Screening Articles for Systematic Reviews with ChatGPT

Eugene Syriani[1,*], Istvan David[2], Gauransh Kumar[1]

[1] *DIRO, Université de Montréal, Canada*
[2] *McMaster University, Canada*

Abstract

Systematic reviews (SRs) provide valuable evidence for guiding new research directions. However, the manual effort involved in selecting articles for inclusion in an SR is error-prone and time-consuming. While screening articles has traditionally been considered challenging to automate, the advent of large language models offers new possibilities. In this paper, we discuss the effect of using ChatGPT on the SR process. In particular, we investigate the effectiveness of different prompt strategies for automating the article screening process using five real SR datasets. Our results show that ChatGPT can reach up to 82% accuracy. The best performing prompts specify exclusion criteria and avoid negative shots. However, prompts should be adapted to different corpus characteristics.

Keywords: generative AI, GPT, empirical research, large language model, literature review, mapping study, screening

1. Introduction

Systematic reviews (SRs) are a scholarly method for synthesizing and organizing knowledge from primary studies within a specific research field. These reviews document the state of the art and provide a foundation for scholars to guide their research toward impactful directions. In the field of software engineering, the number of published systematic reviews has been steadily increasing [52]. Despite their importance, conducting SRs can be challenging and labor-intensive. Among the various phases of an SR, screening—i.e., the selection of relevant articles and their subsequent comprehensive reading—has been reported as the most time-consuming [12]. This phase is also a primary source of errors in building the article corpus due to its manual nature, introducing threats to internal validity such as fatigue, attrition, and researcher biases [37]. Experts have identified article screening as one of the most significant barriers [1] to efficient SR processes.

Researchers commonly employ strategies to mitigate errors, such as assigning multiple reviewers to each article, introducing a validation step by a senior reviewer, and, to alleviate the increased workload, restricting selection to titles and abstracts [30]. While these practices help address some of the challenges associated with manual screening, they do not scale well with large article corpora, ultimately making human performance a bottleneck in the SR process. Given that working with corpora of thousands of articles is not uncommon, the screening phase remains a critical problem in conducting SRs.

In response to scalability challenges, supporting the human in the review process has been a topic of particular interest in software engineering [17, 27, 53]. However, state-of-the-art SR tools do not fully automate the screening phase, as it has traditionally been considered challenging to achieve [17]. Instead, they guide human reviewers by ranking articles based on the likelihood of inclusion, eventually reaching a point after which articles are not screened. Unfortunately, determining this stopping point remains a challenge, as evidenced by the significant variations across different reviews [38].

With the advent of large language models (LLMs), such as GPT [20], the automation of screening activities has become feasible. LLMs are

*Corresponding author
Email address: syriani@iro.umontreal.ca (Eugene Syriani)

AI models pre-trained on vast amounts of textual data, enabling them to capture extensive knowledge that can be utilized, e.g., for classifying articles within a corpus. At the time of writing, OpenAI's GPT family of models[1] constitutes the largest LLMs. In our experiments, we use ChatGPT as the representative LLM, widely employed in various domains beyond software engineering, including health care [11], climate research [10], and creative writing [41].

This study aims to investigate the effectiveness of ChatGPT for automating article screening and its effect on the SR process. Our previous work [52] reveals that ChatGPT performs on par with classifiers traditionally used in SR automation without the burden of training and tuning. In this study, we perform a similar evaluation by exploring variants of a new prompt template, and discuss their effect on the SR process. Given the binary classification problem of deciding whether to include or exclude an article in an SR, we assess the classification performance of ChatGPT on five corpora. To this end, we pose the following research questions.

RQ1. *What is the effect of different prompting techniques on the classification performance of ChatGPT?* We investigate to what extent ChatGPT correctly decides about selecting articles for an SR. We evaluate combinations of shot learning [36] and chain-of-thought prompting [60].

RQ2. *How does the classification performance of ChatGPT compare to that of traditional classifiers trained specifically for screening the corpus of an SR?* We investigate if the decisions of ChatGPT are comparable with common classifiers used in SR.

RQ3. *How do the characteristics of the corpus impact the decisions made by ChatGPT?* We investigate whether the classification performance of ChatGPT generalizes across the five datasets with reasonable sensitivity. Although we investigate *a posteriori* characteristics, e.g., inclusion rate, these characteristics still shed light on the strengths and weaknesses of ChatGPT.

Our results indicate that zero-shot learning tends to achieve better classification performance than few-shot learning in screening articles. We also show that it is possible to engineer prompts that tend to work generally well on SR corpora of different characteristics, but some characteristics of corpora might require different prompt variants. To this end, we contribute a prompt template that can be instantiated in various ways for different SRs. The contributions of this work are the following.

- a general prompt template that can be instantiated for different SRs;
- a curated list of real SR datasets;
- a systematic evaluation of different prompt strategies on the datasets.
- recommendations for next-generation SR tools relying on LLMs;
- recommendations to adapt SR guidelines and processes accordingly.

The replication package containing the data and analysis scripts is publicly available for independent verification and replication.[2]

The rest of this paper is structured as follows. In Sec. 2, we review related works. In Sec. 3, we discuss the experimental method. In Sec. 4, we elaborate on our prompt engineering strategy. In Sec. 5, we discuss the threats to validity. In Sec. 6, we present the results and address the research questions. In Sec. 7, we discuss the results and their implications on SR processes. Finally, in Sec. 8, we draw the conclusions and identify future work.

2. Related work

In the past two decades, researchers from various domains have explored different approaches to reduce the screening effort in SRs.

Some of these efforts involve improving well-established protocols. For instance, Kosar et al. [31] propose a variation of the Kitchenham protocol [30] by reducing the number of articles to screen within a certain margin of error.

Rozanc and Mernik [49] propose automating the screening task of systematic mapping studies through text analysis, employing text statistic analysis to count the occurrence of important works, and enabling the user to define rules to decide how the screening process should interpret these occurrences. In particular, users state which words are required, have a positive or negative effect towards

[1] https://openai.com/chatgpt/

[2] Available in the replication package https://doi.org/10.5281/zenodo.10257742

the decision outcome. The approach is supported by a tool, which—as opposed to our technique—needs to be configured iteratively by the human for each study.

Other efforts focus on automation through machine learning techniques. By reviewing 41 relevant studies, van Dinter et al. [55] conclude that the most commonly used machine learning models for SR automation are Support Vector Machines (SVM) [26] and Bayesian Networks, with Bag of Words and Term Frequency-Inverse Document Frequency (TF-IDF) being the most popular natural language processing representation techniques. Their review highlights that no study has yet investigated the use of deep neural network models specifically for the screening phase of SRs. One of the main challenges the authors identified is the issue of imbalanced datasets, with many excluded articles dominating the distribution. Such skewed distributions often lead classifiers to maximize accuracy for only one of these two classes of articles. Additionally, machine learning requires manual training and fine-tuning of features for every SR, limiting its practical applicability.

2.1. A posteriori reduction of screening work

Most screening automation techniques require a labeled dataset for training and effort savings can only be assessed after the screening has been conducted manually, defeating the purpose of automation.

Martinez et al. [39] propose a technique to prioritize corpora by ranking articles from most to least likely to be included. They use a generic text retrieval search engine and a classifier that reranks articles. Cohen et al. [15] train a boosted perceptron-based classifier to predict new articles to be added to SRs on drug class efficacy for the treatment of disease. Matwin et al. [40] use factorized Complement Naive Bayes classifier to maximize recall. Ji et al. [24] utilize information retrieval to establish relationships and ontology-based semantics of the articles.

Watanabe et al. [59] evaluated the efficacy of using SVM with varing scales of χ^2 features on eight systematic review updates. They report that this text classifier achieve up to perfect recall in some of the datasets. However, the classifier was trained with the data from the original SRs.

As opposed to these techniques, the ChatGPT-based automation we evaluate aims to avoid *a posteriori* decision making in screening.

2.2. Ranking articles by active learning

Active learning is a machine learning technique wherein the learning agent autonomously selects training data for learning and queries an oracle to label previously unlabeled instances [50]. Active learning is often used for ranking articles by relevance for screening. By prioritizing articles that are more likely to be included, reviewers can make inclusion decisions at a higher pace. Conversely, prioritizing articles the algorithm is doubtful about enables faster learning and allows ranking the remaining articles with higher confidence.

Marshall and Wallace [38] describe an active learning variant based on certainty, continuously trained on manually screened articles. It predicts the probability of relevance for all unseen articles and then reorders them, presenting the most relevant ones to the reviewer first. When uncertainty sampling is used, papers predicted with the least certainty are presented first to improve the model's accuracy more efficiently.

The following SR tools support this process: Abstrackr, Colandr, EPPI reviewer, SWIFT-Review, and RobotAnalyst. The latter two also group articles by similar topics.

Abstrackr [56] employs active learning-based screening with an SVM model. Abstrackr did not perform well in our experiments on a software engineering corpus. A possible explanation is that Abstrackr is tuned for articles in medicine, where abstracts are better structured.

ASReview [54] lets the user choose between different machine learning models, including Naive Bayes, SVM, Logistic Regression, and Random Forest. They offer various feature extraction models as well: embedding with Inverse Document Frequency, TF-IDF, Sentence BERT [48], Doc2Vec [35], and LSTM networks.

Ferdinands et al. [18] show that a Naive Bayes classifier with TF-IDF outperforms SVM for their four datasets. However, they note that dataset characteristics influence classification performance.

Despite the apt idea, active learning-based screening is seldom encountered due to its limited generalizability [28] and efficiency [43]. ChatGPT-based automation improves generalizability over data sets, particularly by a uniform prompt and assessment methods.

2.3. Large language models and ChatGPT

The recent advances in LLMs, especially popularized with the infatuated AI-driven chatbot Chat-

GPT has instigated a revolutionary paradigm shift in software engineering and other disciplines [67]. ChatGPT is a Chatbot Generative Pre-trained Transformer that employs an auto-regressive language model pre-trained on large datasets with billions of tokens from CommonCrawl, Wikipedia, and other publicly available text sources. ChatGPT relies on a deep neural network with a transformer architecture to estimate the conditional probability of a sequence of tokens given a context.

