Applying Cluster Hypothesis to the Next Release Problem

Hemanth Gudaparthi
Governors State University
University Park, IL, USA
hgudaparthi@govst.edu

Nan Niu
University of Cincinnati
Cincinnati, OH, USA
nan.niu@uc.edu

Jianzhang Zhang
Hangzhou Normal University
Hangzhou, China
jianzhang.zhang@foxmail.com

Juha Savolainen
Danfoss Drives A/S
Graåten, Denmark
juha.savolainen@danfoss.com

***Abstract*—The aim of the next release problem (NRP) is to decide the most suitable subset of candidate requirements to include in the software system's upcoming version. To advance the automation degree of contemporary NRP solutions, we investigate in this paper the cluster hypothesis where we balance the to-be-released candidates with the already implemented features. Clustering the balanced set gives rise to a fully automatic NRP solution, using only the features' natural language descriptions and a project's release history. Our experiments on a total of 1,296 requirements from four real-world systems' 78 NRP instances show that k-means best fulfills the cluster hypothesis, and also outperforms the zero-shot learning capability of large language models (LLMs) in solving the NRP.**

***Index Terms*—strategic release planning, requirements prioritization, machine learning, large language models**

I. Introduction

The next release problem (NRP) is to determine which subset of the candidate requirements should be deployed and which other software requirements would not be delivered in the upcoming version [1]. Manual approaches like the 100-point method [2] and the analytic hierarchy process (AHP) [3] can be effective for ranking only a small number of requirements. A majority of automated support comes from the search-based software engineering (SBSE) paradigm [1], [4], [5]. SBSE represents NRP as an optimization problem (e.g., maximizing the requirements values perceived by the stakeholders while minimizing the estimated implementation costs [5]), and then applies meta- and hyper-heuristic search algorithms to find solutions. Manual work is often required to define the SBSE problem, e.g., estimating the implementation costs. Other NRP support is also semi-automated, including augmenting SBSE with interactive optimization [6], and case-based reasoning via the pairwise preferences elicited directly from stakeholders [7].

Surprisingly, contemporary methods have not solved the NRP fully automatically [8], [9]. In this paper, we explore an automatic approach based on the *cluster hypothesis*. In classification, the hypothesis states that if data points are in the same cluster, they are likely to be of the same class [10]. In the context of the NRP, we examine this hypothesis by considering both the candidate requirements to be released $\{r_1, r_2, \dots\}$ and the already implemented requirements $\{R_1, R_2, \dots\}$. According to the cluster hypothesis, if a resulting cluster consists of many R's and only a few r's, e.g., $\{R_1, R_2, R_3, R_4, R_5, r_1\}$, then we posit r_1 would share the class label of R_1–R_5 (i.e., "implemented"). Consequently, we would deem r_1 a part of the solution to the NRP. In contrast, if the resulting cluster has a similar number of r's and R's, or a greater number of r's than R's, then we regard those r's as being less likely to be part of the solution to the NRP.

Informed by the use of cluster hypothesis in requirements traceability [11], we experiment the same five clustering algorithms as in [11] to assess how effectively they solve the NRP: k-means, bisecting, single-link, complete-link, and average-link. We conduct our experiments with a total of 1,296 requirements from four real-world software systems, where the requirements release decisions are made automatically based on the resulting clusters. In order to compare the release decisions made on clusters, we derive a baseline by resorting to large language models (LLMs). Our baseline invokes a pre-trained LLM in a zero-shot manner to select a subset of requirements to release based on their associations with the already implemented requirements. For the baseline, we also prompt the LLM to summarize the reasons behind the associations to explain its generated release decisions.

The experimental results show that k-means (k=5) best implements the cluster hypothesis among the investigated algorithms, and that the k-means-based release decisions are more accurate than the LLM-based ones. The LLM baseline, on the other hand, offers new insights into requirements' interrelations. Such novelty can be exploited in future work to advance learning-based approaches to the NRP.

This paper makes two main contributions: (1) We rigorously investigate the cluster hypothesis, shedding light on a new angle for automatically solving the NRP; and (2) Using 1,296 requirements released by four software vendors, we quantitatively evaluate the effectiveness of different clustering algorithms in realizing the cluster hypothesis, and compare the clustering performances to an LLM baseline. The rest of the paper is organized as follows. Section II provides the background of the NRP. We then examine the cluster hypothesis experimentally in Section III, discuss related work in Section IV, and conclude the paper in Section V.

