Technical Documents Version Management System Based on Large Language Models

S. S. Zalivaka
SK hynix memory solutions Poland
224 Juliusza Slowackiego,
Gdansk, 80-298

***Abstract*-Analysis of the technical documents is a crucial part of the design and verification process. Since manual analysis requires significant efforts from the engineers, automation using natural language processing techniques is a promising direction of the efficiency improvement. This research shows that utilization of the information from the previous versions of the document increases the overall performance of the automation tools. The proposed method is based on two models, i.e., sentence embedding for the existing requirements detection and an ensemble of large language models, which recognizes new requirements. As a result, the combination of these algorithms gives an increase in F1-score comparing to the existing works (0.901 vs 0.754). Furthermore, the proposed approach works with the requirements of varying length and takes the context in a more flexible way.**

I. Introduction

Development of any technical system is based on requirements, which are implemented by the designers and thoroughly examined by verification engineers. At the same time requirements are based on different kinds of technical documents. According to the ISO 01.110 standards, there are various types of technical documents: patents, specifications, datasheets, test methods, manufacturing standards, system requirements, system architecture, system design, etc. [1].

Nowadays natural language processing (NLP) methods are becoming ubiquitous for the text processing and analysis tasks. Several works [2-3] have shown that requirements for software and hardware development tasks can be automatically extracted with decent accuracy (more than 80 %). However, the existing approaches do not take into account previous versions of the documents [4] and analyze limited and fixed amounts of text (e.g., sentence or paragraph) [2-3]. There are also Large Language Model (LLM) driven approaches based on prompt engineering techniques [5]. Nevertheless, as shown in [3], even a very large generic model performs less successfully if it was not trained within a specific domain.

The proposed technique is a two-stage algorithm. First, it compares the requirements from the previous version of the document with the newer version and detects the requirements with the same text or slightly modified ones based on the sentence embedding technique [6]. Second, it applies an ensemble [7] of LLM based classifiers to detect new requirements. This approach allows detecting 100% of previously existed requirements and around 80% of the new requirements, which increases the overall accuracy up to over 90%.

II. The Proposed Approach

A. Dataset description

The M-PHY 4.1 specification [8] has been analyzed manually for the purpose of verification by extracting requirements and designing test environment to check the correctness of the protocol operation. As a result, the specification has been split into 219 pages of text consisting 4629 sentences: 772 of them are related to the requirements and 3857 are not. An example of data extracted from page 24 is shown in Fig. 1.

Since the majority of the technical documents are represented in a form of *.pdf file, the parsing process is challenging due to the difficulties of extracting different types of information, i.e., text data, images, tables, lists, footnotes, etc. [9]. This process can be simplified by converting initial pdf-file into *.doc/*.docx format by using Adobe tools [10] or openly available Python libraries (e.g., pdf2docx [11]). The doc-file can be parsed in an easier way because it is stored in an XML-like format, which makes parsing and processing simpler.

To parse M-PHY 4.1 specification, the specification file has been converted to *.docx format using pdf2docx and further analyzed using Unstructured library [12]. Unfortunately, the current way of parsing does not always provide 100% quality and requires manual review to confirm that the parsed text blocks are correctly split into sentences and the content of figures, tables and footnotes is separated. Then, the text elements are combined with labels into a *.json format, as shown in Fig. 1.

Different technical documents are typically edited in various ways depending on preferences of the publisher. As a result, it is almost impossible to design a universal parsing algorithm handling all the issues that arise during the document processing. Thus, currently the manual post-processing is an unavoidable practice.

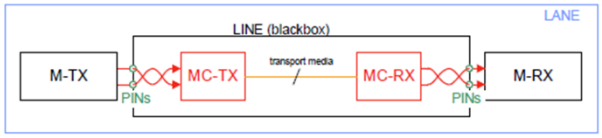

Figure 2 Example LANE Configuration with Media Converter

115 An interface based on M-PHY technology shall contain at least one LANE in each direction. There are no symmetry requirements from an M-PHY perspective for the number of LANEs in each direction.

116 All LANEs in the same direction within a LINK are denoted as a SUB-LINK. Two SUB-LINKs with opposite directions plus additional LANE management, which provides bidirectional data transport functionality agnostic to the actual LANE composition, is called a LINK. A set of M-TXs and M-RXs in a device that compose one interface port is denoted as an M-PORT.

117 This document specifies LANEs and their individual parts including M-TX, M-RX, interconnect, and optionally Media Converters. Furthermore, this specification sets some boundary conditions for M-TX and M-RX inside a single M-PORT, which puts some constraints for the usage of LANEs within SUB-LINKs. This document does not specify the LANE management function in order to allow maximum flexibility of LANE exploitation by protocols. Therefore, the composition of LANEs in the two SUB-LINKs and the specification of LANE management, which completes the LINK, is left to protocols applying M-PHY technology.

4.2 LINE States

118 M-PHY technology exploits only differential signaling. a LINE can show the following states:

119 • A positive differential voltage, driven by the M-TX, which is denoted by LINE state DIF-P

120 • A negative differential voltage, driven by the M-TX, which is denoted by LINE state DIF-N

121 • A weak zero differential voltage, maintained by M-RX, which is denoted by LINE state DIF-Z

122 • An unknown, floating LINE voltage, or no LINE drive, which is denoted by LINE state DIF-Q

123 *Table 1* list all possible LINE conditions with the resulting LINE state

Table 1 LINE Conditions and Resulting LINE States

Differential LINE Voltage	M-TX Output Impedance	M-RX Input Impedance	LINE State Set by	LINE State Name
Positive	Low	Any	M-TX	DIF-P
Negative	Low	Any	M-TX	DIF-N
Zero	High	Medium	M-RX	DIF-Z
Unknown or floating	High	High	None	DIF-Q

124 For data transmission, only DIF-P and DIF-N are exploited. DIF-Z can only occur during power-up and power-saving states.