Pre-training renders the model applicable to a wide array of problems with only minor effort (e.g., few-shot learning) required for tuning the model. Such a minor effort is few-shot learning, in which a few input-output pairs are provided as examples to the LLM [66]. LLMs improve over the techniques discussed in Sec. 2.2 thanks to the negligible effort of adopting the technique to the problem at hand. Thus, using ChatGPT to reduce the reviewer's workload in screening articles without training it specifically on the corpus of the SR seems to be a promising direction.

Fine-tuning LLMs is achieved in two ways: hyperparameter tuning and prompt engineering. On the one hand, ChatGPT's main hyperparameters are `temperature`, controlling the diversity of its response, and `max_tokens` to restrict the length of the output. Other hyperparameters can also be controlled, such as `top_p` sampling, which deals with the randomness in the model output by selecting the top tokens whose cumulative probability exceeds a defined threshold $p \in [0, 1]$. Another hyperparameter `stop_token` can also be used to control the output length of the LLM.

On the other hand, the prompt can be engineered in different ways. Different wording styles of the prompt affect ChatGPT's response [61]. The presence of shots in the prompt is also a factor [36]. A zero-shot prompt is when the user explains the task without providing any labeled examples. Few or N-shots prompt provide N labeled solutions to the task in the prompt. One can also include reasoning steps along with instructions in chain-of-thought prompts.

2.4. Application of ChatGPT in SRs

Recent (not peer-reviewed) reports discuss early results on using LLMs in SRs.

Wang et al. [57] study the use of ChatGPT to automatically formulate search queries to retrieve articles. They experiment on a dataset from PubMed for a medical SR with 70 titles and abstracts from a standard test benchmark [2] and obtain high precision but low recall. They observe variations in the generated queries from the same prompt and remain inconclusive about the effectiveness of ChatGPT in generating SR search queries.

Waseem et al. [58] propose a method for Human-Bot collaboration in conducting SRs. However, screening is still manual and ChatGPT is used as a decision support system not to replace human effort.

Wilkins [62] evaluates conversations with GPT4 to screen articles in six clinical scoping reviews. They report similar recall and specificity to what we report in [52]. However, they require specific prompts tailored to each SR and have a much higher token count which, they admit, may hinder its usage in SR tools.

Khraisha et al. [29] compare GPT's performance to human performance in both the screening and data extraction phases. Their findings indicate substantial potential to replace humans with LLMs in SR, however, their prompts score lower compared to ours and their prompt engineering method is not disclosed.

Hasan et al. [22] report on their attempt to use GPT4 when assessing the risk of bias in SRs conducted in Cochrane. The token count limit in the prompt context limits forces them to extract parts of the full text of articles to submit to the LLM. They favor an interative prompt strategy requiring a significant amount of user intervention to obtain the best performing prompt. Instead, our approach assesses the potential of using LLMs in an automayed way thanks to predefined prompt templates.

Our earlier report [52] determined the predictive performance of ChatGPT to be on par with that of traditional classifiers. However, the report employed a single zero-shot prompt for ChatGPT. Yet, it showed that ChatGPT is relatively consistent when outputting decisions to screen articles. Although we use the same datasets as [52] in this paper, we improve the methodology in several ways. When comparing its performance, traditional classifiers were trained and tested based on the F_2 metric and cross-validation did not take into account the imbalanced nature of the problem. In this paper, we train the classifier based on the more accurate metric MCC taking into account the imbalance of the data. We consider different prompt variants for ChatGPT. In fact, we show that the prompt used [52] is not the best performing prompt. We

also have a detailed discussion on the impact of the results in practice, whereas the report focuses on cost-benefit discussion of using ChatGPT instead of manually screening articles.

3. Experiment design

We followed the *Data Science* method [47], relying on a data-centric analysis method in our study. First, we construct the datasets from reliable sources. Then, we engineer different prompts to interface with ChatGPT. Finally, we conduct a comparative analysis of the performance of ChatGPT and machine learning techniques widely used in SR automation.

3.1. Data collection

We now elaborate on the details of the data collection strategy. Here, we discuss the *Data collection* phase of the study.

We use the ReLiS review software as the data source in our experiments to extract valuable data sets through careful pre-processing and filtering steps.

3.1.1. Data source

ReLiS [9] is a cloud-based tool for planning, conducting, and reporting SRs.[3] Although most SRs publish data in a replication package or appendix, replication packages contain only the final corpus of included articles. ReLiS stores the whole history of SR projects, including information about articles excluded during the screening phases, the exclusion criteria that were applied, and whether the decision was unanimous or required the resolution of a disagreement among the reviewers. In essence, ReLiS projects provide corpora labeled by highly qualified experts. Therefore, they can be considered as the ground truth to evaluate the decisions of ChatGPT about the inclusion or exclusion of articles in an SR.

We extract the datasets used in our experiments through careful pre-processing and filtering steps explained in what follows.

3.1.2. Collecting datasets

As of July 2023, ReLiS contained a total of 104 SR projects. We queried the SQL databases of ReLiS for SR projects with a screening phase and obtained 21 projects as a result. We then manually inspected these projects to select those containing real and meaningful data of proper quality. We require that the project either (i) is concluded and led to a scholarly publication or (ii) is still in progress and we can verify its quality.

We confirmed the correspondence of publications to ReLiS projects by directly contacting the authors of the publications. We also asked for their permission to use the data of their project in our work. Eventually, we identified six relevant ReLiS projects: four concluded projects with an associated publication and two ongoing projects with multiple rounds of screening and a substantial number of articles. Furthermore, we investigated the included articles in each project to ensure the correctness of the decisions. We found one project where the articles included did not match the topic of the review. After confirming with the authors, we discarded this project. Eventually, we obtained five usable data sets listed in Table 1.

Although ReLiS hosts SR projects with topics in different disciplines, our final dataset only contains SRs in software engineering.

3.1.3. Data extraction

For each project, we extract all the screening information (including all screening phases and snowballing). We corroborate that the selection criteria extracted from each ReLiS project correspond to the one used in the SRs by contacting the respective authors. For each article, we extract the following data: title, full abstract, decision whether the article was included or excluded by the reviewer(s) (this is the ground truth), decision reason (this is the selection criteria), number of ReLiS users who reviewed the article, and whether the decision was the result of a conflict that was eventually resolved among the reviewers.

We filter data records to retain only those that have all the above data. For example, we discard screened articles without an abstract recorded in ReLiS or articles that are still pending reviewer decisions. Furthermore, we exclude duplicate entries within projects.

3.1.4. Characteristics of the datasets

As shown in Table 1 and Figure 1, the datasets vary in size from 205 to 2 683 total records. The ratio of included articles ranges from 7% to 52%, with a median of 9% and an average of 18%. This

[3] https://relis.iro.umontreal.ca/

Table 1: The five datasets used in our experiments.

Project	Publication	Size	Incl.	Excl.	Conflicts	Reviewers	Project title
MPM4CPS	Barišić et al. [4]	205	107 (52.2%)	98	49 (23.9%)	2	Multi-paradigm modeling of cyber-physical systems
MobileMDE	Brunschwig et al. [13]	292	55 (18.8%)	237	154 (52.7%)	3	Modeling on mobile devices
UpdateCollabMDE	David et al. [16]	875	57 (6.5%)	818	65 (7.4%)	3	Collaborative modeling systematic update
RL4SE	*In progress*	1 089	94 (8.6%)	995	100 (9.2%)	6	Reinforcement learning for software engineering
DSMLCompo	*In progress*	2 683	150 (5.6%)	2 533	76 (2.8%)	4	Domain-specific modeling language composition
Total		**5 144**	**463**	**4 681**	**444**		

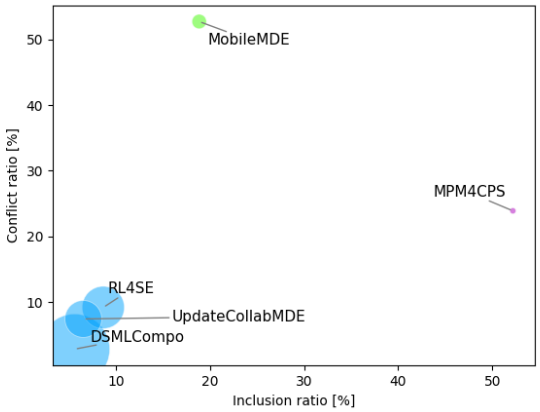


Figure 1: Inclusion ratio and conflict ratio of the dataset. Size proportional to corpus size.

is typical for SRs that are imbalanced with a predominance of excluded articles. The three largest datasets have similar inclusion ratios. Note that UpdateCollabMDE is a systematic update study, meaning that all articles are collected through forward snowballing, not using a carefully designed query like the other SRs. The inclusion ratio of the two smaller datasets varies significantly. MPM4CPS includes more than half of the articles, whereas MobileMDE includes nearly 19% of the articles, more than twice the ratio of the larger datasets. The number of articles with conflicts has a similar distribution, with an average of 19%. The datasets also differ in conflict ratio. DSMLCompo has only a few conflicts (3%). RL4SE and UpdateCollabMDE have similar conflict ratios, around 8%. MPM4CPS and MobileMDE report conflicts for a quarter and half of the articles, respectively.

3.2. Metrics

Given an article with a set of features and a ground truth decision to include or exclude the article from an SR, the problem of screening an article is to define a classifier that outputs the same decision for the article. We record a true positive (TP) when the classifier correctly includes an article, true negatives (TN) upon correct exclusion, false positives (FP) upon incorrect inclusion, and false negatives (FN) upon incorrect exclusions. These are the primary metrics to assess the performance of the classifiers. However, given that the datasets vary in size and inclusion/exclusion ratio, we rely on aggregated metrics that are derived from them and serve as a basis of comparison on a $[0, 1]$ ratio scale.

We rely on standard metrics to ensure the classifier includes articles correctly. **Precision** ($Prec$) measure if the included articles should have indeed been included. **Recall** (Rec) measures if any articles to be included have been missed. We also in-

clude metrics equivalent to precision and recall, tailored for exclusions. **Negative predictive value** (NPV) measures the ability to exclude only articles that should be excluded. **Specificity** ($Spec$) measures the ability to exclude all articles that should be excluded. They are respectively analogous to precision and recall but for negative decisions.

