data are lower than the cross-validation results in RQ2 (Table 5.6). This discrepancy in Precision is particularly expected for the moderate and lenient filters, noting that, for these two filters, Table 5.6 reports performance over an under-sampled dataset. As for Recall, the results of Fig. 5.12, 5.13 and 5.14 indicate that the filters behave consistently with the trends seen in cross-validation. This consistency provides evidence that filtering (i.e., getting rid 'non-relevant' information) does not overfit to the training data. Thus, this provides sufficient evidence for the generalizability of the classifier. Figs. 5.16, 5.17 and 5.18 show the word-prediction Accuracy and Coverage results *after* filtering. These results can be compared directly against those reported in RQ1 (Figs. 5.2 and 5.3).

Which filtering option the user selects depends on how the user wishes to balance overhead. The user can select from the three listed filters in Figs. 5.16, 5.17 and 5.18 to compromise their level of tolerance for noise versus identifying potentially useful but missing terminology. Aggressive filtering results in nearly half of all predictions containing valuable terminology but ultimately, predictions cover only one-fifth of the potentially missing terminology. Moderate filtering yields one-third of filtered predictions being useful and reveals 28% of novel terms. Lenient filtering predicts only one-fourth of all withheld terms but preserves Coverage and hints at 35% of the potentially missing terminology.

Table 5.7 shows the results of statistical significance testing in which each column compares the unfiltered model against the three filtering models, with respect to Accuracy

and Coverage. Accuracy shows statistically significantly better results for all three filtered models. Even the weakest filtering doubles the Accuracy of the unfiltered model; however, filtering aggression impacts the size of Accuracy improvement. For Coverage, only the Aggressive and Moderate models have a statistically significant difference. This may be interpreted as the Lenient model preserving Coverage (at the expense of Accuracy) and vice versa for the Aggressive and Moderate models.

The decrease in Coverage for the Aggressive and Moderate models could be due to the trade-off of improving Accuracy. The Lenient model successfully preserves Coverage as it does not have a statistically significant difference compared to the RQ1 model. As we observe, the lenient filter increases Accuracy by an average ≈13% while decreasing Coverage by an average ≈5%. The strict filter increases Accuracy by an average ≈36% while decreasing Coverage by an average ≈20%. All filters impact both Accuracy and Coverage in a statistically significantly manner with medium to large effect sizes, with the exception of the (small) Coverage decrease brought about by the lenient filter. It should be noted that there are only 20 data points made for comparison, and these results have been impacted by the small data size.

		Accuracy	Coverage
Unfiltered vs Strict Filter	p-value	1.451E-11	1.407E-09
	Â12	0 (L)	0.9775 (L)
Unfiltered vs. Moderate Filter	p-value	1.451E-11	0.0008358
	Â12	0 (L)	0.8 (L)
Unfiltered vs. Lenient Filter	p-value	5.412E-09	0.1653
	Â12	0.0325 (L)	0.63 (S)
Effect size: Large (L), Medium (M), Small (S), Negligible (N)			

Table 5.7: Statistical Significance of RQ4 over Test Set.

5.4.5 RQ5. Does the size of the withheld portion impact the quality of predictions made BERT?

Research Question 5 (RQ5) aims to explore the minimum size limit for the section of an input RS that can be withheld while still producing relevant predictions made by BERT to answer RQ5. This experiment will also analyze the results obtained from RQ4 in order to determine if the size of the omitted portion of the RS has any influence on the accuracy and precision of the words predicted by BERT. The goal of RQ5 is to determine BERT's potential for predicting missing terminology when there is a smaller amount of incompleteness in the requirements.

5.4.5.1 Experimental Procedure for RQ5

This experiment seeks to answer RQ5, which aims to explore the impact of varying the size of the withheld portion in detecting incompleteness through the predictions generated

by BERT. The goal is to determine whether the quality of the predictions produced by BERT is affected by changing the size of the disclosed and withheld sections of an input RS. The experiment aims to compare the results from RQ4 with those of RQ5 to determine whether the size of the omitted portion of the RS influences the accuracy and precision of the predicted terms both before and after filtering. To achieve these goals, the experiment replicates the procedure used in EXPIV and evaluates the performance of BERT in identifying relevant terms from the disclosed subset of the input RS. However, this time, we change the size of the withheld portion from 50% to 20% to evaluate the impact of decreasing the size of the omitted portion.