125 The transition point between DIF-Z and DIF-N is defined by the squelch threshold level, which is positioned between the DIF-N and DIF-Z electrical LINE levels (*Section 5.2.5*).The transition point between DIF-P and DIF-N is defined at the zero crossing of the differential signal.


```
"sentences":
[
    "Specification for M PHY Version 4.1",
     "01 Dec 2016",
     "115 An interface based on M PHY technology shall contain at least one LANE in each direction",
     "There are no symmetry requirements from an M PHY perspective for the number of LANEs in each direction",
     "116 All LANEs in the same direction within a LINK are denoted as a SUB LINK",
     "Two SUB LINKs with opposite directions plus additional LANE management, which provides bidirectional data transport functionality agnostic to the actual LANE composition, is called a LINK",
     "A set of M TXs and M RXs in a device that compose one interface port is denoted as an M PORT",
     "117 This document specifies LANEs and their individual parts including M TX, M RX, interconnect, and optionally Media Converters",
     "Furthermore, this specification sets some boundary conditions for M TX and M RX inside a single M PORT, which puts some constraints for the usage of LANEs within SUB LINKs",
     "This document does not specify the LANE management function in order to allow maximum flexibility of LANE exploitation by protocols",
     "Therefore, the composition of LANEs in the two SUB LINKs and the specification of LANE management, which completes the LINK, is left to protocols applying M PHY technology",
     "4.2 LINE States",
     "118 M PHY technology exploits only differential signaling, a LINE can show the following states",
     "119 A positive differential voltage, driven by the M-TX, which is denoted by LINE state DIF-P",
     "120 A negative differential voltage, driven by the M-TX, which is denoted by LINE state DIF-N",
     "121 A weak zero differential voltage, maintained by M-RX, which is denoted by LINE state DIF-Z",
     "122 An unknown, floating LINE voltage, or no LINE drive, which is denoted by LINE state DIF-Q",
     "123 Table 1 list all possible LINE conditions with the resulting LINE state",
     "124 For data transmission, only DIF P and DIF N are exploited",
     "DIF Z can only occur during power up and power saving states",
     "125 The transition point between DIF Z and DIF N is defined by the squelch threshold level, which is positioned between the DIF N and DIF Z electrical LINE levels (Section 5.2.5).",
     "The transition point between DIF P and DIF N is defined at the zero crossing of the differential signal",
     "Copyright 2008 2017 MIPI Alliance, Inc",
     "All rights reserved",
     "Confidential"
]
"labels":
[0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0]
```


Figure 1. An example of data extraction from M-PHY 4.1 specification (page 24).

For the validation purpose, the M-PHY 6.0 [13] specification has been used. This specification has been parsed in a different way, i.e., excluding headers and footers text, section and subsection titles, etc., into 249 pages containing 2892 sentences: 1246 of them are requirements and 1646 are non-requirements.

B. Structure of the version management system

Consider two versions of a document (specification): the first version of the document is x (older version), the second version of the document is y ($x < y$). The block diagram of the proposed method for processing these two documents is shown in Fig. 2.

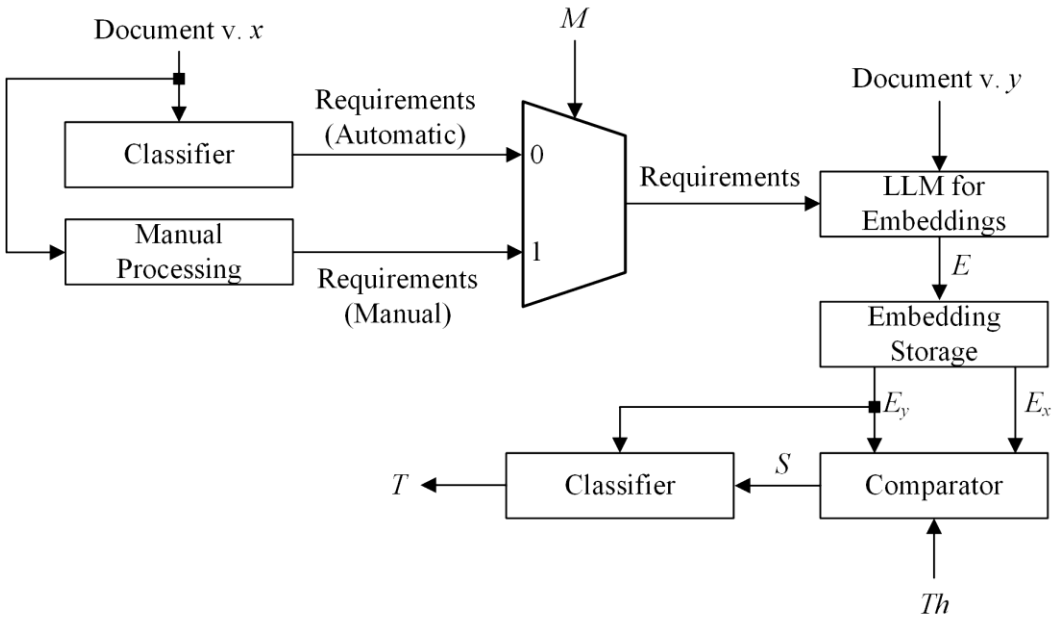

Figure 2. Block diagram of the document version management system.

Requirements from the document v. x can be extracted with a pre-trained classifier (M=0) or manually (M=1). Then, the extracted requirements are processed with an LLM, which computes embeddings for each atomic part. The same LLM is used for the embeddings computation for the content of the document v. y, i.e., the whole content, not just requirements, as they are unknown and need to be recognized. As a result, embeddings of the requirements from the document v. x (E_x) and embeddings of the atomic parts from the document v. y (E_y) are saved to the storage for the embeddings. The storage can be implemented in different ways, e.g., as a vector database [14].