II. Next Release Problem (NRP)

Given a set of candidate requirements r=$\{r_1, r_2, \dots, r_n\}$, the NRP is concerned with selecting a subset of r, $r_{next} \subseteq r$,

2835-5776/24/$31.00 ©2024 IEEE
DOI 10.1109/IRI62200.2024.00060

to implement and deliver in the software's upcoming version. The unselected part, i.e., those requirements in $(r \setminus r_{next})$, will not be prioritized within the software's next release.

Requirements can be prioritized manually, e.g., the 100-point method lets stakeholders distribute their points individually and then aggregates the points for ranking the requirements [2]. Another manual approach is AHP [3] in which the decision maker compares every pair of requirements in terms of value and cost. These pairwise comparisons are then fed into the eigenvalue calculations in order to determine the requirements priorities. Karlsson *et al.* [12] evaluated several manual methods, and showed that AHP was the most promising though limited in scalability. For n requirements, AHP requires $\frac{n\times(n-1)}{2}$ comparisons in one dimension (e.g., value). Perini *et al.* [13] reported that AHP suffers scalability issues with requirements sets larger than about 50.

In pursuit of scalable solutions, researchers have explored automated support which we divide in three main categories: SBSE, constraint satisfaction, and machine learning (ML). The category of search-based approaches formulates the NRP as an optimization problem, and then employs computational search to find high quality but possibly suboptimal solutions. For example, Finkelstein *et al.* [4] investigated the trade-offs in various notions of fairness between multiple customers in the context of the NRP. One of their optimization objectives was to maximize the average absolute number of fulfilled requirements for all the customers whilst minimizing the standard deviation of the absolute number fulfilled requirements for each customer. Clearly, manual work is still needed for SBSE, e.g., defining which customer requires what requirements. Yet, once the search-based problem and the optimization objective are established, search techniques like the evolutionary algorithms could solve the NRP in an efficient manner [1], [4], [5].

The category of constraint satisfaction approaches encodes the NRP and its solution alternatives as Satisfiability and Optimization Modulo Theories (SMT/OMT) formulas, and then exploits automated solvers like OptiMathSAT [14] to identify solutions. Notably, Aydemir *et al.* [14] used a manually constructed goal model to elicit requirements interdependencies, e.g., "a child goal/requirement r_2 refines a parent goal/requirement r_1" means "r_1 requires r_2" [15], which can be further encoded into "$r_1 \implies r_2$" as a constraint. Carlshamre *et al.* [15] reported that four types of requirements interdependencies are of practical value in release planning, i.e., "AND" (R_1 requires R_2 to function, and R_2 requires R_1 to function), "REQUIRES" (R_1 requires R_2 to function, but not vice versa), "VALUE" (R_1 positively or negatively affects the value of R_2 for a customer), and "COST" (R_1 positively or negatively affects the cost of implementing R_2). Once the constraints or interdependencies are modeled, the SMT/OMT solvers can be viable for prioritizing hundreds of requirements [14], [16].

The category of existing ML approaches solves the NRP with the help of human interactions. Araújo *et al.* [6] proposed to learn the decision maker's subjective evaluation in an interactive way. Such an interactive optimization component was then integrated into the SBSE process. By gathering the subjective evaluation data, Araújo *et al.* [6] trained two ML models—linear regression and multilayer perceptron—to replace the human decision maker in the remainder of the search process. Perini *et al.* [7] developed a case-based reasoning algorithm, called CBRank, to prioritize requirements. CBRank's learning depends on the pairwise preferences elicited directly from stakeholders. Perini *et al.* [7] showed that CBRank was more cost-effective than AHP.

In summary, current approaches to the NRP either rely on complete manual work or require manual effort in problem formulation and training data preparation. To explore fully automatic NRP solutions, we next examine clustering—an unsupervised ML technique that does not require human's subjective evaluation [6], stakeholders' pairwise preferences [7], or any other labeled training data.

III. Cluster Hypothesis Applied to the NRP

A. Conceptualization

Cluster hypothesis, originated in the information retrieval (IR) literature [17], states that documents relevant to a user's query tend to be more similar to each other than to irrelevant documents. This hypothesis has been shown to be valid on multiple occasions [18]–[21]. In requirements engineering, Niu and Mahmoud [11] mapped the cluster hypothesis to the task of IR-based trace link generation. They tested five clustering algorithms: k-means, bisecting, single-link, complete-link, and average-link. Their experiments carried out on three datasets (iTrust, eTour, and CM-1) showed that single-link at the k=8 granularity level performed consistently well to fulfill the potential suggested by the cluster hypothesis.