The same approach we use in RQ4 is applied to the predicted terms in RQ5 to assess BERT's ability to generate relevant predictions under smaller withheld portion sizes. To ensure the reliability of the results, we repeat the experiment five times for each document in P_2. In total, we conduct the experiment 300 times, considering that there are 20 documents in P_2, and three withheld portion sizes are being evaluated ($20 \times 5 \times 3 = 300$ runs). The primary objective of this experiment is to explore the applicability of the approach in a scenario where there is less incompleteness present in the requirements. This can be used to determine BERT's usefulness and its capacity for hinting at potential omissions in an RS when there is a limited amount of data being withheld. The insights gained from this experiment can be used in future work for enhancing the effectiveness of completeness checking using language models.

5.4.5.2 RQ5 Results

In our experimentation so far, we use an even split for the disclosed and withheld portions, letting each account for 50% of the content in a given RS. In lieu of user studies with subject-matter experts, we cannot know what split ratio will result in the most useful predictions. As such, we consider another split ratio, where we disclose 80% of an RS and use our approach for predicting the content of the remaining 20%. The intuition for this split ratio is that the document under analysis is assumed to be fairly complete, and as such, there would be fewer opportunities to find relevant terminology that is missing. In RQ5, we compare the effectiveness of withholding 20% of the data against withholding 50% of the data from the requirements given to BERT. We observe that the 20% withheld data exhibits significantly lower coverage after applying a filtering process. This can be attributed to the fact that less information is withheld in the 20% data set, leading to a reduced amount of terminology that BERT can predict and that is not already part of the disclosed portion. Consequently, the diminished coverage of the 20% data implies that retaining a larger percentage of the data may not yield any substantial improvement in detecting incompleteness in natural language requirements.

Fig. 5.19 illustrates that, prior to filtering, Coverage results are very similar between the 20% and 50% withheld data models. However, Accuracy performance of the 20% withheld unfiltered model is three times worse than that of the RQ4 model. Although BERT is exposed to more terminology in RQ5, it is constrained to what is deemed relevant in the

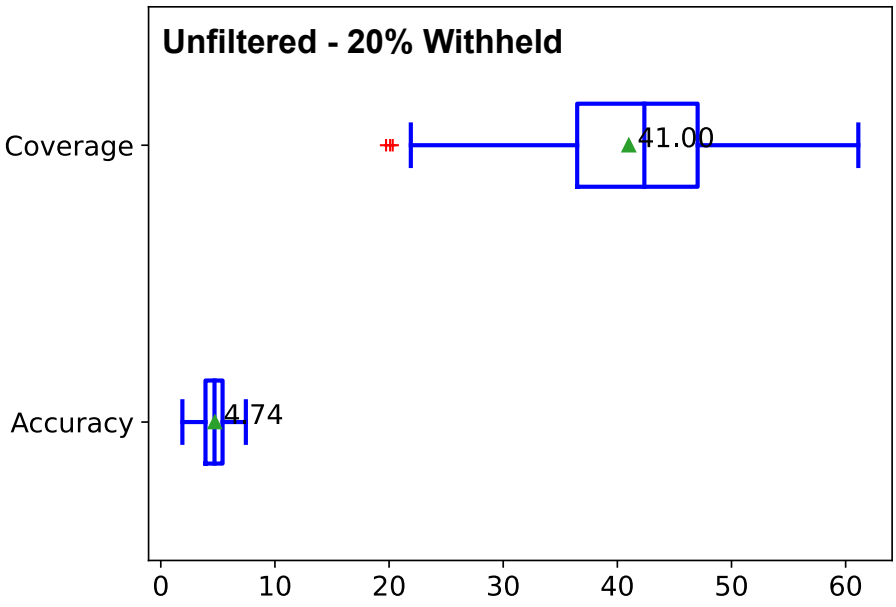

Figure 5.19: 20% Withheld - Accuracy and Coverage over P_2 without filtering.