After the preparation step, the embeddings from the storage can be processed to determine the class of the each atomic part of the document v. y. The two steps should be completed for each embedding vector E_y of the newer version of the document. First, vector E_y is compared to the vectors of requirements E_x to find the vector closest to E_y. The similarity metric can be different (e.g., cosine similarity [15]) as well as the algorithm of finding the closest vectors (e.g., approximate nearest neighbor search [16]). Based on the similarity metric, the closest vector gets a similarity metric S (the real number from 0 to 1), which is compared to the threshold value Th (the real number from 0 to 1). If the value of S = 1.0, the vector E_y can be classified as an "exact match" (class 0, T=0). If the value of S is greater than Th and smaller than 1.0, the vector E_y can be classified as a "good match" (class 1, T=1). These two classes can be determined without using the classifier (machine learning model) per se, but can be used as an additional data feature during the classifier training. If the value of S is smaller than Th, the second stage of the proposed algorithm is initiated, i.e., the classifier determines the type of the vector E_y as the one having requirement (class 2, T=2) or as the one without the requirement (class 3, T=3).

As a result, all the atomic parts of the newer version of the document are classified into four types. Class 1 (T=1) can be used to show the editing differences between requirements in the older and newer versions of the document. The quality of the requirements detection highly depends on the classifier performance as it may perform worse on the parts of the document it has not been trained on. Subsequently, the quality of the classifier can be improved by using a large amount of diverse data extracted from technical documents.

C. Classifier description

The basic element of the proposed approach is multi-label classifier based on an LLM. The structure of the classifier with S-sentence context window is shown in Fig. 3.

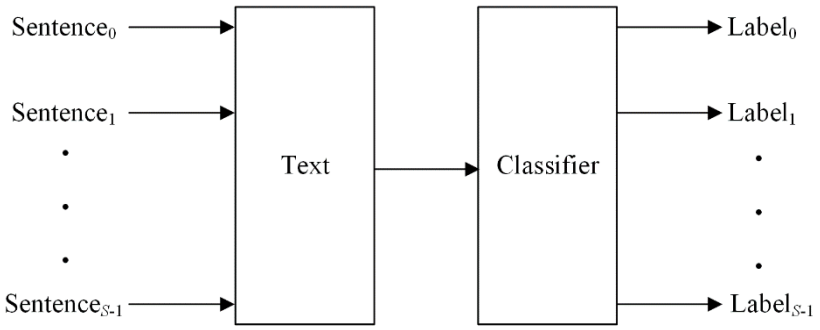

Figure 3. Multi-Label classifier block diagram.

If a dataset has N sentences, the number of data points for this classifier is $ceil(N/S)$. Therefore, the bigger context window size (S) is chosen, the smaller dataset is used for the training process. Since different documents require different sizes of the context window, the multiple (K) multi-label classifiers are used to form an ensemble model.

Ensemble model is based on K models with different context windows $S_0 < S_1 < ... < S_{K-1}$. These models are trained on the same dataset, but split into different number of data points ($ceil(N/S_0)$, $ceil(N/S_1)$, ..., $ceil(N/S_{K-1})$, correspondingly). Each model generates N labels (S_0 - $Label_0^{(S0)}$, ..., $Label_{N-1}^{(S0)}$, S_1 - $Label_0^{(S1)}$, ..., $Label_{N-1}^{(S1)}$, ..., S_{K-1} - $Label_0^{(SK-1)}$, ..., $Label_{N-1}^{(SK-1)}$). Based on label values, the ensemble neural network is trained in order to improve the quality of generated labels ($Label_0^{(E)}$, ..., $Label_{N-1}^{(E)}$). The structure diagram of the proposed approach is shown in Fig. 4.