Work Saved over Sampling (WSS) [15] is a frequently used non-standard metric to evaluate automated screening tools, that balances between high recall and sufficient NPV. However, as Kusa et al. [32] show, normalizing WSS to a $[0, 1]$ scale results in $WSS = Spec$. Therefore, we use specificity instead of WSS.

These metrics consider inclusion and exclusion separately. For the screening task, it is important to consider both classes jointly. A successful classifier for screening should miss as few relevant articles as possible (maximize recall) and save time for the reviewers by removing as many irrelevant articles as possible (maximize specificity). Moreover, SR corpora are almost always imbalanced, favoring the exclusion class: there are more articles to exclude than to include in SRs, as evidenced by Table 1. **Balanced accuracy** ($bAcc$) captures the accuracy of inclusion and exclusion decisions. It is better suited than traditional accuracy metrics for imbalanced classes. It corresponds to the area under the receiver operating characteristic curve (AUC) when only one run is available [51]. Therefore, we rely on $bAcc$ when reporting the accuracy of a classifier. However, this metric does not take into account FN explicitly. The **Matthews correlation coefficient** (MCC) is often used as the singular metric for imbalanced data [6, 14] due to the more realistic performance estimation of binary classifiers [34]. MCC balances the ability to classify all articles as included or excluded correctly. Thus, we compare the overall performance of classifiers by their MCC. We use the normalized MCC to ensure values are in the $[0, 1]$ scale.

The following equations summarize the metrics we use in this study:

$$Prec = \frac{TP}{TP + FP} \tag{1}$$

$$Rec = \frac{TP}{TP + FN} \tag{2}$$

$$NPV = \frac{TN}{TN + FN} \tag{3}$$

$$Spec = \frac{TN}{TN + FP} \tag{4}$$

$$bAcc = \frac{Rec + Spec}{2} \tag{5}$$

$$MCC = 0.5 + \frac{TP \times TN - FP \times FN}{2 \times \sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{6}$$

3.3. Evaluated classifiers

In this study, we choose the **ChatGPT** 0613 snapshot of GPT version 3.5 Turbo. We compare ChatGPT's results with representative baseline classifiers to contextualize the results. We follow the methodology we developed in previous work [52] and choose the same four classifiers frequently used in SR screening automation (see Sec. 2.2 and [54]): Logistic regression (**LR**) [33], Complement Naive Bayes (**CNB**) [63], C-Support Vector Classification (**SVC**) [26], and Random forest (**RF**) [23]. We rely on the scikit-learn Python library [45] to implement the classifiers. We also implement a random classifier (**RAND**) that randomly assigns inclusion/exclusion decisions to articles to estimate the minimal required classification performance.

The baseline classifiers need to be trained and tuned on data sampled from the specific dataset. For training, we use distribution balanced stratified cross-validation [65] with 5 folds on each dataset, repeated 10 times. For fine-tuning, we use randomized grid search [8]. During tuning and cross-validation, we optimize the fitting process for MCC for the reasons explained in Sec. 3.2. We employ the Word2Vec algorithm to represent the features of articles (title and abstract), which captures the semantics and word relationships [42]. This way, we extract richer contextual information and enhance the representation of article features. We develop a Python program to orchestrate the experiments with all six classifiers.

4. Engineering prompts for ChatGPT

Like in any LLM, prompt formulation plays a crucial role in shaping the output of ChatGPT. Thus, we must meticulously construct the prompt and tune the hyperparameters of ChatGPT appropriately on suitable samples from the datasets.

4.1. Sampling

To ensure our experiments with the prompt are cost and time-efficient, we sample smaller batches from the overall RL4SE dataset. Each batch comprises 20–40 articles and is processed multiple

times. We develop a script to randomly select articles from the dataset with a specified ratio of inclusions and exclusions. The fixed ratio ensures statistical similarity and mitigates threats to internal validity, while random sampling improves statistical power.

4.2. Hyperparameters

Our experiments require consistent responses and are devoid of creativity in ChatGPT's output decision. Therefore, we set the `temperature` parameter to 0. We also aim to keep the response short and standardized. Thus, we opt for one-word responses from ChatGPT. We found that setting the `max_tokens` parameters to 3 yields the desired output. Other hyperparameters are left at default.

4.3. Prompt template

We set four requirements to identify the best prompt.

- The prompt should only use the information available during the planning phase of the SR to enable a rigorous integration of ChatGPT in the SR process.
- Following the recommendations in [52], the prompt should yield a high MCC score, as it considers all four *TP*, *TN*, *FP*, and *FN* quantities.
- The prompt should apply to any SR topic, ensuring a systematic and automated use of the prompt.
- The token count should be minimal, reducing costs and bandwidth.

By experimenting with various prompt alternatives using the sample (Sec. 4.1), we identified Listing 1 as the best-performing prompt template that meets all our requirements. Most variations we considered affect the *Context* and *Instructions* components.[4] The prompt template can be customized to adapt to different SR strategies.

[4]The replication package archives the less-performing prompts.

4.3.1. Context

The context describes the query's scope (line 2), i.e., conducting an SR, informing ChatGPT about the topic, and emphasizing a strong focus on it. The latter is essential when adjacent topics might be undesirable. For example, for the RL4SE review, ChatGPT should include articles focusing on reinforcement learning for software engineering, not on software engineering for reinforcement learning. Not all SR projects define their topic and scope in one succinct sentence we could use as the `TOPIC`. Hence, we assign topic descriptions to each dataset following the process:

1. Determine the scope of the SR based on elements from the published paper or protocol, especially: goal, search strings, selection criteria, and overview of expected results.
2. Formulate the scope starting from the publication title or the ReLiS project title and rephrase it into a more precise sentence based on step 1.
3. Evaluate the formulation by asking ChatGPT if it understands the scope and verify the explanation it gives.
4. Refine the formulation through iteration of step 3 until ChatGPT's explanation is satisfactory.

In our experiments, we sometimes needed to iterate over step 4 up to three times to reach a satisfactory agreement with ChatGPT's explanation. For example, the topic of the project UpdateCollabMDE is *"techniques where multiple stakeholders collaborate and manage on shared models in model-driven software engineering."*[2]

4.3.2. Examples

The example component (lines 3–5) represents few-shot prompting options. Shots are examples provided to ChatGPT to guide it to respond in a specific way. In our case, a shot consists of an article and the expected decision based on the ground truth of the dataset. We support positive shots (articles that should be included) and negative shots (articles that should be excluded).

Positive shots can be sampled from the reference set of SRs. A reference set is a collection of articles to be included, defined during the planning stage of an SR [30]. This reference set is used to calibrate the search query, exclusion criteria, and data extraction form. Unfortunately, ReLiS does not store reference sets. Hence, for our experiments, we randomly choose N^+ articles that have been included

```
1  <Prompt> ::= <Context> <Examples>? <SelectionCriteria>? <Instructions> <Task>
2  <Context> ::= 'I am screening papers for a systematic literature review. The topic of the
       systematic review is {TOPIC}. The study should focus exclusively on this topic.'
3  <Examples> ::= <ExampleHeader> <Example>+ (<ExampleHeader> <Example>+)?
4  <ExampleHeader> ::= 'I give ({N+}|{N-}) examples with {FEATURE} that should be (included|
       excluded).'
5  <Example> ::= ('Example '[1-9] ':' <Task>)+
6  <SelectionCriteria> ::= <CriteriaHeader> <Criterion>+ (<CriteriaHeader> <Criterion>+)?
7  <CriteriaHeader> ::= '(Include|Exclude) the article if (all|any) of the following ({Ni}|{Nx})
       criteria (are|is) true.'
8  <Criterion> ::= [1-9] ': {CRITERION}'
9  <Instructions> ::= 'Decide if the article should be included or excluded from the systematic
       review. I give the {FEATURE}+ of the article as input. Only answer {INCLUDE_WORD} or {
       EXCLUDE_WORD}. Be lenient. I prefer including papers by mistake rather than excluding them
        by mistake.'
10 <Task> ::= '-{FEATURE}: {INPUT}'
```

Listing 1: Prompt template definition using an EBNF notation with parameters

unanimously among the reviewers, i.e., without a conflict.

Useful negative shots are harder to obtain. Using articles that are trivial to exclude (i.e., for which the topic is clearly outside the scope of the SR) does not help span the right separating hyperplane between positive and negative objects. Useful negative shots are titles and abstracts that do not provide enough evidence that they fit the scope of the SR or raise ambiguities related to the exclusion criteria. Such articles may lead to conflicting decisions among the reviewers. Hence, for our experiments, we randomly choose N^- articles marked as conflicting and have been excluded.

4.3.3. Selection criteria

Inclusion and exclusion criteria provide a rationale for deciding whether to include an article. Therefore, the selection criteria component (lines 6–8) represents an open domain chain-of-thought [64]. Inclusion and exclusion criteria are determined during the planning stage. An article should be included if it satisfies all the inclusion criteria and excluded if it satisfies at least one exclusion criterion.

Since most inclusion criteria can be formulated as negated exclusion criteria, inclusion criteria are sometimes not explicitly defined in an SR. ReLiS, specifically, promotes using exclusion criteria. Therefore, we consider exclusion criteria only in our experiments.

There are two types of exclusion criteria in an SR: content-based and meta-information–based. Content-based exclusion criteria state the irrelevance of the article to the SR topic or the absence of specific techniques used. This information is usually found in the title and abstract. Meta-information-based exclusion criteria rely on factors such as publication date, type of article (e.g., book or thesis), or the language in which it is written. To properly capture meta-information, we use the `FEATURE` parameter (line 9). For the datasets we collected, we only consider the first type of exclusion criteria. However, some datasets have very generic exclusion criteria. For example, one of the criteria in the MobileMDE dataset is the article being *"Off topic"*. In such cases, we rephrase the exclusion criteria following the same process as for the `TOPIC` (Sec. 4.3.1).