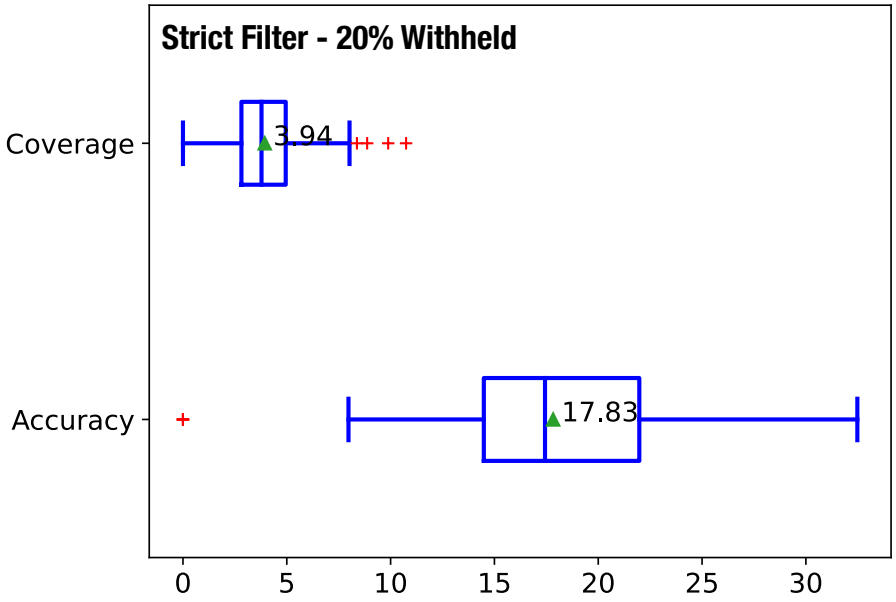

Figure 5.20: 20% Withheld - Accuracy and Coverage over P_2 with Strict Filter.

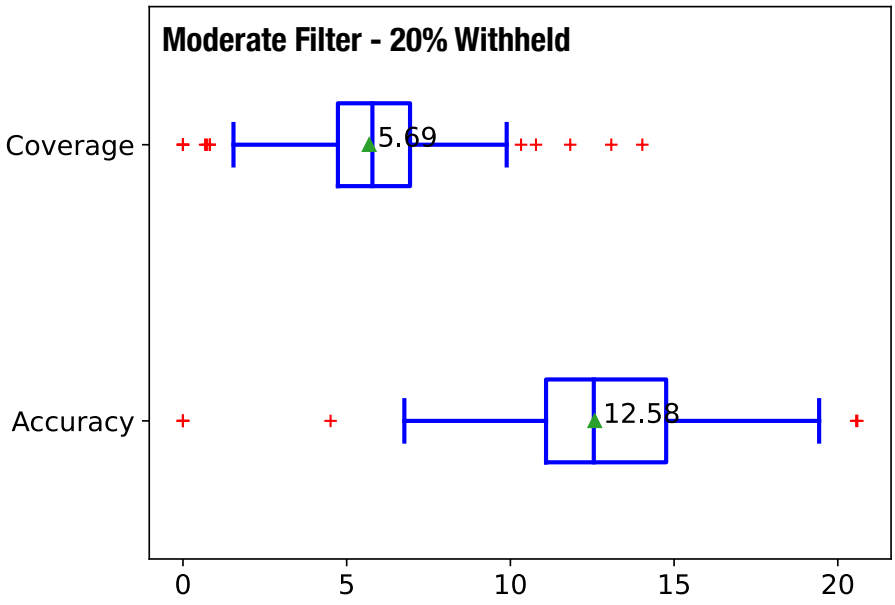

Figure 5.21: 20% Withheld - Accuracy and Coverage over P_2 with Moderate Filter.

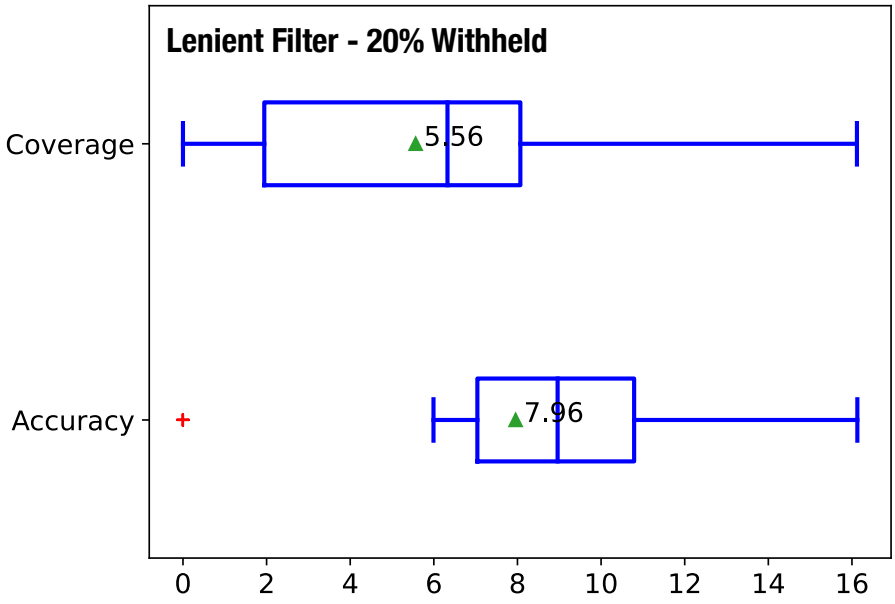

Figure 5.22: 20% Withheld - Accuracy and Coverage over P_2 with Lenient Filter.