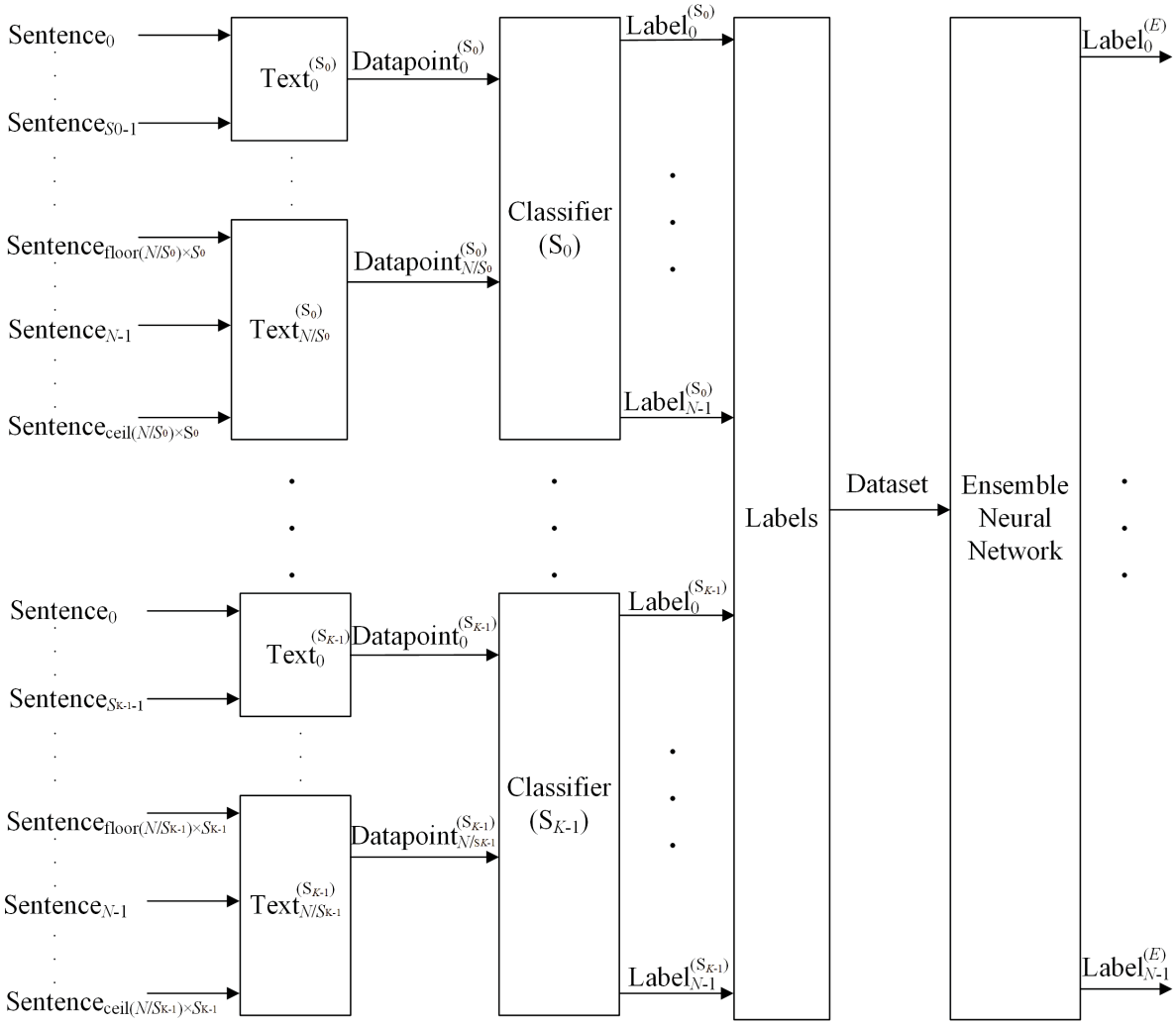

Figure 4. Ensemble model block diagram.

Consider classifier with S_i ($0 \le i \le K$-1) sentences context window. It receives $ceil(N/S_i)$ data points, each of them containing S_i sentences. If N is not a divisible by S_i, the last data point should be appended by N - ($floor(N/S_i)\times S_i$) sentences with some text (e.g., "This is not a requirement") to make the number of sentences in the last data point exactly S_i. All $N\times K$ labels form a dataset for training ensemble neural network, which tunes the weights in order to predict final label value for each sentence based on the K labels provided by classifiers.

As a result, the model provides a balanced prediction of the requirements for the document text taking into account context windows of different size.

III. Experimental Results

A. *Experiment description*

The proposed method has been tested on M-PHY specifications [8-9] versioned as x=4.1 and y=6.0. These documents have been manually processed by verification engineers as a part of the verification IP design process. The atomic parts from both documents have been processed using Sentence Transformers library [17] with the "SFR-Embedding-Mistral" LLM [18] to obtain the embedding vectors. The embedding comparison algorithms have been implemented in three ways: Qdrant vector database [19], embedded semantic search from Sentence Transformers [20], Levenshtein distance based algorithm [21] (this approach does not require embedding vector and can be applied directly to strings). The threshold value *Th* is 0.9 for all algorithms.

B. *Classifier training*

The basic element of the ensemble model is an LLM-based multi-label classifier. In this experiment, the S-label Mistral v.0.1 model [22] has been utilized as a basic classifier. Classifiers with different parameter S = {2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100} (43 classifiers) have been trained on the M-PHY based dataset (70% – training data, 30% – validation data). Each classifier with a fixed hyper-parameter S has been trained for 30 epochs using default Adam optimizer (β_1 = 0.9, β_2 = 0.999), initial learning rate α = 0.0001 and weight decay to decrease the overfitting effects. Each model has been also quantized to 4 bits to fit the size of the GPU memory. The distribution of F1-scores for the classifiers is shown in Fig. 5.

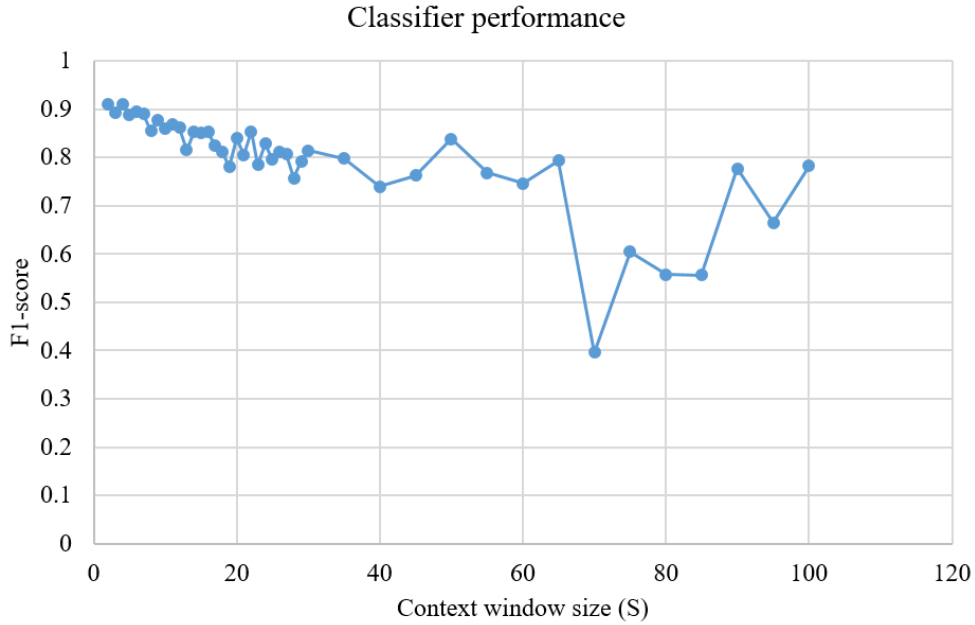

Figure 5. The performance of trained classifiers depending on context window size (S).