Informed by this related work [11], we experiment the same five clustering algorithms in our current effort to solve the NRP. Different from the prior work where the clusters are formed in response to a given to-be-traced query-requirement [11], we apply the cluster hypothesis by formulating the NRP as a classification problem. Fig. 1 presents our conceptualization. Specifically, we consider the class label "implemented" and choose to assign a set of already released requirements R=$\{R_1, R_2, \ldots, R_n\}$ with the "implemented" label. Note that we want a built-in balance between the already labeled data points in R and the data points yet to be labeled. Because the to-be-labeled set is r=$\{r_1, r_2, \ldots, r_n\}$, the size of both R and r is n. With this formulation, to solve the NRP is to classify the candidate requirements in r to determine which requirements will end up belonging to the "implemented" class and which ones will not. We thus form r_{next} with the "implemented"-classified candidate requirements, and put the remaining candidate requirements into $(r \setminus r_{next})$.

Mixing the requirements in R and r provides a basis for performing clustering by using only the inherent natural language (NL) descriptions of the requirements [11]. Suppose the clustering results include **Cluster**$_1$–**Cluster**$_4$ depicted in Fig. 1. Then, according to the cluster hypothesis, the data points being clustered together are likely to be of the same class [10]. The r in **Cluster**$_1$ likely shares the "implemented"

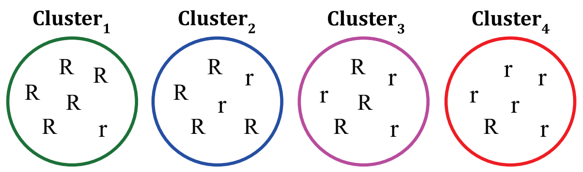


Fig. 1. Clustering all the requirements in $R \cup r$ where R=$\{R_1, R_2, \ldots, R_n\}$ includes a set of already implemented requirements and r=$\{r_1, r_2, \ldots, r_n\}$ contains the set of candidate requirements to be implemented potentially in the next release. The cluster hypothesis suggests that the data points being clustered together are likely to be of the same class [10]. Hence, the r in **Cluster**$_1$ likely shares the "implemented" label of the R's in **Cluster**$_1$. By contrast, the r's in **Cluster**$_4$ are less likely to be classified with the "implemented" label. The density ratio $\frac{|R|}{\text{cluster size}}$ can be used to rank the likelihood of r's being part of the NRP's solution r_{next}.

label of the R's in **Cluster**$_1$, and so do the r's in **Cluster**$_2$ though to a lesser degree. By contrast, the r's in **Cluster**$_3$ and **Cluster**$_4$ are less likely to be classified with the "implemented" label due to the smaller proportions of R's in those two resulting clusters. Therefore, we can use the density ratio of R per cluster to rank the r's for solving the NRP. In Fig. 1, for example, the R density of **Cluster**$_1$, **Cluster**$_2$, **Cluster**$_3$, and **Cluster**$_4$ is $\frac{5}{6}$, $\frac{4}{6}$, $\frac{3}{6}$, and $\frac{1}{6}$ respectively. As far as r_{next} is concerned, the r of **Cluster**$_1$ should be prioritized over the r's of **Cluster**$_2$, and so on.

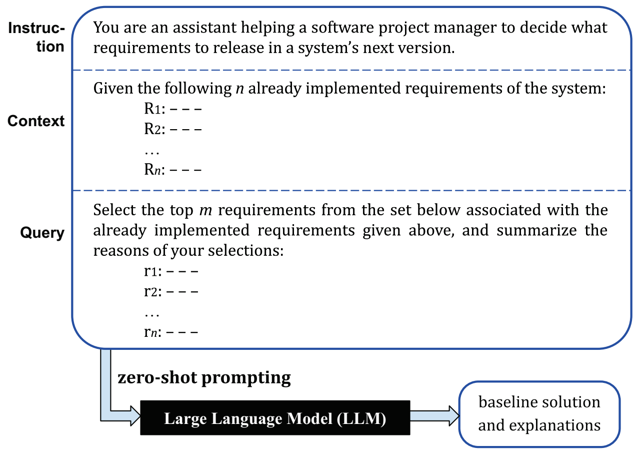
B. LLM Baseline

Pre-trained LLMs introduce a new paradigm in natural language understanding, inference, and generation of human-like responses. An LLM, such as T5 and the GPT series, is trained on massive amounts of raw text and can support a wide variety of natural language processing (NLP) tasks with prompt engineering [22], [23]. By simply describing NL prompts, an LLM can be invoked to perform specific tasks without requiring additional training or hard coding. We take advantage of the zero-shot learning of an LLM, and design a baseline shown in Fig. 2 to automatically solve the NRP.