		Accuracy	Coverage
50% vs. 20% Unfiltered	p-value	2.20E-16	1.75E-01
	Â12	0.0059 (L)	0.5556 (N)
50% vs. 20% Strict	p-value	2.20E-16	2.20E-16
	Â12	0.14675 (L)	0.0047 (L)
50% vs. 20% Moderate	p-value	2.20E-16	2.20E-16
	Â12	0.09635 (L)	8e-04 (L)
50% vs. 20% Lenient	p-value	2.20E-16	2.20E-16
	Â12	0.0729 (L)	1e-04 (L)
Effect size: Large (L), Medium (M), Small (S), Negligible (N)			

Table 5.8: Statistical Significance of RQ5 over Test Set.

withheld half due to its limited size. Further investigation is necessary, but we surmize that this discrepancy arises from the uneven distribution of detected incompleteness throughout the document. Consequently, the opportunity to find relevant matches is diminished, resulting in the lower coverage observed in Figures 5.20, 5.21, and 5.22. In fact, after filtering, the coverage for all three models suffers significantly compared to the post-filtered models in RQ4. Figure 5.19 offers an overview of the statistical significance between the models from RQ4 and RQ5, highlighting the differences in performance between the varying levels of withheld data. Without conducting user studies, we are unable to definitively determine the degree of incompleteness one can anticipate in real-world applications. As the strictness of relevant terms increases, the likelihood of identifying matches decreases. However, the precise amount of incompleteness can only be ascertained through user studies.

5.5 Limitations and Validity Considerations

Below, we discuss potential limitations and threats to validity.

5.5.1 Limitations.

The experimentation we conduct in this thesis is subject to several notable limitations that warrant further investigation. The first limitation results from a lack of access of domain experts who can effectively identify genuine cases of incompleteness in the requirements. As a result, we resort to simulating incompleteness by withholding content from existing requirements. While this allows for some measure of evaluation for our proposed approach's effectiveness, it may not accurately represent the actual incomplete requirements encountered in real-world scenarios. To develop a more precise understanding of our approach's utility, it is essential to conduct future user studies involving domain experts who can provide insights into the nature and extent of incompleteness in requirements. Another potential limitation is the size and diversity of our dataset employed in this thesis. Although our dataset consists of 15 distinct domains, it may not be representative of all possible scenarios for different industries and contexts. To provide a more comprehensive evaluation of the approach, future experimentation should consider utilizing the entirety of PURE [30] and potentially other datasets. This will allow for a broader assessment of our method's effectiveness across various domains. Lastly, limited availability of computational resources and manual effort required for document cleanup prevented further systematic experimentation.

5.5.2 Internal Validity.

We intentionally introduce incompleteness into our experiment to evaluate the effectiveness of our approach in detecting missing requirements. However, we recognize that random variation could potentially influence our findings. In order to mitigate the impact of random variation on our results, we employ several strategies. First, we utilize a substantial dataset consisting of 40 RS, as depicted in Table 5.3. This large dataset helps minimize the effects of random variation, thereby increasing the robustness of our findings. Furthermore, we enhance the reliability of our experiment by repeating each test for a given RS five times. This is achieved by shuffling the withheld and disclosed portions of the RS, as outlined in Section 5.4. By doing so, we are able to further reduce the impact of random variation on our results.

5.5.3 Construct Validity.

In order to ensure that our approach is accurate and comprehensive, we take several steps to refine our analysis. First, we exclude any terminology already present in the disclosed portion of the requirements documents. This process eliminates the possibility of including non-novel terms. Additionally, we remove duplicates, common words, and stopwords from our analysis. These measures ensure that we provide an objective assessment of the novel terminology identified by the BERT MLM algorithm.

5.5.4 Conclusion Validity.

In order to ensure the validity of our conclusions, we highlight two critical factors that should be considered. Firstly, our experiments primarily focus on a 50-50 split between disclosed and withheld requirements documents. While we assume that the useful terminology is evenly distributed throughout the withheld portion, we believe that comparable benefits could be obtained with different split ratios, provided the withheld portion is not excessively small. It is also essential to recognize there is a limit to how small the withheld portion can be before it becomes challenging to accurately predict its contents based solely on the disclosed portion. This threshold will determine the sensitivity of our approach to incompleteness, and additional research is needed to establish and quantify this limit. We attempt to address this concern in RQ5 by implementing an 80-20 split. However, as stated in the discussion, user studies are necessary to arrive at more conclusive results.