As shown in Fig. 5, the performance of the classifier decreases with increasing the context window size (S). Bigger value of the hyper-parameter S leads to decreasing the number of data points in the training set. Therefore, classifiers with smaller context window size usually perform better. Some performance drops (e.g., S = 75) can be explained by poor split into text blocks. Therefore, it is reasonable to use classifiers with $S \le 30$ as the number of data points does not drop under 100 items. Training time for each classifier is around 2 hours and it slightly decreases (by 2-3%) with the increase of the parameter S value.

Since the classifiers have different sizes of the validation set (in sentences), it is hard to assess them by the same metric. Therefore, the number of erroneously classified sentences within the whole specification is chosen as a metric to compare the performance of a single classifier and an ensemble model, utilizing a couple of classifiers. Since the

number of the classifiers is relatively large (43), there is no possibility to compare all possible combinations (2^{43}). Instead, the classifiers were combined sequentially: the first ensemble model contains only one classifier (S={2}), the second – two classifiers (S={2, 3}), the third – three (S={2, 3, 4}), ... the 43th - 43 (S={2, 3, ..., 30, 35, 40, ..., 100}. The comparison between 43 single classifiers and 43 ensemble models is shown in Fig. 6.

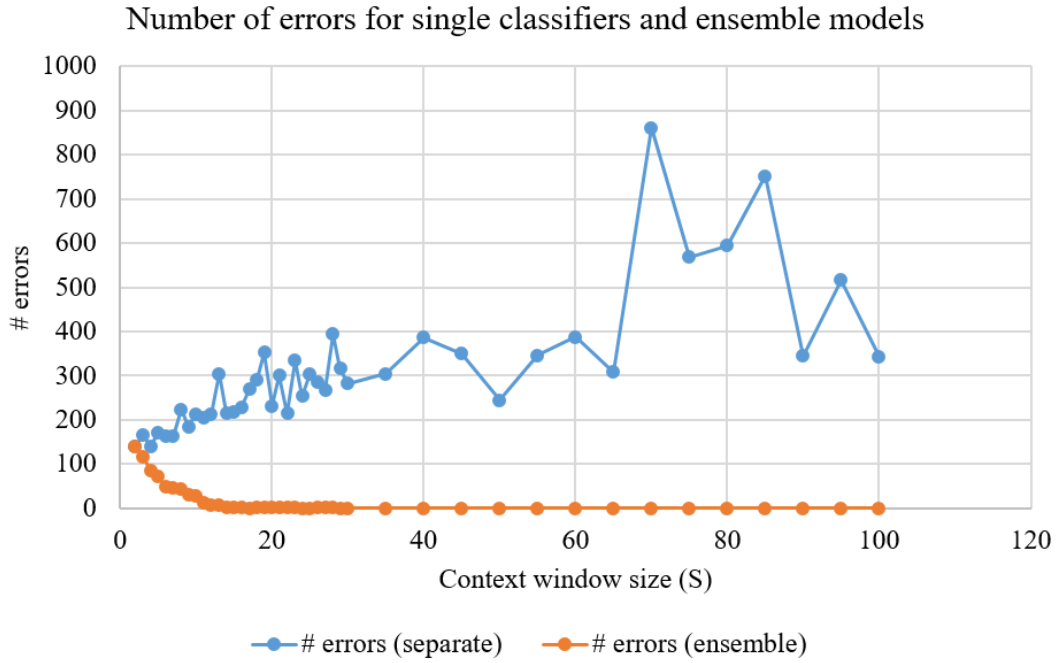

Figure 6. The comparison between single classifiers and ensemble models.

The architecture of the ensemble model is a 5-layer fully-connected network (K Linear (input) -> 1024 ReLU (hidden) -> 512 ReLU (hidden) -> 256 ReLU (hidden) -> 128 ReLU (hidden) -> 1 Sigmoid (output)). ReLU is a rectified linear unit activation function.

The ensemble model has a hyper-parameter K determining the number of context windows used as inputs. The inputs for the model are the probability values generated by chosen classifiers. The model has been trained for 30 epochs using default Adam optimized with initial learning rate α = 0.001. The dropout technique (p = 0.2) for each hidden layer has been utilized to reduce overfitting. The training time even without using GPU is negligible (less than one minute) as the proposed model is relatively small.

To achieve the negligible number of errors, it is enough to get from K=11 (8 errors) to K=16 classifiers (0 errors) model. Increasing the number of classifiers to the ensemble model does not provide a significant increase in performance, i.e., some of the combinations may give 1-2 errors, but the overall quality is comparable.

Unfortunately, the amount of data from one specification is not enough to provide a robust performance on unknown data. The inference of the ensemble model utilizing K=15 classifiers has been tested on M-PHY specification v.6.0 [13]. The ensemble model recognizes 100% of the requirements, which are inherited from the M-PHY v. 4.1 specification, but recognizes only 80% of the new requirements. The outcome of the training process highly depends on the type of the document, its format, the style of the language, format of the tables and figures, etc. Thus, the algorithms trained on one type of specification would unlikely perform very well on another types of technical documents, as it was shown previously [3].

C. Two-stage method results

The requirements from M-PHY v.4.1 specification (772 sentences) has been vectorized using the model for embedding detection without fine-tuning. The embedding vectors for the whole content of the M-PHY v.6.0 specification (2892 sentences) have been also computed. Based on the two-stage algorithm described in section II, 3 requirements with the same text and 550 requirements with similar text have been detected. The ensemble classifier, consisting of K=15 Mistral models fine-tuned on two H100 GPUs [23], has detected 569 new requirements. The other 1770 sentences have been classified as non-requirements. Embedding computation requires around 3-5 minutes and another 15 minutes are required for the classification with ensemble model. The results of the experiment are summarized in Table I.

TABLE I

THE RESULTS OF THE PROPOSED METHOD OBTAINED ON M-PHY SPECIFICATIONS.