We follow the literature on applying LLMs in zero-shot settings [24]–[26] to structure the baseline's prompt in three parts: instruction, context, and query. As illustrated in Fig. 2, the LLM is first instructed to play the role of an assistant to help make a software project's next release decisions. The context then defines the n already implemented requirements (R) in their NL descriptions. Finally, the query formulates the task at hand: selecting m requirements from r based on their associations with R and explaining the selections. The prompt depicted in Fig. 2 is sent to an LLM whose responses of the m selected r's as well as the selection explanations are recorded as the baseline's outputs.

C. Experimental Design

The objective of our experimental evaluation is to assess the performance of clustering and the LLM baseline. Quantitatively, we measure the accuracy of the NRP solutions generated by k-means, bisecting, single-link, complete-link, average-link, and LLM. As the selected set of requirements, m, is the same as the size of the ground-truth r_{next}, recall, precision, and F-measure are exactly the same. Thus, we report F-measure in order to quantify accuracy. Qualitatively, we also analyze the LLM baseline's explanation summaries, especially for those NRP solutions generated correctly by the LLM but incorrectly by clustering.


Instruc-tion
Context
Query
zero-shot prompting
Large Language Model (LLM)
baseline solution and explanations

You are an assistant helping a software project manager to decide what requirements to release in a system's next version.

Given the following *n* already implemented requirements of the system:
R1: – – –
R2: – – –
…
R*n*: – – –

Select the top *m* requirements from the set below associated with the already implemented requirements given above, and summarize the reasons of your selections:
r1: – – –
r2: – – –
…
r*n*: – – –

Fig. 2. Using an LLM in a zero-shot manner as a baseline to decide the NRP's solution with summarized explanations.

We choose to experiment with four subject systems for a couple of reasons. First, they are real-world software systems evolving constantly to meet the requirements of large user bases. Second, these systems' release notes are publicly available [27]–[30], allowing the ground-truth r_{next} to be authoritatively established and thereby eliminating the need for manual labeling. Despite of the various application domains shown in Table I, our data collection period for all the four subject systems is from October 2021 to October 2023. Over this period of time, Zoom, Webex, Microsoft 365, and Discord released 557, 313, 272, and 154 requirements respectively, as presented in Table I.

Our operational unit is an *NRP instance*. Table II illustrates one such unit in the Zoom dataset. This instance has 38 total requirements: 14 were released in January 2022, 5 in February 2022, and 19 in December 2021. The combination of 19 January and February data forms the candidate requirements set r, demanding a balanced set of "implemented" requirements R (i.e., $|R|$=$|r|$=n=19). In this NRP instance, the 14 requirements released in January 2022 serve as the ground-truth r_{next} (i.e., $|r_{next}|$=m=14).

We plot the size information of our datasets' 78 NRP instances in Fig. 3. From this figure, we can see that most of the large instances belong to Zoom whereas the Webex instances exhibit a relatively stable $\frac{m}{n}$ ratio. In terms of the NL size, the last column of Table I shows the average number of words per requirement in the datasets. Notably, Webex requirements are longer and hence may contain more details than the requirements of the other three software systems.

Our LLM baseline exploits OpenAI's gpt-3.5-turbo, since

TABLE I
SUBJECT SYSTEMS AND THEIR DATASET CHARACTERISTICS: ALL THE COLLECTED REQUIREMENTS ARE FROM OCTOBER 2021 TO OCTOBER 2023

system [source of req.s]	application domain	total # of NRP instances	total # of req.s	average # of words per req.
Zoom [27]	video conferencing	23	557	41.3
Webex [28]	video conferencing	24	313	52.4
Microsoft 365 [29]	online Microsoft Office suite	24	272	28.2
Discord [30]	messaging & VoIP social platform	7	154	32.7

TABLE II
EXCERPTS FROM ONE NRP INSTANCE OF THE ZOOM DATASET: THE TOP 14 REQUIREMENTS WERE RELEASED IN JANUARY 2022 GIVING RISE TO THE GROUND-TRUTH r_{next} WITH m=$|r_{next}|$=14; THE MIDDLE 5 REQUIREMENTS WERE RELEASED IN FEBRUARY 2022 WHICH, COMBINED WITH THE TOP 14 REQUIREMENTS, DEFINE THE SET OF CANDIDATE REQUIREMENTS r WITH n=$|r|$=19; THE BOTTOM 19 REQUIREMENTS FORM THE "IMPLEMENTED"-LABELED R CONSISTING OF ALL THE 19 REQUIREMENTS RELEASED IN DECEMBER 2021