4.3.4. Instructions

Instructions direct ChatGPT to perform the requested task (line 9). To decrease the number of *FN* at the cost of increasing *FP*, we ask ChatGPT to *be lenient*. We found that adding the sentence "I prefer including papers by mistake rather than excluding them by mistake" is also necessary to decrease *FN*. `FEATURE` parameters list the names of the meta-information to consider when screening, i.e., any of the title, abstract, keywords, venue, and authors. In our experiments, we follow the best practices of SRs [30] and screen based on title and abstract. Although with proper automation, screening based on the full text might yield more precise results, we do not consider it feasible when relying on an LLM-as-a-service, such as ChatGPT, especially with a large corpus. The `INCLUDE_WORD` and `EXCLUDE_WORD` specify the output format of ChatGPT's decision. In our experiments, we request the output words *"INCLUDE"*

```
1 I am screening papers for a systematic literature review.
2 The topic of the systematic review is reinforcement learning for software engineering.
3 The study should focus exclusively on this topic.

5 I give 2 examples with title and abstract that should be included.
6 Example 1:
7 -Title: A DQN-based agent for automatic software refactoring
8 -Abstract: Context: Nowadays, technical debt has become a very important issue in software
      project...
9 Example 2:
10 -Title: A Reinforcement Learning-Based Framework for the Generation and Evolution of Adaptation
       Rules
11 -Abstract: One of the challenges in self-adaptive systems concerns how to make adaptation...

13 Exclude the article if any of the following 2 criteria are true.
14 1: Article does not define or use a reinforcement learning method.
15 2: Software engineering is not the problem reinforcement learning is used for.

17 Decide if the article should be included or excluded from the systematic review.
18 I give the title and abstract of the article as input.
19 Only answer INCLUDE or EXCLUDE.
20 Be lenient. I prefer including papers by mistake rather than excluding them by mistake.

22 -Title: PARMOREL: a framework for customizable model repair
23 -Abstract: In model-driven software engineering, models are used in all phases of the
      development process...
```

Listing 2: Excerpt of the *PositiveX* prompt with two positive shots, applied to the RL4SE dataset

and *"EXCLUDE,"* from ChatGPT, respectively.

4.3.5. Task

Unlike the previous components that are fixed for a given SR, the task component (line 10) is specific to the articles within the SR. It states the value of each FEATURE of the article. In our case, the INPUT are the title and abstract of the screened article.

4.4. Prompt variants

We instantiate six variants[2] of the prompt template (Table 2) and compare their performance on the five corpora. Each of the *Simple*, *Positive*, and *Balanced* prompts come in an additional variant in which exclusion criteria, i.e., chain-of-thought, are explicitly listed (denoted by the suffix '*X*'). Simple prompts do not use shots. Positive prompts specify only positive shots. Balanced prompts specify the same number of positive and negative shots. To ensure conciseness, we limit the number of shots to three or two positive shots and two negative shots. Listing 2 shows the *PositiveX* prompt variant for the RL4SE dataset with [5] as the input article to screen.

The prompt template generates 4! = 24 prompt variants when we vary shots and selection criteria. We do not report prompts with only negative shots since, realistically, this information is not meaningful to the decision making (as reported later in Table 3, ChatGPT has almost perfect NPV but low precision). In terms of selection criteria, we do not report prompts with inclusion criteria as explained in Sec. 4.3.3.

5. Threats to validity

We review the main threats to the validity of the study and the way we mitigated them.

5.1. Construct validity

Choosing the measures of evaluation poses the most substantial threat to construct validity. To mitigate this threat, we followed community standards when opting for MCC as our guiding metric. The results of comparison with baselines might be artifacts of the training characteristics of classifiers we used, rather than meaningful observations about the performance of ChatGPT. To mitigate this threat, we used grid search to tune the models as recommended by community standards [47]. Each model was retrained specifically for each dataset using cross-validation. Thus, the models are not meant to be used to screen any SR and are therefore biased towards each dataset. This is to say that we can consider the baseline classifiers as "good

Table 2: Prompt variants we consider for our experiments

Prompt variant	Positive shots	Negative shots	Exclusion criteria
Simple	0	0	No
SimpleX	0	0	Yes
Positive	3	0	No
PositiveX	3	0	Yes
Balanced	2	2	No
BalancedX	2	2	Yes

enough" to establish a baseline when assessing the performance of ChatGPT.

We relied on plain text corpora that might contain editorial errors due to special characters and their encoding. For example, an en dash ("–") might be encoded as "\endash", and percentages might follow LaTeX conventions, e.g., "25\%". These errors might impact the performance of ChatGPT. We mitigated this threat by either removing problematic articles from the dataset or by applying meaningful clean-up transformations.

5.2. Internal validity.

We used manually classified SR datasets in our experiments to determine performance metrics. Due to the manual labor, these metrics are subject to threats to internal validity. To mitigate threats, we used datasets that are either associated with peer-reviewed publications, or ongoing efforts the authors of the current paper are involved in and can judge the quality of.

5.3. External validity.

We sampled only SRs that are from the software engineering domain, were conducted in the ReLiS tool, and a specific version of ChatGPT was the only LLM we evaluated. These choices pose threats to the external validity as generalizations to other domains, LLMs, and datasets require careful consideration.

6. Results

We now present the results[2] of our experiments.

6.1. RQ1. Prompting technique

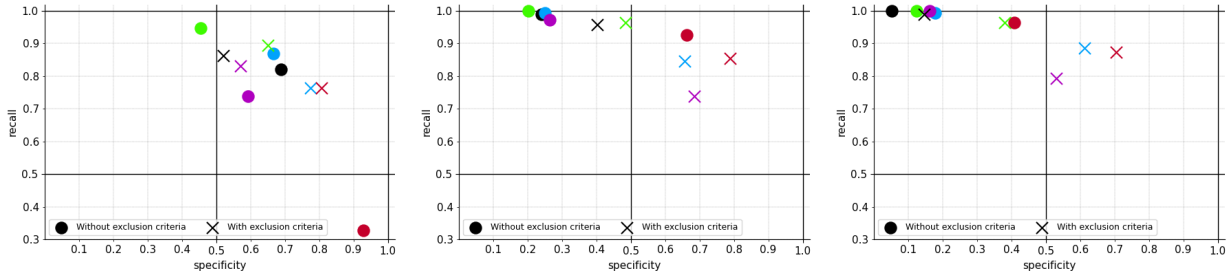
In previous work [52], we report that running prompts multiple times in similar settings to ours yields consistent decisions with ChatGPT. We confirm this observation in our current study for *Simple*, *Positive*, and *Balanced* prompt types. Hence, we run each prompt only once for each article.

Table 3 reports the performance metrics of the prompts in the different datasets. We note a generally low precision and high recall. We also note a particularly high NPV, and specificity similar to or lower than recall. These figures indicate the tendency of ChatGPT to include too many articles but rarely exclude articles incorrectly.

6.1.1. Zero-shot vs. few-shot prompts

Figure 2 highlights that **zero-shot prompts perform generally better than few-shot prompts**. Few-shot prompts (Figures 2b and 2c) have a higher recall than zero-shot prompts (Figure 2a). However, the specificity of few-shot prompts tends to deteriorate as recall increases; leaving zero-shot prompts perform better for specificity. As shown in Figure 2a, every prompt with exclusion criteria (marked with ×) is in the top-right quadrant, while other variants (marked with ●) tend to spread over more quadrants. Figure 2a also shows that adding exclusion criteria to *Simple* improves the recall for the MobileMDE dataset, and specificity for UpdateCollabMDE. The data points for *SimpleX* are closer, reflecting a more consistent recall across the five datasets. These observations are supported by quantitative evidence, in particular with MCC in Table 3. *Simple* performs best for the RL4SE dataset with *SimpleX* and *PositiveX* being the closest in performance. *SimpleX* performs best for the DSMLCompo and UpdateCollabMDE datasets, followed by *Simple* and *PositiveX*. *PositiveX* performs best for the MobileMDE and MPM4CPS datasets, followed by *SimpleX* and *Positive*. *Balanced*, *BalancedX*, and *Positive* are often the least performing variants, despite their nearly perfect recall and NPV.

Pair-wise χ^2 tests of the prompts reveal significant differences at $p = 0.000$, with a Cramer's V $\in [0.598, 0.930]$, a particularly strong effect size.[2] These figures indicate that prompt variants that perform better, do so in a statistically significant manner.



(a) Zero-shot (*Simple* and *SimpleX*)

(b) Few-shot (*Positive* and *PositiveX*)

(c) Few-shot with negative (*Balanced* and *BalancedX*)

Figure 2: Specificity vs. recall of the prompts in all datasets: RL4SE, DSMLCompo, UpdateCollabMDE, MPM4CPS, and MobileMDE

6.1.2. The effect of exclusion criteria

As shown in Figures 2b and 2c, explicitly listing exclusion criteria in few-shot prompts improves their specificity at the cost of recall. This effect is less pertinent in zero-shot prompts (Figure 2a). However, explicit exclusion criteria substantially improve the recall in the MobileMDE dataset.

In general, **adding exclusion criteria improves MCC** (Table 3). The only exception is the RL4SE dataset, where *Simple* outperforms *SimpleX* due to improved specificity. In three of the five datasets (RL4SE, MobileMDE, and MPM4CPS), *SimpleX* has a higher recall than *Simple*, while *Simple* has higher specificity than *SimpleX*.

6.1.3. The effect of negative shots

Our results indicate that **adding examples of articles that should be excluded deteriorates performance**. For example, *Balanced* and *BalancedX* miss more articles to exclude (lower specificity) than all other prompt variants. Balanced variants miss fewer articles that should be included (higher recall) in three datasets, but have a lower recall in the other two.

Answer to RQ1

Zero-shot prompts tend to perform better than few-shot prompts. Listing exclusion criteria tends to increase specificity. Using negative shots deteriorates the performance.

6.2. RQ2. Classification performance

For each dataset, we compare the highest-scoring prompt with the highest-scoring baseline classifier for MCC (see Sec. 3.3). The results are reported in Table 5 and visualized in Figure 3.

Our results indicate that both the best prompt variant and the best baseline classifier consistently outperform RAND on all metrics. The only exception is in the MPM4CPS dataset where RAND performs better than the four baseline classifiers for recall. For the remaining prompts and classifier, we notice that some perform worse than RAND for recall and specificity. **The best two or three prompt variants typically outperform the best baseline classifiers**. The only exception is RL4SE, where all baseline classifiers outperform all ChatGPT prompts for MCC. In this case, the prompt variants have the highest recall but the lowest specificity.

The *bAcc* measures[2] indicate that ChatGPT prompts tend to classify articles with higher accuracy in all datasets. However, only *Simple* has better accuracy than LR, in the RL4SE dataset. Table 4 shows that *SimpleX*, *PositiveX*, and *Simple* have the most consistent accuracy across all datasets, centered around 75%, 73%, and 70%.