5.5.5 External Validity.

Our evaluation is based on an extensive analysis of 40 RS sourced from the publicly available PURE dataset [30]. These RS span a diverse range of 15 different domains and originate from a variety of sources. The size and diversity of our dataset provide a certain level of confidence in the generalizability of our findings. However, to ensure the external validity of our results and gain a more comprehensive understanding of our approach's effectiveness, further case studies are necessary to investigate its performance on different types of RS.

In order to ensure that our findings are applicable to real-world scenarios, it is crucial to consider other sources of RS beyond the PURE dataset. Expanding the scope of our dataset serves a dual purpose: it not only increases the diversity of our training set but also offers an opportunity to test the robustness of our approach across various domains that may not have been examined during our initial experimentation. By incorporating a broader range of RS, we can better understand the applicability and efficacy of our approach in addressing incompleteness in natural language requirements.

Chapter 6

Conclusion

The central aim of this thesis was to examine the effectiveness of language models as a viable source of knowledge within the context of requirements engineering. In particular, the study set out to determine whether masked-word predictions produced by the BERT model could pinpoint potential areas of incompleteness within requirements. To ensure that our approach was applicable across various domains, we made use of a subset of the PURE dataset composed of 40 documents that covered a range of 15 distinct domains. Subsequently, we refined the predicted terms to optimize the predictions and minimize any noise produced by the BERT model. The main objective of the filtering process was to discard non-relevant terms and, as a result, increase the accuracy of the predictions.

In order to assess the effectiveness of our approach, we introduced incompleteness into the requirements by deliberately withholding content from the model. The methodology for

our investigation was comprised of three main steps. The first step involved determining the optimal number of predictions per masked word, striking a balance between the noise and relevant predictions generated by the BERT model. During the second step, we examined various machine learning classifiers to determine the most suitable one for the purpose of eliminating noise from the acquired predictions. Finally, we evaluated the outcomes by contrasting the filtered predictions against the terms found within the withheld section of the requirements specification on new, unseen data.

The findings of this thesis indicate that employing masked-word predictions generated by the BERT model can effectively hint at potential incompleteness within requirements. The filtering process substantially enhances the accuracy of the predictions by reducing the prevalence of non-relevant terms. Nevertheless, our experiments underscore the importance of conducting additional research in this area. Specifically, future studies should explore the applicability of this approach to different domains and we highlight the importance of incorporating user studies into subsequent research. Below, we summarize the main findings of this thesis and outline avenues for future research.

6.1 Main Findings

The main findings from the research conducted in this thesis are the following:

BERT's optimal number of generated predictions per masked word. When requirements omissions are simulated by withholding, having BERT make 15 predictions

per mask is the best trade-off for detecting missing terminology. BERT predicts terms that, on average, hint at ≈4 out of 10 omissions (Coverage ≈38%). On average, ≈1 in 8 predictions is relevant (Accuracy ≈12%).

BERT's performance vis à vis baselines. As there are not direct research to compare our results with, we evaluated the performance of our approach to common-sense baselines. The baselines consisted of a list of words common to the English language, a list of TF-IDF values for terms appearing in a given RS' domain-specific corpus, and a list of synonyms to words appearing in the disclosed portion of a given RS. These baselines enabled us to determine the usefulness of BERT's context-specific predictions against less expensive methods to hint at requirements incompleteness. We concluded that although two of the three baselines offered improved accuracy, the coverage of the predictions was significantly worse (≈5000%) than that obtained of BERT.

Best machine learning model for filtering noise from BERT's predictions. Our analysis revealed that the Random Forest (RF) and Support Vector Machine (SVM) classifiers were the most accurate filters for removing unuseful predictions. RF was found to be more suitable for aggressive filtering, while SVM was better suited for more lax filtering, which is better at preserving Recall.

Performance of our approach on unseen data. Depending on how aggressively one chooses to filter noise from BERT's masked-word predictions, the average Accuracy of our approach ranges between ≈49% and ≈25%. With a strict filter, approximately one in two recommendations made by our approach is relevant, whereas with a lenient filter,

approximately one in four is. With a lenient filter, the recommendations hint at ≈35% of the (simulated) missing terminology. With a strict filter, this number decreases to ≈20%.