#	Requirements Description	Release Time
1	Enabled by default, this controls the state of the Use hardware acceleration to optimize video sharing option, found under the Advanced section of the Share Screen settings tab.	Jan 2022
...	... ...	...
14	Users can select and use any virtual background before or during any Zoom meeting. However, after the meeting is over, the user's background will be automatically changed to a default virtual background chosen by the admin.	Jan 2022
1	Language interpreters can listen to either the main audio or another translator's audio, which can be useful when a translator doesn't know the main language being spoken, but knows another language the audio is being translated to.	Feb 2022
...	... ...	...
5	In addition to notifications when an App is pre-approved for use, users will also get notifications when Pre-approval has been disabled, allowing users to install any App on the Marketplace.	Feb 2022
1	Our emoji suite has been upgraded to provide almost 900 more emojis for use in in-meeting chat and Zoom chat. These new emojis include various skin tones and are Unicode 13.1 compatible.	Dec 2021
...	... ...	...
19	Users with the Power Pack add-on can search for contacts and transfer calls directly from the Desktop Power User experience, without having to open the main Zoom client window.	Dec 2021

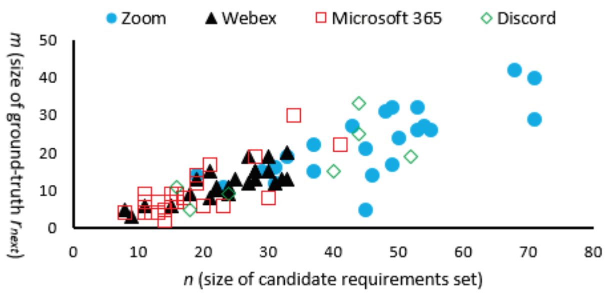


Fig. 3. Size information of all the 78 NRP instances: x-axis represents n (i.e., the number of candidate requirements in each NRP instance) whereas y-axis represents m (i.e., the number of ground-truth requirements in that instance).

it is among the state-of-the-art models and has also been shown to be promising in supporting requirements engineering downstream tasks [26], [31]. Because the token limit for gpt-3.5-turbo is 4,096, we evenly allocate the same number of tokens for each requirement in a given NRP instance. In case that a requirement is longer than the allocated tokens, we truncate the requirement by preserving the beginning of its NL description to avoid exceeding the LLM's length limit. Our truncation operation is informed by the recent work of Gao and his colleagues [32]. We set the temperature to 0 to obtain the deterministic output from the LLM. Since LLM uses embeddings, we also prepare a comparable data representation for clustering. In particular, we invoke OpenAI's GPT-3 encoder, and then use cosine similarity weighted by tf-idf [11] to run k-means, bisecting, single-link, complete-link, and average-link on the requirements in each NRP instance.

D. Results and Analysis

The LLM and clustering results are shown in Fig. 4. Because k is irrelevant to the LLM baseline, the figure uses a horizontal line to depict the baseline's F-measure averaged over all the NRP instances in a dataset. For LLM and clustering, since the standard deviations of F-measure values are small for all the datasets, we have not shown the standard deviations in the plots of Fig. 4 to reduce clutter.

A general trend of all the five clustering algorithms in Fig. 4 is that the NRP solutions' quality becomes worse when k is too small or too large. We believe that a small k could lead many false positives to be included in the NRP solution, while a large k would result in some similar $\frac{|R|}{\text{cluster size}}$ ratios thereby reducing the distinguishing power between the clusters. A surprising observation from Fig. 4 is that not all the clusterings at their optimal k values outperform the LLM baseline. Notably, LLM achieves better NRP solutions than single-link in Webex, as well as bisecting in Discord.