Characteristic	Vector database	Semantic search	Levenshtein distance
Hardware	2 H100 GPUs	2 H100 GPUs	Intel i7 CPU
Time for computations (s)	248.498 + 923.45 (classifier time)	167.150 + 923.45 (classifier time)	1.915 + 923.45 (classifier time)
***T*=0 (Exact match)**	3/3	3/3	3/3
***T*=1 (Good match)**	550/550	550/550	490/550
***T*=2 (New Requirements)**	569/694	569/694	582/694
***T*=3 (Non-Requirements)**	1770/1646	1770/1646	1817/1646
Percentage of recognized requirements	90.05	90.05	86.27

The vector database and semantic search approaches provide almost the same results as they are based on the same principles. The slight difference in their performance can be explained by a relatively small amount of data, which is more efficiently processed by the semantic search algorithm. The Levenshtein distance based algorithm has missed 47 requirements because string comparison does not take into account the semantics of the compared strings. The results of the string comparison algorithm can be worse if the amount of data is greater. However, if the computational resources are limited, this algorithm can be considered as an alternative.

The "good match" requirements (*T*=1) are slightly modified from previous specification to the newer one. Some of the examples of these requirements are shown in Table 2 with corresponding changes.

TABLE II

EXAMPLES OF REQUIREMENTS WITH INSIGNIFICANT CHANGES.

Requirements in M-PHY v. 4.1	Requirements in M-PHY v. 6.0	Differences
113 A LANE is a unidirectional, single signal, physical transmission channel used to transport information from point A to point B	**{- 113 }**A LANE is a unidirectional, single signal, physical transmission channel used to transport information from point A to point B**{+ .}**	The number 113 is removed. Full stop is added.
266 TX_HS_ADAPT_Length >= RX_HS_ADAPT_INITIAL_Capability	**{- 266 }**TX_HS_ADAPT_Length **{>= -> ≥}** RX_HS**{+ _Gx}**_ADAPT_INITIAL_Capability	The number 266 is removed. The symbol >= is changed to ≥ Symbols Gx are added after HS.
As shown in Table 8, the number and format of bits differ for different BURST states depending on the signaling scheme	As shown in Table **{8 -> 9}**, the number and format of bits differ for different BURST states depending on the signaling scheme**{+ .}**	The table index is changed from 8 to 9. Full stop is added.
Table 60 M-RX-DATA Interface Signals (continued)	Table **{60 -> 51}** M-RX-DATA Interface Signals**{- (continued)}**	The table index is changed from 60 to 51. The text "(continued)" is removed.

The quality of "good match" requirements detection depends on threshold value *Th*. Therefore, if this value is small (<0.3), the algorithm can wrongly detect non-requirement statements as requirements. On the other hand, the high *Th* values (>0.95) lead to the detection of only "exact match"-like requirements. As a result, tuning this parameter in a proper way allows avoiding erroneous requirements detection in a newer document if the requirements in the original document are correctly identified.

The M-PHY v. 6.0 specification has 1246 requirements and 553 of them have been detected based on requirements similarity from M-PHY v. 4.1. Besides, the classifier has detected 569 new requirements. As a result, 1122 requirements have been recognized and improved the accuracy of the fixed text classifier [3] from approximately 80% to approximately 90%.

IV. THE USE OF THE PROPOSED METHOD IN A VERIFICATION WORKFLOW

The verification process is quite standard and usually based on specifications as a foundation. The modification of the usual workflow is shown in Fig. 7.

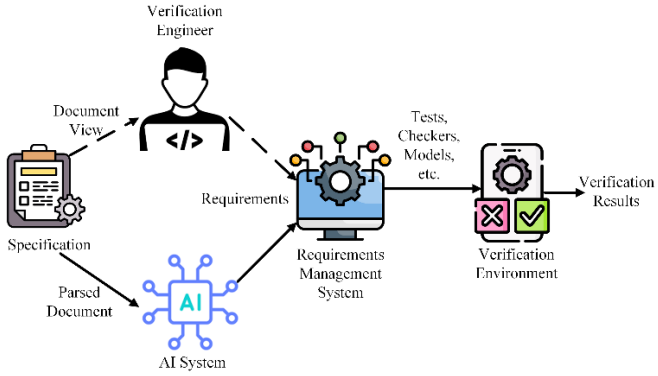

Figure 7. The modified verification workflow.

Specifications are usually analyzed by an experienced verification engineer in order to extract requirements. The proposed method targets to replace this stage with an automatic artificial intelligence (AI) system. The output of this system for a part of the M-PHY specification is shown in Fig. 8.

4 Architecture and Operation

This section specifies the concept, communication principles, signaling schemes, interface structure and operation of M-PHY interfaces.

4.1 PIN, LINE, LANE, SUB-LINK, LINK, and M-PORT

{- 113 }A LANE is a unidirectional, single-signal, physical transmission channel used to transport information from point A to point B. A LANE consists of an M-PHY transmit MODULE (M-TX), an M-PHY receive MODULE (M-RX), and a LINE, which is the point-to-point interconnect between the M-TX and M-RX. An M-TX or M-RX has only one differential electrical output or input LINE interface, respectively, which corresponds with two signaling PINs for each MODULE. The PINs are individually denoted as DP and DN{- -} where DP is defined as the positive node of the differential signal. An optional prefix, TX or RX, can be used to indicate the M-TX or M-RX PINs, respectively. Specifications in this document are defined at the PINs of the M-TX and M-RX, and PINs-to-PINs through the LINE. ***Figure 1*** illustrates the relationship between different parts of an M-PHY LINK.