Answer to RQ2

ChatGPT classifies articles more accurately than baseline classifiers. Its prompts reach higher recall but lower specificity.

6.3. RQ3. Impact of corpus characteristics

From Table 1, we group datasets into three categories. RL4SE, DSMLCompo, and UpdateCollabMDE are *typical* SR corpora with low inclusion and conflict rates (both under 10%). MobileMDE has moderate inclusion (around 19%) and high conflict rate (over 50%), where the reviewers had a significant amount of disagreement, revealing perhaps ambiguous selection criteria. MPM4CPS has a high inclusion rate (over 50%) and a moderate conflict

Table 3: Performance of prompt variants. Bold is best.

		Simple	SimpleX	Positive	PositiveX	Balanced	BalancedX
RL4SE	Rec	0.821	0.864	0.989	0.957	**1.000**	0.989
	Prec	**0.199**	0.146	0.110	0.131	0.091	0.099
	Spec	**0.688**	0.521	0.241	0.402	0.052	0.147
	NPV	0.976	0.976	0.996	0.990	**1.000**	0.993
	bAcc	**0.755**	0.692	0.615	0.680	0.526	0.568
	MCC	**0.649**	0.608	0.578	0.604	0.534	0.556
DSMLCompo	Rec	0.869	0.763	**0.993**	0.847	**0.993**	0.887
	Prec	0.133	**0.167**	0.073	0.127	0.067	0.119
	Spec	0.666	**0.774**	0.250	0.656	0.179	0.612
	NPV	0.988	0.982	**0.998**	0.986	**0.998**	0.989
	bAcc	0.767	**0.769**	0.622	0.752	0.586	0.749
	MCC	0.628	**0.642**	0.566	0.620	0.553	0.616
UpdateCollab.	Rec	0.947	0.895	**1.000**	0.965	**1.000**	0.965
	Prec	0.108	**0.151**	0.080	0.116	0.074	0.098
	Spec	0.455	**0.650**	0.203	0.485	0.125	0.380
	NPV	0.992	0.989	**1.000**	0.995	**1.000**	0.994
	bAcc	0.701	**0.773**	0.601	0.725	0.562	0.673
	MCC	0.600	**0.638**	0.564	0.612	0.548	0.589
MobileMDE	Rec	0.327	0.764	0.927	0.855	**0.964**	0.873
	Prec	**0.514**	0.477	0.389	0.485	0.275	0.407
	Spec	**0.928**	0.806	0.662	0.789	0.409	0.705
	NPV	0.856	0.936	0.975	0.959	**0.980**	0.960
	bAcc	0.628	0.785	0.795	**0.822**	0.686	0.789
	MCC	0.654	0.743	0.732	**0.767**	0.654	0.730
MPM4CPS	Rec	0.738	0.832	0.972	0.738	**1.000**	0.794
	Prec	0.664	0.679	0.591	**0.718**	0.566	0.649
	Spec	0.592	0.571	0.265	**0.684**	0.163	0.531
	NPV	0.674	0.757	0.897	0.705	**1.000**	0.703
	bAcc	0.665	0.702	0.619	**0.711**	0.582	0.663
	MCC	0.667	0.710	0.670	**0.711**	0.652	0.669

ratio (around 24%), with a similar amount of articles included and excluded, indicating that the reviewers may have prefiltered the article's search significantly.

Our results show that the characteristics of datasets influence the performance profiles of ChatGPT prompts. As shown in Figure 3, the best prompt exhibits a similar performance profile in typical SR corpora, with particularly low precision. Increasing the conflict ratio significantly mostly impacts precision negatively. Increasing the inclusion rate at par with the exclusion rate balances the classification performance with a similar score on all metrics.

Spearman's rank correlation tests of the metrics in Table 5 with respect to the inclusion ratio of datasets reveal significant positive associations ($p \leq 0.050, \rho = 0.975$) for precision of all prompt variants except *SimpleX*. They also reveal significant negative associations ($p = 0.037, \rho = -0.900$) for NPV of *Simple*, *SimpleX*, and *Positive*. These figures indicate ChatGPT's tendency to include articles correctly better, with a higher number of articles to include. Conversely, it excludes articles better, with a higher number of articles to exclude.

Spearman tests for conflicts reveal negative associations ($p = 0.037, \rho = -0.900$) for recall of *Simple* and *Positive*. Nonetheless, in Table 5 we observe substantial increase in precision as conflict ratio increases to moderate levels—around 6× more for all prompt variants. However, as the conflict ratio further increases (e.g., in MobileMDE), precision increases less prominently, around 4× more. NPV is negatively affected by a moderate conflict ratio.

In typical datasets, precision is lower. Table 5 shows precision ranges between 15–20% in typical datasets and between 49–72% in atypical ones. There is a difference in the best-performing prompt strategy as well. Zero-shot prompts perform better in typical datasets—*Simple* in the RL4SE dataset, and *SimpleX* in DSMLCompo and UpdateCollabMDE. However, few-shot prompts, specifically *Pos-*

Table 4: Moment statistics of *bAcc* for the best prompt variants and baseline classifiers across all datasets. Bold is best.

Classifier	Min	Max	Mean	Median	Std. dev.	IQR	Kurtosis
SimpleX	**0.692**	0.785	**0.744**	**0.769**	**0.044**	0.071	-3.022
PositiveX	0.680	**0.822**	0.738	0.725	0.054	**0.041**	1.307
Simple	0.628	0.767	0.703	0.701	0.059	0.089	-1.985
CNB	0.550	0.721	0.663	0.718	0.080	0.111	-1.709
LR	0.566	0.739	0.645	0.619	0.086	0.168	**-3.092**
SVC	0.525	0.767	0.637	0.666	0.105	0.159	-2.149

itiveX, perform best in atypical datasets. *PositiveX* also outperforms the best classifier in atypical datasets in all metrics but specificity.

Answer to RQ3

The characteristics of the corpus affect the performance profile of ChatGPT. Different corpus characteristics might favor different prompt strategies.

7. Discussion

We discuss the results, elaborate on the impact of LLMs on SRs, and provide recommendations to integrate ChatGPT into SR tools. We highlight the key takeaways in bold.

7.1. Effectiveness of ChatGPT in screening

In this work, we were interested in the potential of ChatGPT in screening automation. We evaluated the pre-trained ChatGPT for different SR topics in software engineering.

As the key takeaway, we found that **ChatGPT outperforms traditional classifiers** used in SR automation, and that, without training. Training baseline classifiers improves their specificity but they still miss more articles to include than ChatGPT. Furthermore, traditional classifiers, like the ones used in Abstrackr (c.f. Sec. 2.2), need to be retrained for each SR, which limits their applicability. In contrast, ChatGPT can be optimized with optional hyperparameters and few-shot prompting.

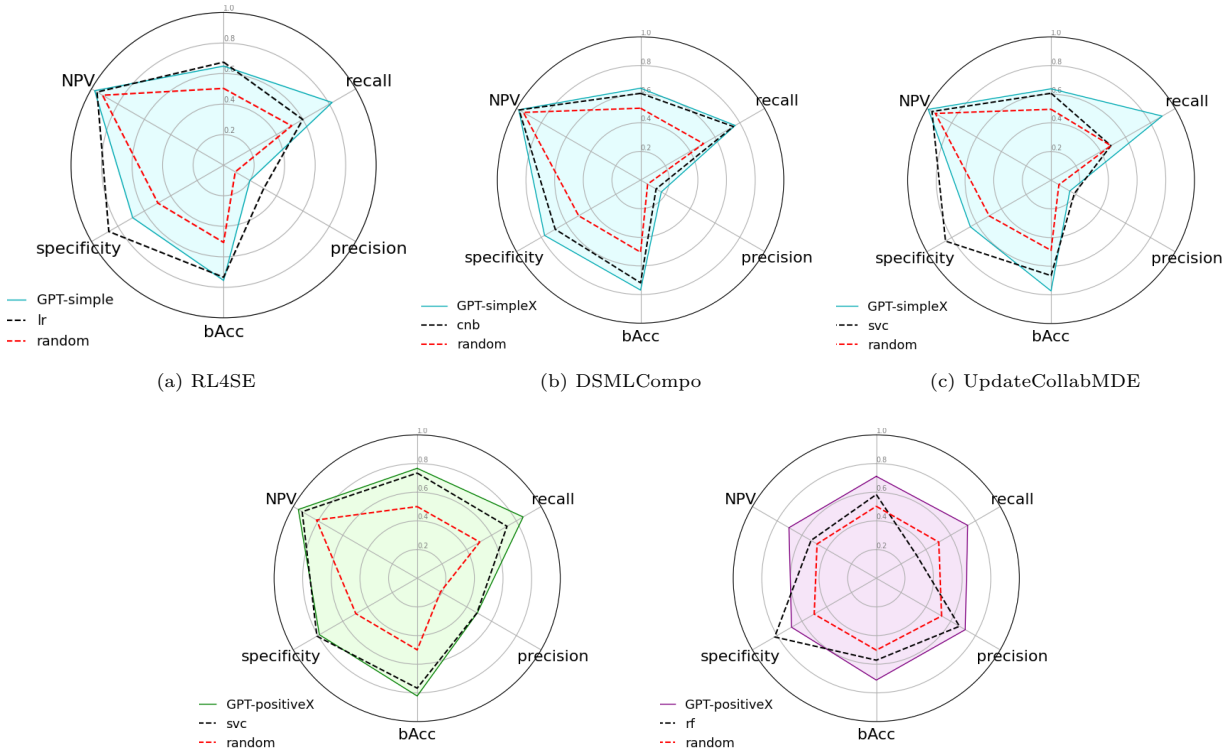
Although the results are appealing, more evidence and assessment is needed before ChatGPT can become part of SR processes. Given the limited number of datasets we experimented with, we cannot generalize the results of the prompts to any corpus of an SR project. Nevertheless, the datasets we collected are heterogeneous in terms of size, inclusion ratio, and conflict ratio, increasing the representativeness of our results.