In most cases, clusterings' performances at their optimal granularity levels are better than the LLM baseline. This confirms that the cluster hypothesis generally holds in the NRP context. In particular, the average F-measure of Webex hits a maximum when k-means is used to produce 4 clusters. For Zoom and Discord, k-means achieves the best performance at 4–5 clusters. For Microsoft 365, a local maximum of average F-measure is obtained when 6 clusters are generated by either complete-link or k-means. From our analysis, k-means turns out to be the best candidate for realizing the cluster hypothesis in solving the NRP. The optimal clustering granularity of k-means is found to be k=5 across the four datasets. Applying

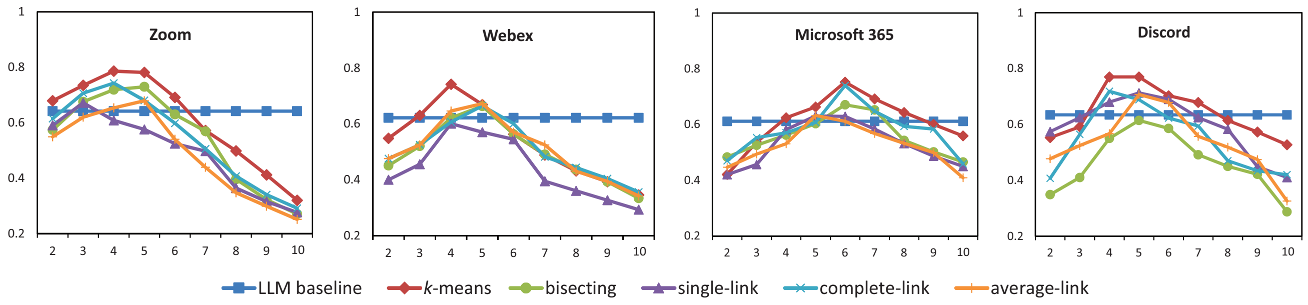


Fig. 4. Solution quality for the four datasets: x-axis represents the k values (i.e., the number of resulting clusters), and y-axis represents the average F-measure values of all the NRP instances in the dataset.

the nonparametric Mann-Whitney-Wilcoxon test on all our datasets' 78 NRP instances shows that the F-measure values of k-means (k=5) are significantly higher than those of the LLM baseline (p=0.039).

Our findings of k-means with k=5 differ from the previous observations that single-link with k=8 performed the best in automated requirements traceability [11]. One plausible explanation rests on using only one to-be-traced query-requirement as the basis to perform the clustering [11]; yet, in our current work, a balanced number of "implemented"-labeled requirements are deployed in the clustering process. Consequently, more meaningful neighborhoods might be formed with relatively diverse centroids in our work. In contrast, little impact on diversity is made by the one and only one tracing source, leaving the agglomerative hierarchy a viable structure for retrieving the candidate traceability links [11].

As for the differences of the optimal clustering granularity, a possible reason might lie in the size of to-be-clustered items. In our study, each NRP instance has an average of 60.5 requirements to be clustered, which is less than the 187.3 to-be-clustered artifacts averaged across the three traceability datasets [11]. Thus, having too many resulting clusters (e.g., k=8) could weaken their distinguishing powers, as mentioned earlier. Based on the quantitative results, we conclude that the cluster hypothesis generally holds in the context of the NRP, and that k-means, at the k=5 clustering granularity, best implements the cluster hypothesis among the five algorithms investigated in our research.

Our qualitative analysis seeks to identify complementary aspects that the LLM could offer to clustering. To that end, we first choose the NRP instances whose LLM solutions are more accurate than the best clustering solutions. We then manually inspect the LLM explanations for these NRP instances. The final results of our analysis cover not only the "AND", "REQUIRES", "VALUE", and "COST" interdependencies identified by Carlshamre *et al.* [15], but also new relations like "shared implementation". Leveraging these relations to enhance the accuracy and explainability of learning-based NRP solutions is part of our future work.

E. Threats to Validity

The construct validity can be affected by the LLM baseline's prompt, which we devise via an instruction-context-query structure. While prompt patterns begin to emerge [33] and are worth experimenting with, we believe the prompt of Fig. 2 conveys our intent to solve the NRP by associating candidate requirements with the already implemented ones.

The use of LLM also poses a couple of internal validity threats. One concerns data leakage where the LLM has been trained on our experimental data so that it merely memorizes the results instead of predicting them. We mitigate this threat by collecting the requirements data released from October 2021, as gpt-3.5-turbo used in our baseline is trained with the data up to September 2021 [34]. The other threat relates to gpt-3.5-turbo's token limit. Our even-allocation and truncation operations follow Gao *et al.*'s work [32]; however, uneven allocations or other length-handling methods (e.g., removing older texts [35]) may lead to different baseline results.

Threats to the external validity include our investigations of only five clustering algorithms on four requirements datasets. In light of other algorithms and datasets, our results may not generalize. In addition, our baseline uses only gpt-3.5-turbo in a zero-shot fashion, though the backbone can be replaced at will as long as the LLM or relevant API is accessible.