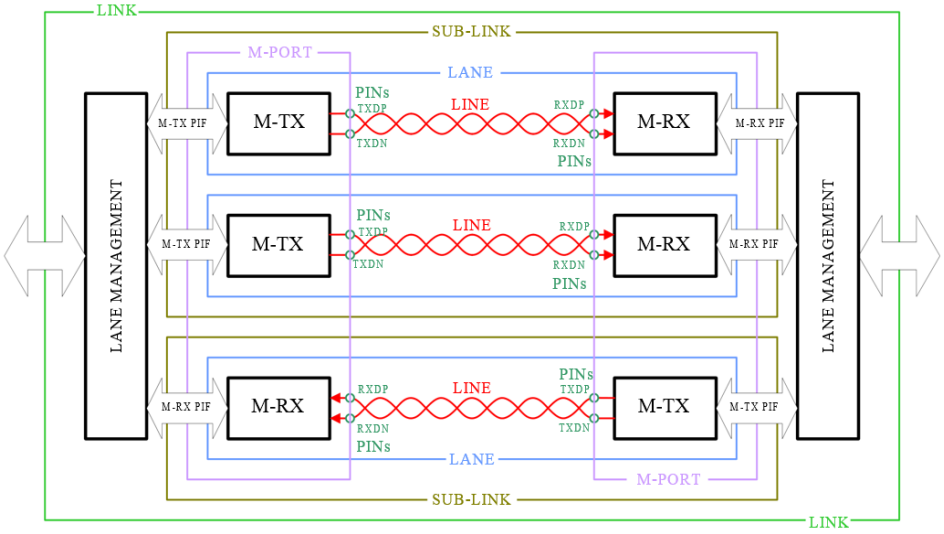

Figure 1 M-PHY Lane Example

The LINE consists of two differentially-routed wires connecting the LINE interface PINs of the M-TX and M-RX. Typically, these wires are transmission lines. Guidelines for LINE characteristics are described in ***Section 6***. For data transfer purposes, such a LINE might be considered as a black box with end-to-end signal transfer requirements defined at the PINs.

{- 115 }An interface based on M-PHY technology shall contain at least one LANE in each direction. There are no symmetry requirements from an M-PHY perspective for the number of LANEs in each direction.

All LANEs in the same direction within a LINK are denoted as a SUB-LINK. Two SUB-LINKs with opposite directions plus additional LANE management, which provides bidirectional data transport functionality agnostic to the actual LANE composition, is called a LINK. A set of M-TXs and M-RXs in a device that compose one interface port is denoted as an M-PORT.

This document specifies LANEs and their individual parts including M-TX, M-RX, interconnect. Furthermore, this specification sets some boundary conditions for M-TX and M-RX inside a single M-PORT,

Figure 8. The result of the automatic requirements recognition.

The requirements inherited from previous versions of the specification are highlighted as dark blue (T=0, “exact match”) and as purple (T=1, “good match”). The rest of the requirements are highlighted with colors from yellow (high probability of being a requirement, i.e., from 0.5 to 1.0) to orange (medium probability, i.e., from 0.2 to 0.5) and white (no highlighting, low probability, i.e., less than 0.2).

Based on this modification of the source specification, an engineer checks the quality of the recognized requirements, i.e., confirms or rejects the proposed output manually. Fig. 9 shows one of the processed requirements, which is related to the specification text and manually reviewed by a verification engineer. It can be further linked to test benches, checkers, models, etc.

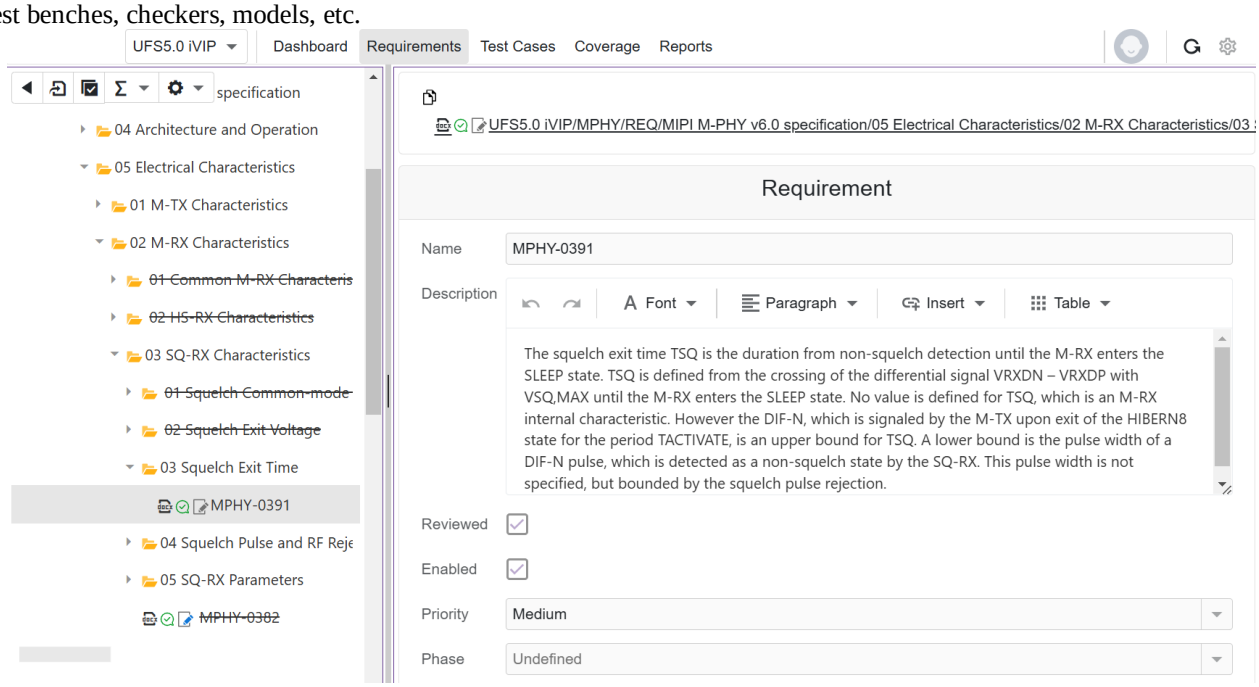

Figure 9. A requirement processed with the requirement management system.