Among the prompt variants we studied, *SimpleX*, i.e., **a prompt without shots but with explicitly listed exclusion criteria has shown the most consistent accuracy** independent of the characteristics of the corpus, while also keeping the length of the prompt minimal. It provides the best balance between not missing articles to include and to exclude: with a high recall (75–90%) and moderate specificity (50–80%). Fine-tuning GPT for the task can improve performance, however, this requires a larger training dataset and more computational resources than SR tools typically rely on. Overall, we recommend specifying exclusion criteria and avoiding negative shots. However, to confirm generalizability, experiments with further prompt variants and more datasets are needed.

7.2. Engineering the right prompts

Through our experiments, we derived a prompt template for SRs following the process described in Sec. 4. It satisfies the requirements of using information from the planning phase, maximizing MCC, being applicable to different SR topics, and minimizing cost. However, it might not be universally optimal for SRs in domains much different from software engineering. Finding the right balance between recall and specificity is paramount, as screening articles is an imbalanced dichotomous classification problem. Thus, **allowing researchers to experiment to find the best prompt for a given SR** might be an important feature in LLM-driven SR tools.

There is an emerging need for systematic prompt engineering methods [61]. Tools like PromptMaker [25] and ChainForge [3] may inspire SR tool builders to assist researchers in finding right prompts for their SRs. Allowing the user to specify the articles to use for their experiments for prompt engineering and supporting the generation of small, randomized samples will likely improve the adoption of LLMs into SR processes. Generating prompts by ChatGPT from labeled inputs (e.g.,



(a) RL4SE (b) DSMLCompo (c) UpdateCollabMDE

(d) MobileMDE (e) MPM4CPS

Figure 3: Radar plots showing the performance profiles of ChatGPT on different data characteristics. Color coding corresponds to Figure 1: typical, high conflict, and high inclusion.

the gold set or a pilot study) might be another useful feature.

7.3. Integration in SR tools

We derive recommendations for SR tool builders to integrate ChatGPT into their tools.

7.3.1. The choice of the prompt

The best prompt should miss as few articles to include as possible (maximize *Rec*) and reduce the reviewers' effort by reducing the number of articles to exclude (maximize *Spec*). Thus, **the best prompt should maximize *bAcc***. Inspecting all the 5 144 decisions from all five datasets combined, *SimpleX* has the highest balanced accuracy at 76% overall. It provides a trade-off between missing 19% of the articles that should be included and correctly excluding 70%.

What does this mean in practice? Let us assume an SR tool, like ReliS, relies on this prompt variant to screen articles of an SR on any of the five topics we considered in this study automatically. On the one hand, the review may miss around 10–24% of relevant literature according to our results. This is a significant threat to the internal validity of the SR because of selection bias. On the other hand, it will have correctly discarded 52–80%, which significantly reduces the work effort of the reviewers.

Nonetheless, if an SR tool relies on this prompt variant, **it threatens the internal validity** of the SR because of selection bias. However, **it substantially reduces the work effort of the reviewers**. Although exact recall and specificity scores cannot be known before screening is complete, the SR tool could provide an estimate. Users of the SR tool should be aware of this trade-off, so they can devise a protocol for their SR accordingly.

Current SR tools that automatically screen articles rely on certainty-based active learning and uncertainty sampling (see Sec. 2.2). However, this technique requires a human reviewer to screen articles to help the SR tool rank articles by likelihood of relevance. In practice, the reviewer may need to manually include half of the articles to include before the system achieves an acceptable classification performance [38]: the optimal stopping point

Table 5: Performance comparison of the best prompt, the best baseline, and random based on MCC. Bold is best.

RL4SE		Simple	LR	RAND
	Rec	**0.821**	0.599	0.515
	Prec	0.199	**0.304**	0.088
	Spec	0.688	**0.869**	0.497
	NPV	**0.976**	0.958	0.916
	bAcc	**0.755**	0.734	0.506
	MCC	0.649	**0.675**	0.503
DSMLCompo		SimpleX	CNB	RAND
	Rec	**0.763**	0.748	0.508
	Prec	**0.167**	0.125	0.057
	Spec	**0.774**	0.688	0.499
	NPV	**0.982**	0.979	0.945
	bAcc	**0.769**	0.718	0.504
	MCC	**0.642**	0.606	0.502
UpdateCollab.		SimpleX	SVC	RAND
	Rec	**0.895**	0.482	0.482
	Prec	0.151	**0.187**	0.063
	Spec	0.650	**0.850**	0.498
	NPV	**0.989**	0.959	0.932
	bAcc	**0.773**	0.666	0.490
	MCC	**0.638**	0.607	0.495
MobileMDE		PositiveX	SVC	RAND
	Rec	**0.855**	0.725	0.505
	Prec	**0.485**	0.483	0.189
	Spec	0.789	**0.809**	0.495
	NPV	**0.959**	0.928	0.812
	bAcc	**0.822**	0.767	0.500
	MCC	**0.767**	0.734	0.500
MPM4CPS		PositiveX	RF	RAND
	Rec	**0.738**	0.324	0.504
	Prec	**0.718**	0.671	0.527
	Spec	0.684	**0.820**	0.501
	NPV	**0.705**	0.527	0.478
	bAcc	**0.711**	0.572	0.502
	MCC	**0.711**	0.584	0.502

can only be determined in retrospect. In contrast, with the prompt template we propose, the SR tool could achieve acceptable performance without forcing the reviewers to screen any article (*SimpleX*) or requiring them to only find a few articles relevant to the SR topic (*PositiveX*).

7.3.2. Level of automation

Kosar et al. [31] have demonstrated on three corpora that missing 25–45% of relevant literature would not compromise the results of the SR within a 5% margin of error. Although their findings need to be verified on more SR projects, all prompts we evaluated achieve a recall above this threshold. In particular, *Positive* and *Balanced* consistently achieve a recall above 95%. Thus, these two variants could be used to fully automate the screening process, although investigation on more datasets would need to confirm this. However, their low precision would still require excluding a substantial amount of articles manually; therefore, they are not useful in practice. The high NPV achieved by all prompt variants indicates that ChatGPT excludes articles correctly.

Our results show that screening articles cannot be fully automated with ChatGPT yet: it is not ready to replace article screening by humans. However, it is a promising solution to assist reviewers in the screening process, e.g., to discard all the articles ChatGPT has excluded. For example, ChatGPT can perform an initial screening pass to classify the articles. Then, the human reviewer can only validate a sample of the excluded articles.

7.3.3. Leveraging the conversational features of ChatGPT

Our experiments simulate situations where ChatGPT is integrated into an SR tool through its API. We spawn new sessions for each query to mitigate potential bias stemming from conversation history. This simulates the use case where, given a corpus with additional information on the specific SR (topic, exclusion criteria, positive shots), the SR tool would then record a boolean decision for each article of the corpus without human intervention. However, some activities in the SR process might require true conversational features, such as conflict resolution between the human and the machine. Conflicts are resolved by discussing the reasons that led to differing decisions about a particular article [30]. In the case of human-machine collaboration, the human might want to query the machine about the reasons behind its decision and resolve the conflict accordingly. ChatGPT could then calibrate its decisions for subsequent articles of the corpus, similarly to active learning.

We recommend tool builders to prepare for a wide range of interaction patterns between humans and LLMs. In particular, **we suggest developing conversational features**. However, dangers of biasing and over-fitting must be mitigated carefully.

7.3.4. Support for SR strategies and SR process generation

Our investigation shows that there is no universal optimization strategy to tune prompts for LLMs. While reasonable default SR processes are important to allow inexperienced researchers to get on board with empirical methods, it is also important

to empower researchers to make their own decisions about how the SR process should be optimized. Ideally, researchers should be able to compare the results of different prompting techniques on a small subset of the corpus to better understand which strategy works for them best. For example, when conducting rapid reviews, one might want to optimize for including articles that should indeed be included (i.e., minimize FP). However, one might want to optimize for not losing articles on account of incorrect exclusions (i.e., minimize FN).

We recommend tool builders to **develop functionality that allows researchers expressing their desired SR strategy** and, based on these high-level descriptions, generate the SR process, supported by appropriate GUIs (e.g., web forms [9]). Some of the important strategy variability points to anticipate are the following: being lenient or rigorous in screening (requires different validation strategies), number of shots used, and automatically calculating necessary validation ratio based on accuracy metrics.

7.3.5. Technical limitations and pitfalls

Although integrating ChatGPT in an SR tool reduces the reviewer's effort, it comes with drawbacks. **Network brittleness might become a limiting factor in working with LLM-as-a-service solutions** like the ones OpenAI offers. During our experiments with ChatGPT, we encountered occasional errors due to timeout or rate limits. We handled these situations by mechanisms that allowed us to re-run queries on previously erroneous responses. We noticed slower processing time partly due to network delays, but mainly due to the limited number of requests OpenAI allows per minute. OpenAI recently released a new Batch API[5] to submit multiple requests simultaneously for a lower price. However, the longer waiting time to receive the answers may not be suitable for integrating it in SR tools. Therefore, it is crucial for tool builders to prioritize the development of features that can effectively handle these reliability challenges. **On-premises LLMs may solve these issues**; however, they require a robust infrastructure and specialized expertise.

Monetary cost may also be an issue to offer ChatGPT services in an SR tool. As an example, this study used over 75 million tokens for the cost of 153.79 USD. We recommend tool builders to **develop informative and transparent cost reporting** that researchers can trust in estimating and conducting their studies.

Another limitation is that **article features must be curated before being sent in the prompt**. Getting abstracts from search engines is not always available (e.g., DBLP), not properly formatted (e.g., special characters), or incomplete (e.g., Google Scholar). For this study, we manually curated the abstracts to ensure ChatGPT has the complete article features to output its decision. SR tools should provide support to ensure the quality of the features they record. Nevertheless, if the requested feature requires access to the full text of the articles, copyright restrictions should be properly handled.

7.4. Impact on SR processes

Employing ChatGPT to automate screening has an impact on other phases of the SR process as well, such as validation and piloting. The formal guidelines traditionally used for SR should be adapted when humans are not the sole reviewers and LLMs are used. We outline how current SR methodologies will be impacted.