IV. Related Work

We discuss three research thrusts related to our work: strategic release planning, requirements clustering, and LLMs in requirements engineering. Svahnberg *et al.* [9] surveyed 24 release planning models and identified 11 factors used in the models. Ameller *et al.* [8] updated Svahnberg *et al.*'s survey with an additional 17 models and confirmed these 11 factors. The most influential and commonly used factors include stakeholder values, effort estimates, and requirements interdependencies [8], [9]. Our work leverages the cluster hypothesis to group candidate requirements with what has already been implemented.

Clustering requirements has been shown to support many tasks: constructing core assets for software product lines [36], provoking exploratory and combinational creativities [37], and so on. Closely related to our work is the application of cluster hypothesis to enhance automated traceability [11]. Although our work investigates the same five clustering algorithms as in [11], a fundamental difference is our inclusion of the "implemented"-labeled requirements to balance the to-be-released candidates.

The emergence of LLMs attracts immediate attention among requirements engineering researchers. For example, Dahiya *et al.* [38] used ChatGPT to predict requirements testability, and Zhang *et al.* [23] evaluated ChatGPT on requirements information retrieval tasks. Our work's zero-shot learning thus explicitly prompts the LLM to offer the explanations for its decisions.

V. Concluding Remarks

Deciding precisely what to build, according to Brooks [39], is the hardest part of software engineering. The NRP embodies this difficulty, and practically impacts what software functionalities are delivered to end users and when. In this paper, we solve the NRP by rigorously investigating the cluster hypothesis in an unsupervised fashion.

Our future work includes testing more requirements data from diverse applications and domains. We are also interested in exploiting requirements interdependencies [40] and user reviews [41] to improve the accuracy and explainability of learning-based NRP solutions.

Acknowledgment

This work is partially supported by Engineering Research Center of Mobile Health Management System of Chinese Ministry of Education (Grant No. 2022GCZXGH04).

References

[1] A. J. Bagnall *et al.*, "The next release problem," *Information & Software Technology*, vol. 43, no. 14, pp. 883–890, December 2001.

[2] D. Leffingwell and D. Widrig, *Managing Software Requirements: A Use Case Approach*. Addison-Wesley, 2003.

[3] J. Karlsson and K. Ryan, "A cost-value approach for prioritizing requirements," *IEEE Software*, vol. 14, no. 5, pp. 67–74, Sept/Oct 1997.

[4] A. Finkelstein *et al.*, "A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making," *Req.s Eng.*, vol. 14, no. 4, pp. 231–245, December 2009.

[5] Y. Zhang *et al.*, "An empirical study of meta- and hyper-heuristic search for multi-objective release planning," *ACM Transactions on Software Engineering and Methodology*, vol. 27, no. 1, pp. 3:1–3:32, June 2018.

[6] A. A. Araújo *et al.*, "An architecture based on interactive optimization and machine learning applied to the next release problem," *Automated Software Engineering*, vol. 24, no. 3, pp. 623–671, September 2017.

[7] A. Perini *et al.*, "A machine learning approach to software requirements prioritization," *IEEE Transactions on Software Engineering*, vol. 39, no. 4, pp. 445–461, April 2013.

[8] D. Ameller *et al.*, "A survey on software release planning models," in *Proc. of PROFES*, Trondheim, Norway, November 2016, pp. 48–65.

[9] M. Svahnberg *et al.*, "A systematic review on strategic release planning models," *Information & Software Technology*, vol. 52, no. 3, pp. 237–248, March 2010.

[10] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning*. The MIT Press, 2006.

[11] N. Niu and A. Mahmoud, "Enhancing candidate link generation for requirements tracing: The cluster hypothesis revisited," in *Proc. of RE*, Chicago, IL, USA, September 2012, pp. 81–90.

[12] J. Karlsson *et al.*, "An evaluation of methods for prioritizing software requirements," *Information & Software Technology*, vol. 39, no. 14-15, pp. 939–947, 1998.

[13] A. Perini *et al.*, "An empirical study to compare the accuracy of ahp and cbranking techniques for requirements prioritization," in *Proc. of CERE*, New Delhi, India, October 2007, pp. 23–35.

[14] F. B. Aydemir *et al.*, "The next release problem revisited: A new avenue for goal models," in *Proc. of RE*, Banff, Canada, August 2018, pp. 5–16.

[15] P. Carlshamre *et al.*, "An industrial survey of requirements interdependencies in software product release planning," in *Proc. of RE*, Toronto, Canada, August 2001, pp. 84–91.