Sensitivity to incompleteness. Our experimental design of withholding 50% of the text does not necessarily reflect the incompleteness in real-world settings. We therefore conducted experimentation where we withheld only 20% of the content to find instances of incompleteness that would be more difficult to pinpoint. We found that before filtering, the 20% withheld model offered similar coverage to the 50% withheld model results, but with worse accuracy results. The 20% withheld post-filter results for all three filtering levels were significantly worse compared to the 50% withheld post-filter results. This analysis highlights the importance of user-studies in determining the degree of incompleteness present in real-world scenarios. Overall, we believe that our contributions provide valuable insights into the effectiveness and applicability of our approach for requirements completeness checking.

6.2 Areas for Improvement and Future Works

The research conducted in this thesis examined the use of language models for detecting incompleteness in requirements. Throughout the research, several areas for improvement came to light. These areas can be further explored to improve our understanding of the current results as well as to refine our methodology for detecting incompleteness.

One significant area of improvement is the expansion of the dataset used in the study.

For the purposes of the thesis, we only utilized half of the PURE dataset to balance the effort of data cleaning and computation costs to having a reasonably sized dataset. However, by incorporating the entire PURE dataset instead of only half, the sample size will be larger and more diverse, which can consequently enhance the models' training and testing. A more comprehensive dataset will also provide a better representation of the complexities of real-world requirements engineering scenarios. Thus, the use of the complete PURE dataset will contribute to more robust, reliable, and generalizable findings.

Another critical area for improvement lies in the evaluation of our approach. Our current evaluation relies on simulating omissions through withholding, which, while providing useful insights, does not fully capture the practical implications of our approach. Therefore, it is imperative to incorporate user studies in future research to gather contextual insights into the models' performance in real settings. These user studies should involve domain experts, requirements engineers, and other stakeholders who can vet BERT's predictions as being either superfluous or genuine instances of incompleteness. A human-in-the-loop evaluation process is crucial for several reasons. First, it allows us to verify the accuracy of the models' predictions and determine their practical value in the requirements engineering process. Second, domain experts can provide feedback on the usability and effectiveness of the model, which is imperative in refining the models and developing more user-friendly tools. Third, human involvement can help identify potential biases or blind spots in the language models that may not be apparent through automated evaluations. This feedback can be utilized to further enhance the models, making them more effective at identifying

and addressing incompleteness in requirements engineering.

As we look forward to future work in this thesis, several promising avenues for research can be explored to further enhance the efficacy of language models in detecting incompleteness in requirements engineering. One such direction is the investigation of BERT variants and their potential for improving the accuracy and reliability of predictions. By experimenting with different BERT architectures, we can optimize the models' performance by leveraging the strengths of each variant and identifying the most suitable configuration for the task at hand. Fine-tuning these language models on domain-specific corpora can further increase their relevance to the requirements engineering context. This process will enable the models to better pick up on the nuances of the language used in requirements documents, ultimately leading to more accurate and contextually relevant predictions of incompleteness. Another promising avenue for future research is the exploration of the benefits of using noun-phrase and verb-phrase structures instead of solely relying on noun and verb part-of-speech tags. By considering these more complex linguistic structures, the language models can capture deeper semantic relationships between the various elements of a requirement, thereby enhancing their ability to identify gaps or omissions that may otherwise be overlooked. Yet another area for future work is to integrate our approach into existing requirements management tools to make it more usable for practitioners.

In future work, we plan to pursue as many of the above-described directions as possible in order to more comprehensively assess the usefulness of language models for handling incompleteness in requirements.

References

[1] Muhammad Abbas, Alessio Ferrari, Anas Shatnawi, Eduard Paul Enoiu, and Mehrdad Saadatmand. Is requirements similarity a good proxy for software similarity? an empirical investigation in industry. In *Requirements Engineering: Foundation for Software Quality: 27th International Working Conference, REFSQ 2021, Essen, Germany, April 12–15, 2021, Proceedings*, page 3–18. Springer-Verlag, 2021.

[2] Dalal Alrajeh, Jeff Kramer, Axel van Lamsweerde, Alessandra Russo, and Sebastian Uchitel. Generating obstacle conditions for requirements completeness. In *2012 34th International Conference on Software Engineering (ICSE)*, pages 705–715, 2012.

[3] Orlando Amaral Cejas, Sallam Abualhaija, Damiano Torre, Mehrdad Sabetzadeh, and Lionel Briand. AI-enabled automation for completeness checking of privacy policies. *IEEE TSE*, 48(11), 2022.

[4] Chetan Arora, Mehrdad Sabetzadeh, and Lionel Briand. An empirical study on the potential usefulness of domain models for completeness checking of requirements. *EMSE*,

24(4), 2019.