7.4.1. Screening

Most SR guidelines, such as the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) [44], recommend that at least two reviewers screen each article to mitigate selection bias. The average number of authors of software engineering papers is about 2.67 [19], indicating potential difficulties in building large teams for an SR. **Considering ChatGPT as an automated reviewer could allow for pairing it with a solo human reviewer**, paving the way to small-team and solo SRs. In case of conflicts, the human reviewer could consult ChatGPT using its chat facilities to investigate the reason behind the different decisions. Seniority of the reviewer might be a factor to be considered. While more experienced researchers might be able to treat the decisions of ChatGPT with relative confidence, less experienced researchers new to the topic of the SR might get misled by the credible-sounding but sometimes unsound arguments of ChatGPT [7].

Rapid reviews that trade completeness for substantially reduced completion time [46] can benefit

[5] https://platform.openai.com/docs/guides/batch/

from ChatGPT-based automation as well. Typical shortcuts in rapid reviews are related to the size of the corpus, such as restricting literature search, omitting snowballing, and streamlining screening [21]. With the support of LLMs, **rapid reviews can be conducted without quality compromise**. Screening can be fully automated and the effort that would have been spent on screening can be used for validation, quality assessment, and fighting publication bias by snowballing.

All factors considered, we expect ChatGPT to improve the speed and quality of screening, and allow for novel personnel configurations in this crucial phase. We should **revise the formal guidelines traditionally used for SR** [30, 44] to cope with the use of LLMs.

7.4.2. Piloting

Performance profiles of ChatGPT can be obtained in the pilot phase of the SR by screening a small sample of articles and evaluating the performance of ChatGPT. Piloting allows human researchers to experiment with different prompt variants and observe which one works best for the corpus at hand. We have shown that the characteristics of the corpus influence the performance of the LLM. Therefore, **it is crucial to pilot the SR to tune the prompt accordingly.** Since we have shown that no single prompt systematically outperforms another, we need to choose the suitable prompt variant for a given corpus.

Once the prompt has been chosen, screening on the remainder of the corpus can commence. Just like with human reviewers, we also need to appropriately sample the corpus to calibrate the inclusion-exclusion decisions. Sampling includes determining the right amount of articles, selection criteria, the topic of the SR, etc. Pilot samples must remain much smaller than the corpus to avoid defeating the purpose of automation. Nonetheless, **determining the pilot sample size that yields a faithful estimate** of the performance profile requires further investigation.

7.4.3. Calibration thresholds

It is important to determine what is the threshold to consider the LLM decisions reliable. Cohen's κ can be used to measure inter-rater agreement between ChatGPT and the reviewer. For example, if $\kappa < 0.8$, one should not rely on the configuration and test another prompt variant.

There are various metrics to calibrate the prompt's performance. Although **we recommend balanced accuracy**, some SRs may want to focus more on exclusions to use the LLM as a filter of obvious articles to exclude, for example. In this case, specificity may be a more suitable metric for calibration.

Another factor to integrate an LLM in the article selection process is deciding whether to treat the LLM as a full-fledged reviewer in the team of human reviewers. **Proper guidelines should be elaborated to allow challenging its decisions** based on established quality metrics. Conflicting decisions between humans and LLM should be properly managed by identifying suitable resolution strategies, such as majority vote, priority by seniority, and introducing another reviewer to validate and break ties.

7.4.4. Validation of screening

To limit selection bias, e.g., in SRs where each article is screened by one reviewer, another reviewer typically validates an appropriate sample of the excluded papers (typically 20%) [30]. We recommend rethinking validation mechanisms with an LLM-in-the-loop. A quantified performance profile of ChatGPT might allow for gauging the required samples to validate more precisely. These profiles can be calculated from the number of conflicts or by comparing the performance of ChatGPT to the human's performance when screening. Our experiments show a nearly perfect NPV, i.e., ChatGPT only excludes articles that should indeed be excluded. That is, **validation of excluded papers might require smaller samples**. Thus, we recommend developing quantified methods to determine the portion of the corpus to be validated.

There are also opportunities in automating validation, e.g., by additional runs of ChatGPT, using dedicated prompts, although it is substantially (but not perfectly) consistent with its decisions [52].

8. Conclusion

This work provides insights into the opportunities in using ChatGPT to screen articles in SRs. Our experiments confirm the ability of ChatGPT to screen articles with accuracy that outperforms classifiers currently used for SR automation, and that, without prior training. However, our results

also show that screening articles cannot be fully automated with ChatGPT yet, and the human in the loop is still required in SRs.

In future work, we plan to evaluate different LLMs on a broader set of corpora and integrate them within an SR tool, like ReLiS, to gain further insights into the tooling aspect of SR automation. We will consider various aspects of prompts, including context length, hallucination, and role-playing. We plan to test newer GPT models like GPT-4 and GPT-4o and expect that they will achieve better screening performance. We will also consider different LLM families like Claude 3, Google Gemini, and Llama. This comprehensive approach will also ensure the testing of model-specific biases in the future.

References

[1] A. Al-Zubidy, J. C. Carver, D. P. Hale, and E. E. Hassler. Vision for SLR tooling infrastructure: Prioritizing value-added requirements. *Information and Software Technology*, 91:72–81, 2017. doi: 10.1016/j.infsof.2017.06.007.

[2] A. Alharbi, W. Briggs, and M. Stevenson. Retrieving and ranking studies for systematic reviews: University of sheffield's approach to CLEF ehealth 2018 task 2. In L. Cappellato, N. Ferro, J. Nie, and L. Soulier, editors, *Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum, Avignon, France, September 10-14, 2018*, volume 2125 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2018.

[3] I. Arawjo, C. Swoopes, P. Vaithilingam, M. Wattenberg, and E. Glassman. Chainforge: A visual toolkit for prompt engineering and llm hypothesis testing. Report 2309.09128, arXiv, 2023.

[4] A. Barišić, I. Ruchkin, D. Savić, M. A. Mohamed, R. Al-Ali, L. W. Li, H. Mkaouar, R. Eslampanah, M. Challenger, D. Blouin, O. Nikiforova, and A. Cicchetti. Multi-paradigm modeling for cyber-physical systems: A systematic mapping review. *Journal of Systems & Software*, 183:111081, 2022. doi: https://doi.org/10.1016/j.jss.2021.111081.

[5] A. Barriga, R. Heldal, A. Rutle, and L. Iovino. Parmorel: a framework for customizable model repair. *Software & Systems Modeling*, 21(5):1739–1762, Oct 2022. doi: 10.1007/s10270-022-01005-0.

[6] M. Bekkar, H. K. Djemaa, and T. A. Alitouche. Evaluation measures for models assessment over imbalanced data sets. *Journal of Information Engineering and Applications*, 13(10):27–38, 2013.

[7] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Fairness, Accountability, and Transparency*, pages 610–623. ACM, 2021. doi: 10.1145/3442188.3445922.

[8] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012.

[9] B. Bigendako and E. Syriani. Modeling a tool for conducting systematic reviews iteratively. In *Model-Driven Engineering and Software Development*, pages 552–559. SciTePress, 2018. doi: 10.5220/0006664405520559.

[10] S. S. Biswas. Potential use of chat gpt in global warming. *Annals of Biomedical Engineering*, Mar 2023. doi: 10.1007/s10439-023-03171-8.

[11] S. S. Biswas. Role of chat gpt in public health. *Annals of Biomedical Engineering*, 51(5):868–869, May 2023. doi: 10.1007/s10439-023-03172-7.

[12] R. Borah, A. W. Brown, P. L. Capers, and K. A. Kaiser. Analysis of the time and workers needed to conduct systematic reviews of medical interventions using data from the prospero registry. *BMJ Open*, 7(2), 2017. doi: 10.1136/bmjopen-2016-012545.

[13] L. Brunschwig, E. Guerra, and J. de Lara. Modelling on mobile devices. *Software & Systems Modeling*, 21(1):179–205, Feb 2022. doi: 10.1007/s10270-021-00897-8.

[14] D. Chicco and G. Jurman. The matthews correlation coefficient (mcc) should replace the roc auc as the standard metric for assessing binary classification. *BioData Mining*, 16(4), 2023. doi: 10.1186/s13040-023-00322-4.

[15] A. M. Cohen, W. R. Hersh, K. Peterson, and P.-Y. Yen. Reducing workload in systematic review preparation using automated citation classification. *Journal of the American Medical Informatics Association*, 13(2):206–219, 2006. doi: doi.org/10.1197/jamia.M1929.

[16] I. David, K. Aslam, S. Faridmoayer, I. Malavolta, E. Syriani, and P. Lago. Collaborative model-driven software engineering: A systematic update. In *Model Driven Engineering Languages and Systems*, pages 273–284. IEEE, 2021. doi: 10.1109/MODELS50736.2021.00035.

[17] K. R. Felizardo and J. C. Carver. *Automating Systematic Literature Review*, pages 327–355. Springer, Cham, 2020. doi: 10.1007/978-3-030-32489-6_12.

[18] G. Ferdinands, R. Schram, J. de Bruin, A. Bagheri, D. L. Oberski, L. Tummers, and R. van de Schoot. Active learning for screening prioritization in systematic reviews - a simulation study. OSF Preprints, Sep 2020.

[19] D. Fiala and G. Tutoky. Computer science papers in web of science: A bibliometric analysis. *Publications*, 5 (4), 2017. doi: 10.3390/publications5040023.

[20] L. Floridi and M. Chiriatti. GPT-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30(4):681–694, 2020.

[21] R. Ganann, D. Ciliska, and H. Thomas. Expediting systematic reviews: methods and implications of rapid reviews. *Implementation Science*, 5(1):56, Jul 2010. doi: 10.1186/1748-5908-5-56.

[22] B. Hasan, S. Saadi, N. S. Rajjoub, M. Hegazi, M. Al-Kordi, F. Fleti, M. Farah, I. B. Riaz, I. Banerjee, Z. Wang, and M. H. Murad. Integrating large language models in systematic reviews: a framework and case study using robins-i for risk of bias assessment. *BMJ Evidence-Based Medicine*, 2024. doi: 10.1136/bmjebm-2023-112597.

[23] M. Z. Islam, J. Liu, J. Li, L. Liu, and W. Kang. A semantics aware random forest for text classification. In *Information and Knowledge Management*, pages 1061–1070. ACM, 2019. doi: 10.1145/3357384.3357891.

[24] X. Ji, A. Ritter, and P.-Y. Yen. Using ontology-based semantic similarity to facilitate the article screening process for systematic reviews. *Journal of Biomedical Informatics*, 69:33–42, 2017. doi: 10.1016/j.jbi.2017.03.

007.