[16] F. Palma *et al.*, "Using an SMT solver for interactive requirements prioritization," in *Proc. of ESEC/FSE*, Szeged, Hungary, September 2011, pp. 48–58.

[17] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[18] O. Zamir and O. Etzioni, "Grouper: A dynamic clustering interface to Web search results," *Computer Networks*, vol. 31, no. 11-16, pp. 1361–1374, May 1999.

[19] F. Crestani and S. Wu, "Testing the cluster hypothesis in distributed information retrieval," *Information Processing and Management*, vol. 42, no. 5, pp. 1137–1150, 2006.

[20] A. Mahmoud and N. Niu, "TraCter: A tool for candidate traceability link clustering," in *Proc. of RE*, Trento, Italy, August-September 2011, pp. 335–336.

[21] N. Niu *et al.*, "A clustering-based approach to enriching code foraging environment," *IEEE Transactions on Cybernetics*, vol. 46, no. 9, pp. 1962–1973, September 2016.

[22] P. Liu *et al.*, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Computing Surveys*, vol. 55, no. 9, pp. 195:1–195:35, September 2023.

[23] J. Zhang, *et al.*, "Empirical evaluation of ChatGPT on requirements information retrieval under zero-shot setting," in *Proc. of ICNGN*, Hangzhou, China, November 2023.

[24] Y. Lyu *et al.*, "CHRONOS: Time-aware zero-shot identification of libraries from vulnerability reports," in *Proc. of ICSE*, Melbourne, Australia, May 2023, pp. 1033–1045.

[25] J. Li *et al.*, "ZC^3: Zero-shot cross-language code clone detection," *CoRR*, September 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2308.13754

[26] A. Fantechi *et al.*, "Inconsistency detection in natural language requirements using ChatGPT: A preliminary evaluation," in *Proc. of RE*, Hannover, Germany, September 2023, pp. 335–340.

[27] Zoom, "Release Notes for Windows," https://support.zoom.us/hc/en-us/articles/201361953-Release-notes-for-Windows, 2023, Last accessed: June 11, 2024.

[28] Cisco, "Webex App: What's New," https://help.webex.com/en-us/article/8dmbcr/Webex-App-—-What's-New, 2023, Last accessed: June 11, 2024.

[29] Microsoft, "Microsoft 365 Apps for Windows: Archived Release Notes," https://learn.microsoft.com/en-us/officeupdates/monthly-channel-archived, 2023, Last accessed: June 11, 2024.

[30] Discord, "Change Log," https://discord.com/developers/docs/change-log, 2023, Last accessed: June 11, 2024.

[31] A. D. Rodriguez *et al.*, "Prompts matter: Insights and strategies for prompt engineering in automated software traceability," in *Proc. of SST*, Hannover, Germany, September 2023, pp. 455–464.

[32] S. Gao *et al.*, "What makes good in-context demonstrations for code intelligence tasks with LLMs?" in *Proc. of ASE*, Kirchberg, Luxembourg, September 2023, pp. 761–773.

[33] J. White *et al.*, "A prompt pattern catalog to enhance prompt engineering with ChatGPT," *CoRR*, February 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2302.11382

[34] OpenAI, "GPT models," 2023, Last accessed: June 11, 2024. [Online]. Available: https://platform.openai.com/docs/models/gpt-3-5

[35] M. Lempinen *et al.*, "Chatbot for assessing system security with OpenAI GPT-3.5," Last accessed: June 11, 2024. [Online]. Available: http://jultika.oulu.fi/files/nbnfioulu-202306292789.pdf

[36] I. R.-Berger and M. Kemelman, "Extracting core requirements for software product lines," *Req.s Eng.*, vol. 25, no. 1, pp. 47–65, 2020.

[37] T. Bhowmik *et al.*, "Automated support for combinational creativity in requirements engineering," in *Proc. of RE*, Karlskrona, Sweden, August 2014, pp. 243–252.

[38] M. Dahiya *et al.*, "Leveraging chatgpt to predict requirements testability with differential in-context learning," in *Proc. of IRI*, San Jose, CA, USA, August 2024.

[39] F. P. Brooks, Jr., *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley, 1975.

[40] W. Wang *et al.*, "Detecting software security vulnerabilities via requirements dependency analysis," *IEEE Transactions on Software Engineering*, vol. 48, no. 5, pp. 1665–1675, May 2022.

[41] J. Zhang *et al.*, "Software feature refinement prioritization based on online user review mining," *Information & Software Technology*, vol. 108, pp. 30–34, April 2019.