[5] Chetan Arora, Mehrdad Sabetzadeh, Lionel Briand, and Frank Zimmer. Automated checking of conformance to requirements templates using natural language processing. *IEEE TSE*, 41(10), 2015.

[6] Chetan Arora, Mehrdad Sabetzadeh, Lionel Briand, and Frank Zimmer. Automated extraction and clustering of requirements glossary terms. *IEEE TSE*, 43(10), 2017.

[7] Chetan Arora, Mehrdad Sabetzadeh, and Lionel C. Briand. An empirical study on the potential usefulness of domain models for completeness checking of requirements. *Empirical Softw. Engg.*, 24(4):2509–2539, aug 2019.

[8] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *JMLR*, 13(2), 2012.

[9] Daniel M. Berry. In *Assessing Tools for Defect Detection in Natural Language Requirements : Recall vs Precision*, 2017.

[10] Daniel M. Berry. Evaluation of tools for hairy requirements and software engineering tasks. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, pages 284–291, 2017.

[11] Daniel M. Berry, Jane Cleland-Huang, Alessio Ferrari, Walid Maalej, John Mylopoulos, and Didar Zowghi. Panel: context-dependent evaluation of tools for NL RE tasks: recall vs. precision, and beyond. In *RE*, 2017.

[12] Daniel M Berry et al. Panel: context-dependent evaluation of tools for nl re tasks: recall vs. precision, and beyond. In *2017 IEEE 25th International requirements engineering conference (RE)*, pages 570–573. IEEE, 2017.

[13] Daniel M. Berry, Erik Kamsties, and Michael Krieger. From contract drafting to software specification: Linguistic sources of ambiguity, a handbook, 2003.

[14] Jaspreet Bhatia and Travis Breaux. Semantic incompleteness in privacy policy goals. In *RE*, 2018.

[15] Jie Cai et al. Feature selection in machine learning: A new perspective. *Neurocomputing*, 300:70–79, 2018.

[16] J. Anthony Capon. *Elementary Statistics for the Social Sciences: Study Guide*. Wadsworth, 1991.

[17] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra: Pre-training text encoders as discriminators rather than generators, 2020.

[18] Gaoying Cui, Qin Lu, Wenjie Li, and Yi-Rong Chen. Corpus exploitation from Wikipedia for ontology construction. In *LREC*, 2008.

[19] Fabiano Dalpiaz, Ivor van der Schalk, and Garm Lucassen. Pinpointing ambiguity and incompleteness in requirements engineering via information visualization and NLP. In *Requirements Engineering: Foundation for Software Quality*, 2018.

[20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[21] Jonas Eckhardt, Andreas Vogelsang, Henning Femmer, and Philipp Mager. Challenging incompleteness of performance requirements by sentence patterns. In *2016 IEEE 24th International Requirements Engineering Conference (RE)*, pages 46–55, 2016.

[22] Sergio Espana, Nelly Condori-Fernandez, Arturo Gonzalez, and Óscar Pastor. Evaluating the completeness and granularity of functional requirements specifications: A controlled experiment. In *2009 17th IEEE International Requirements Engineering Conference*, pages 161–170, 2009.

[23] Saad Ezzini, Sallam Abualhaija, Chetan Arora, and Mehrdad Sabetzadeh. Automated handling of anaphoric ambiguity in requirements: A multi-solution study. In *ICSE*, 2022.

[24] Saad Ezzini, Sallam Abualhaija, Chetan Arora, Mehrdad Sabetzadeh, and Lionel Briand. Using domain-specific corpora for improved handling of ambiguity in requirements. In *ICSE*, 2021.

[25] Saad Ezzini, Sallam Abualhaija, and Mehrdad Sabetzadeh. WikiDoMiner: Wikipedia domain-specific miner. In *ESEC/FSE*, 2022.

[26] Alberto Fernández et al. Cost-sensitive learning. In *Learning from Imbalanced Data Sets*, pages 63–78. Springer, 2018.

[27] Alessio Ferrari, Felice dell'Orletta, Giorgio Oronzo Spagnolo, and Stefania Gnesi. Measuring and improving the completeness of natural language requirements. In Camille Salinesi and Inge van de Weerd, editors, *REFSQ*, 2014.

[28] Alessio Ferrari, Beatrice Donati, and Stefania Gnesi. Detecting domain-specific ambiguities: an NLP approach based on Wikipedia crawling and word embeddings. In *AIRE*, 2017.

[29] Alessio Ferrari, Giuseppe Lipari, Stefania Gnesi, and Giorgio O. Spagnolo. Pragmatic ambiguity detection in natural language requirements. In *2014 IEEE 1st International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, pages 1–8, 2014.