[25] E. Jiang, K. Olson, E. Toh, A. Molina, A. Donsbach, M. Terry, and C. J. Cai. Promptmaker: Prompt-based prototyping with large language models. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–8. ACM, 2022. doi: 10.1145/3491101.3503564.

[26] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Machine Learning: ECML-98*, volume 1398 of *LNAI*, pages 137–142. Springer, 1998. doi: 10.1007/BFb0026683.

[27] S. R. Jonnalagadda, P. Goyal, and M. D. Huffman. Automating data extraction in systematic reviews: a systematic review. *Systematic Reviews*, 4(1):78, 2015.

[28] M. Khabsa, A. Elmagarmid, I. Ilyas, H. Hammady, and M. Ouzzani. Learning to identify relevant studies for systematic reviews using random forest and external information. *Machine Learning*, 102:465–482, 2016. doi: 10.1007/s10994-015-5535-7.

[29] Q. Khraisha, S. Put, J. Kappenberg, A. Warraitch, and K. Hadfield. Can large language models replace humans in the systematic review process? evaluating gpt-4's efficacy in screening and extracting data from peer-reviewed and grey literature in multiple languages. Preprint 2310.17526, arXiv, oct 2023.

[30] B. A. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. Report EBSE 2007-001, Keele University and Durham University, jul 2007.

[31] T. Kosar, S. Bohra, and M. Mernik. A systematic mapping study driven by the margin of error. *Journal of Systems & Software*, 144:439–449, 2018. doi: https://doi.org/10.1016/j.jss.2018.06.078.

[32] W. Kusa, A. Lipani, P. Knoth, and A. Hanbury. An analysis of work saved over sampling in the evaluation of automated citation screening in systematic literature reviews. *Intelligent Systems with Applications*, 18 (200193), 2023. doi: 10.1016/j.iswa.2023.200193.

[33] M. P. LaValley. Logistic regression. *Circulation*, 117(18):2395–2399, 2008. doi: 10.1161/CIRCULATIONAHA.106.682658.

[34] L. Lavazza, S. Morasca, and G. Rotoloni. On the reliability of the area under the roc curve in empirical software engineering. In *Evaluation and Assessment in Software Engineering*, pages 93–100. ACM, 2023. doi: 10.1145/3593434.3593456.

[35] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, volume 32, pages 1188–1196. PMLR, 2014.

[36] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023. doi: 10.1145/3560815.

[37] R. Mallett, J. Hagen-Zanker, R. Slater, and M. Duvendack. The benefits and challenges of using systematic reviews in international development research. *Journal of Development Effectiveness*, 4(3):445–455, 2012. doi: 10.1080/19439342.2012.711342.

[38] I. Marshall and B. Wallace. Toward systematic review automation: a practical guide to using machine learning tools in research synthesis. *Systematic reviews*, 8(163): 1–10, 2019. doi: 10.1186/s13643-019-1074-9.

[39] D. Martinez, S. Karimi, L. Cavedon, and T. Baldwin. Facilitating biomedical systematic reviews using ranked text retrieval and classification. In *Australasian Document Computing Symposium*, pages 53–60. ADCS, 2008. doi: http://adcs-conference.org/2008/proceedings/p09-martinez.pdf.

[40] S. Matwin, A. Kouznetsov, D. Inkpen, O. Frunza, and P. O'Blenis. A new algorithm for reducing the workload of experts in performing systematic reviews. *Journal of the American Medical Informatics Association*, 17(4): 446–453, 2010. doi: 10.1136/jamia.2010.004325.

[41] R. W. McGee. What will the united states look like in 2050? a chatgpt short story. *SSRN Electronic Journal*, 2023. doi: 10.2139/ssrn.4413442.

[42] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[43] M. Miwa, J. Thomas, A. O'Mara-Eves, and S. Ananiadou. Reducing systematic review workload through certainty-based screening. *Journal of Biomedical Informatics*, 51:242–253, 2014. doi: https://doi.org/10.1016/j.jbi.2014.06.005.

[44] M. J. Page, D. Moher, P. M. Bossuyt, I. Boutron, T. C. Hoffmann, C. D. Mulrow, L. Shamseer, J. M. Tetzlaff, E. A. Akl, S. E. Brennan, R. Chou, J. Glanville, J. M. Grimshaw, A. Hróbjartsson, M. M. Lalu, T. Li, E. W. Loder, E. Mayo-Wilson, S. McDonald, L. A. McGuinness, L. A. Stewart, J. Thomas, A. C. Tricco, V. A. Welch, P. Whiting, and J. E. McKenzie. PRISMA 2020 explanation and elaboration: updated guidance and exemplars for reporting systematic reviews. *BMJ*, (160), 2021. doi: 10.1136/bmj.n160.

[45] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, and V. e. a. Dubourg. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.

[46] P. Ralph, N. bin Ali, S. Baltes, D. Bianculli, J. Diaz, Y. Dittrich, N. Ernst, M. Felderer, R. Feldt, A. Filieri, B. B. N. de França, C. A. Furia, G. Gay, N. Gold, D. Graziotin, P. He, R. Hoda, N. Juristo, B. Kitchenham, V. Lenarduzzi, J. Martínez, J. Melegati, D. Mendez, T. Menzies, J. Molleri, D. Pfahl, R. Robbes, D. Russo, N. Saarimäki, F. Sarro, D. Taibi, J. Siegmund, D. Spinellis, M. Staron, K. Stol, M.-A. Storey, D. Taibi, D. Tamburri, M. Torchiano, C. Treude, B. Turhan, X. Wang, and S. Vegas. Paving the way for mature secondary research: The seven types of literature review. ESEC/FSE 2022, pages 1632–1636, New York, NY, USA, 2022. ACM. doi: 10.1145/3540250.3560877.

[47] P. e. a. Ralph. Empirical standards for software engineering research. Report 2010.03525, arXiv, 2021.

[48] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084, 2019.

[49] I. Rozanc and M. Mernik. Chapter three - the screening phase in systematic reviews: Can we speed up the process? In *Advances in Computers*, volume 123, pages 115–191. Elsevier, 2021. doi: https://doi.org/10.1016/bs.adcom.2021.01.006.

[50] B. Settles. Active learning literature survey. Report TR-1648, University of Wisconsin-Madison, jan 2009.

[51] M. Sokolova, N. Japkowicz, and S. Szpakowicz. Beyond accuracy, f-score and roc: A family of discriminant measures for performance evaluation. In *AI 2006: Advances in Artificial Intelligence*, volume 4304 of *LNAI*, pages 1015–1021. Springer, 2006.

[52] E. Syriani, I. David, and G. Kumar. Assessing the ability of chatgpt to screen articles for systematic reviews. Report 2307.06464, arXiv, jul 2023.

[53] G. Tsafnat, P. Glasziou, M. K. Choong, A. Dunn, F. Galgani, and E. Coiera. Systematic review automation technologies. *Systematic Reviews*, 3(1):74, 2014.

[54] R. van de Schoot, J. De Bruin, R. Schram, P. Zahedi, J. De Boer, F. Weijdema, B. Kramer, M. Huijts, M. Hoogerwerf, G. Ferdinands, A. Harkema, J. Willemsen, Y. Ma, Q. Fang, S. Hindriks, L. Tummers, and D. L. Oberski. An open source machine learning framework for efficient and transparent systematic reviews. *Nature machine intelligence*, 3(2):125–133, 2021. doi: 10.1038/s42256-020-00287-7.

[55] R. van Dinter, B. Tekinerdogan, and C. Catal. Automation of systematic literature reviews: A systematic literature review. *Information and Software Technology*, 136, 2021. doi: https://doi.org/10.1016/j.infsof.2021.106589.

[56] B. C. Wallace, K. Small, C. E. Brodley, J. Lau, and T. A. Trikalinos. Deploying an interactive machine learning system in an evidence-based practice center: Abstrackr. In *International Health Informatics Symposium*, pages 819–824. ACM, 2012. doi: 10.1145/2110363.2110464.

[57] S. Wang, H. Scells, B. Koopman, and G. Zuccon. Can chatgpt write a good boolean query for systematic review literature search? Report 2302.03495, arXiv, feb 2023.

[58] M. Waseem, A. Ahmad, P. Liang, M. Fehmideh, P. Abrahamsson, and T. Mikkonen. Conducting systematic literature reviews with chatgpt. `https://tinyurl.com/bd4kfcta`, mar 2023.

[59] W. M. Watanabe, K. R. Felizardo, A. Candido, Érica Ferreira de Souza, J. E. de Campos Neto, and N. L. Vijaykumar. Reducing efforts of software engineering systematic literature reviews updates using text classification. *Information and Software Technology*, 128:106395, 2020. doi: https://doi.org/10.1016/j.infsof.2020.106395.

[60] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. Report 2201.11903, arXiV, jan 2023.

[61] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt. A prompt pattern catalog to enhance prompt engineering with chatgpt. Report 2302.11382, arXiv, feb 2023.

[62] D. Wilkins. Automated title and abstract screening for scoping reviews using the gpt-4 large language model, 2023.

[63] S. Xu, Y. Li, and Z. Wang. Bayesian multinomial naïve bayes classifier to text classification. In *Advanced Multimedia and Ubiquitous Engineering*, volume 352 of *LNEE*, pages 347–352. Springer, 2017.

[64] Z. Yu, L. He, Z. Wu, X. Dai, and J. Chen. Towards better chain-of-thought prompting strategies: A survey. Report 2310.04959, arXiv, 2023.

[65] X. Zeng and T. R. Martinez. Distribution-balanced stratified cross-validation for accuracy estimation. *Journal of Experimental & Theoretical Artificial Intelligence*, 12(1):1–12, 2000. doi: 10.1080/095281300146272.

[66] S. Zhang, L. Dong, X. Li, S. Zhang, X. Sun, S. Wang, J. Li, R. Hu, T. Zhang, F. Wu, and G. Wang. Instruction tuning for large language models: A survey, 2023.

[67] C. Zhou, Q. Li, C. Li, J. Yu, Y. Liu, G. Wang, K. Zhang, C. Ji, Q. Yan, L. He, H. Peng, J. Li, J. Wu, Z. Liu, P. Xie, C. Xiong, J. Pei, P. S. Yu, and L. Sun. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. Report 2302.09419, arXiv, mar 2023.