At present an engineer needs to decide what the best way to cover the extracted requirement is. Based on the current state of natural language processing, it is possible to provide suggestions and some initial drafts of the test benches. However, the quality of the discussed systems highly depends on the quality and diversity of the labeled data. It would be hard to expect good generalization ability from the algorithm trained on one or two full-text specifications. Furthermore, manual processing of the hundred-paged technical documents requires a significant amount of time of the qualified verification engineer. The cost of this time may exceed the cost of some automation of the verification process. However, since the specification analysis is unavoidable during verification IP development process [3], the inputs generated by the machine learning model may improve the performance of the requirements identification, decisions on the choice of verification environment elements (test benches, test groups, checkers, models, etc.). Despite the difficulties, the automation of the technical document analysis flow has been tested on real projects utilizing M-PHY as a necessary component. Comparing to the previous verification experience, the LLM-guided process is faster and requires less time for on-boarding new members to the project.

Thus, the long-term target is to significantly increase the amount of labeled data, which should exceed tens of thousands data points extracted from different kinds of specifications and/or other types of technical documents. Some minor improvements can be achieved by tuning the LLM-specific parameters [24].

V. Conclusion

The combination of the sentence embedding based search algorithm and ensemble based classifier outperformed (90% vs 80%) previously published works [2-3] on a relatively small amount of data. There were no concerns that the proposed algorithm would suffer from overfitting, however, the bigger amount of various data extracted from technical documents would improve the quality significantly, as modern LLMs show great performance in text classification tasks. It is also worth noting that preparing multiple labeled documents requires significant efforts from experienced verification engineers and this is one of the challenges to be addressed.

The proposed approach requires the full text of the documents, as it analyzes the contexts of varying length. On the other hand, it allows recognizing the non-sequential multiple sentence requirements.

REFERENCES

[1] International Organization for Standards (ISO) "01.110 Technical product documentation. Including rules for preparation of user guides, manuals, product specifications, etc.", *available:* https://www.iso.org/ics/01.110/x/ (2024).

[2] Ivanov, V., et al. "Extracting software requirements from unstructured documents", *International Conference on Analysis of Images, Social Networks and Texts*, Springer International Publishing, pp. 17-29, 2021.

[3] Zalivaka, S., "Requirements Recognition For Verification IP Design Using Large Language Models", *DVCon USA*, pp. 163-171, 2024.

[4] Meqdadi, O., et al. "Mining software repositories for adaptive change commits using machine learning techniques", *Information and Software Technology*, vol. 109, pp. 80-91, 2019.

[5] Arora, C., et al. "Advancing requirements engineering through generative AI: Assessing the role of LLMs", *Generative AI for Effective Software Development*, Cham: Springer Nature Switzerland, pp. 129-148, 2024.

[6] Reimers, N., Gurevych, I., "Sentence-bert: Sentence embeddings using siamese bert-networks.", *arXiv preprint arXiv:1908.10084*, 2019.

[7] Ganaie, M. A., et al. "Ensemble deep learning: A review." *arXiv preprint arXiv: 2104.02395*, 2022.

[8] MIPI Alliance "MIPI M-PHY 4.1. Specification", *available:* https://www.mipi.org/specifications/m-phy (2024).

[9] Zhang, Q., et al. "Document Parsing Unveiled: Techniques, Challenges, and Prospects for Structured Information Extraction." *arXiv preprint arXiv:2410.21169*, 2024.

[10] Adobe Inc. "Convert PDF to Word", *available:* https://www.adobe.com/acrobat/online/pdf-to-word.html (2024).

[11] The Python Package Index (PyPI) "pdf2docx 0.5.8", *available:* https://pypi.org/project/pdf2docx/ (2024).

[12] Unstructured IO "Get your data RAG-ready", *available:* https://unstructured.io/ (2024).

[13] MIPI Alliance "MIPI M-PHY 6.0. Specification", *available:* https://www.mipi.org/specifications/m-phy (2024).

[14] Pan, J. J., et al. "Survey of vector database management systems", *The VLDB Journal*, pp. 1591-1615, 2024.

[15] Muflikhah, L., & Baharudin, B. "Document clustering using concept space and cosine similarity measurement", *International conference on computer technology and development*, IEEE, pp. 58-62, 2009.

[16] Liu, T., et al. "An investigation of practical approximate nearest neighbor algorithms", *Advances in neural information processing systems*, pp. 825-832, 2004.

[17] Reimers, N. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.", *arXiv preprint arXiv:1908.10084*, 2019.

[18] Nahal, S. "Make it Make Sense: Salesforce AI Research's SFR-Embedding, The Top Performing Text-Embedding Model.", *available:* https://blog.salesforceairesearch.com/sfr-top-performing-text-embedding-model (2024).

[19] Qdrant "High-Performance Vector Search at Scale", *available:* https://qdrant.tech (2024).

[20] Sentence Transformers "Semantic Search", *available:* https://sbert.net/examples/applications/semantic-search/README.html (2024).

[21] Po, D.K. "Similarity based information retrieval using Levenshtein distance algorithm", *Int. J. Adv. Sci. Res. Eng*, 6(04), pp.06-10, 2020

[22] Jiang, A. Q., et al. "Mistral 7B" *arXiv preprint arXiv:2310.06825*, 2023.

[23] NVIDIA Corporation "NVIDIA H100 Tensor Core GPU", *available:* https://www.nvidia.com/en-us/data-center/h100/ (2024).

[24] Jebali, M. S., et al. "LoFiT: Localized Fine-tuning on LLM Representations" *arXiv preprint arXiv:2406.01563*, 2024.