[30] Alessio Ferrari, Giorgio Oronzo Spagnolo, and Stefania Gnesi. PURE: A dataset of public requirements documents. In *RE*, 2017.

[31] Gabriella Gigante, Francesco Gargiulo, and Massimo Ficco. A semantic driven approach for requirements verification. In David Camacho, Lars Braubach, Salvatore Venticinque, and Costin Badica, editors, *Intelligent Distributed Computing VIII*, pages 427–436, Cham, 2015. Springer International Publishing.

[32] Ian Goodfellow et al. *Deep learning*. MIT press, 2016.

[33] Hussein Hasso, Katharina Großer, Iliass Aymaz, Hanna Geppert, and Jan Jürjens. Abbreviation-expansion pair detection for glossary term extraction. In Vincenzo Gervasi and Andreas Vogelsang, editors, *Requirements Engineering: Foundation for Software Quality*, pages 63–78. Springer International Publishing, 2022.

[34] Tobias Hey, Jan Keim, Anne Koziolek, and Walter F. Tichy. NoRBERT: Transfer learning for requirements classification. In *RE*, 2020.

[35] Julia Hirschberg and Christopher D. Manning. Advances in natural language processing. *Science*, 349(6245):261–266, 2015.

[36] Daniel Jurafsky and J. Martin. *Speech and Language Processing*. Prentice Hall, 2 edition, 2009.

[37] Tomasz P Krzeszowski. *Contrasting languages: The scope of contrastive linguistics*, volume 51. Walter de Gruyter, 2011.

[38] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[39] Huan Liu and Hiroshi Motoda. *Feature selection for knowledge discovery and data mining*, volume 454. Springer Science & Business Media, 2012.

[40] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, 2019.

[41] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn Van der Werf, and Sjaak Brinkkemper. Improving agile requirements: the quality user story framework and tool. *Requirements Engineering*, 21, 09 2016.

[42] Dipeeka Luitel, Shabnam Hassani, and Mehrdad Sabetzadeh. Replication package, 2023. https://doi.org/10.6084/m9.figshare.22041341.

[43] Dipeeka Luitel, Shabnam Hassani, and Mehrdad Sabetzadeh. Using language models for enhancing the completeness of natural-language requirements. In Alessio Ferrari and Birgit Penzenstadler, editors, *Requirements Engineering: Foundation for Software Quality*, pages 87–104. Springer Nature Switzerland, 2023.

[44] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Syngress, 2008.

[45] Detmar Meurers. Natural language processing and language learning. *Encyclopedia of applied linguistics*, pages 4193–4205, 2012.

[46] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *NAACL-HLT*, 2013.

[47] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. Deep learning–based text classification: A comprehensive review. *ACM Computing Surveys (CSUR)*, 54(3):1–40, 2021.

[48] OpenAI. OpenAI, 2015. Accessed: March 26, 2023.

[49] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *EMNLP*, 2014.

[50] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, 2018.

[51] Abhishek Sainani, Preethu Rose Anish, Vivek Joshi, and Smita Ghaisas. Extracting and classifying requirements from software engineering contracts. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pages 147–157, 2020.

[52] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, 2019.

[53] Yuchen Shen and Travis Breaux. Domain model extraction from user-authored scenarios and word embeddings. In *AIRE*, 2022.

[54] Amin Sleimi, Nicolas Sannier, Mehrdad Sabetzadeh, Lionel Briand, and John Dann. Automated extraction of semantic legal metadata using natural language processing. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pages 124–135, 2018.

[55] Andras Vargha and Harold Delaney. A critique and improvement of the CL common language effect size statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics*, 25(2), 2000.

[56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[57] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.

[58] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 4 edition, 2017.

[59] Ian H. Witten, Eibe Frank, Mark A. Hall, and Christopher J. Pal. *The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques"*. Morgan Kaufmann Publishers Inc., 4th edition, 2016.

[60] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing [review article]. *IEEE Computational Intelligence Magazine*, 13:55–75, 08 2018.

[61] Liping Zhao, Waad Alhoshan, Alessio Ferrari, Keletso J. Letsholo, Muideen A. Ajagbe, Erol-Valeriu Chioasca, and Riza T. Batista-Navarro. Natural language processing for requirements engineering: A systematic mapping study. *ACM Comput. Surv.*, 54(3), 2021.

[62] Didar Zowghi and Vincenzo Gervasi. On the interplay between consistency, completeness, and correctness in requirements evolution. *IST*, 45(14), 2003.