

AraREQ: A Dataset and End-to-End system for Conflict Detection and Resolution in Software Requirements

Tymaa Hammouda^λ, Alaa Aljabari^λ, Nagham Hamad^λ, Mustafa Jarrar^{σ,λ}

^λ Birzeit University, Palestine

^σ Hamad Bin Khalifa University, Qatar

{thammouda, aaljabari, nhamad, mjarrar}@birzeit.edu

Abstract

Conflict detection in software requirements is essential for ensuring specification consistency, improving project efficiency, and ensuring overall software quality. Despite its importance, research on this task, particularly for Arabic, remains limited due to the scarcity of annotated data and linguistic challenges. To address this gap, we introduce *AraREQ*, a large-scale Arabic dataset for requirement-level conflict detection and resolution. The dataset is constructed through a semi-automated *Arabization* process using Large Language Models (LLMs), followed by manual augmentation to address class imbalance. The final dataset comprises 27K Arabic requirement pairs. We benchmark four state-of-the-art LLMs under zero-shot and few-shot settings, establishing the first comprehensive evaluation for Arabic requirements conflict detection. Experimental results show that few-shot prompting consistently improves performance, particularly on the minority conflict class, demonstrating the effectiveness of example-based prompting. Finally, we introduce an end-to-end system that automatically detects potential conflicts in Arabic software requirements and generates resolution suggestions. All datasets, codes, and the end-to-end system are open-source and available at: <https://sina.birzeit.edu/ArReqConflicts/>

Keywords: Natural Language Processing, Large Language Models, Open-Source Dataset, Few-Shot Prompting, Zero-shot Prompting, Text Classification, Dataset Augmentation, Software Requirements, Requirements Engineering, Conflict Detection, Conflict Resolution.

1. Introduction

The Software Requirements Specification (SRS) is a fundamental artifact in the software development life cycle (Gärtner and Göhlich, 2024a). It formally defines both functional and non-functional requirements of a system (Helmeczi et al., 2022). As SRS documents evolve through iterative updates, inconsistencies and conflicts frequently emerge. If left unresolved, such conflicts can delay development, degrade software quality, and increase the risk of project failure (Aldekhail et al., 2016; Osman and Zaharin, 2018). Consequently, early detection and resolution of conflicting requirements are essential for successful project delivery.

Conflict detection is traditionally performed through expert review, where analysts examine requirement descriptions to identify ambiguities, contradictions, and logical inconsistencies. Despite its effectiveness for small-scale requirements documents, this process does not scale well to large, evolving specifications. Furthermore, subtle semantic conflicts may still be overlooked during manual analysis (Guo et al., 2021). These limitations highlight the need for automated, context-aware approaches to conflict detection and resolution. Unresolved conflicts can lead to costly delays, reduced software quality, and an increased risk of project failure. Detection and resolution of conflicting requirements are critical to project success. Recent advances in Large Language Models (LLMs) offer a promising direction toward addressing these challenges (Fantechi et al., 2023). Their ability to

capture complex semantics enables LLMs to identify logical contradictions and maintain consistency across large collections of requirements. Recent studies have further integrated formal logic with LLMs to support not only conflict detection but also automated resolution generation (Gärtner and Göhlich, 2024a). Nevertheless, existing resources and evaluations are predominantly English-centric, leaving automated conflict detection and resolution for low-resource languages, such as Arabic, largely unexplored.

Arabic poses additional challenges due to the language’s rich morphology and flexible syntactic structure, which often leads to lexical and semantic ambiguity (Hamad et al., 2025; Aljabari et al., 2025b; Alwajih et al., 2025a,b; Talafha et al., 2024). These characteristics make it difficult to accurately interpret requirement statements and identify subtle contradictions or inconsistencies between them. While Arabic has been widely studied in other areas—such as machine translation (Darwish et al., 2021), word sense disambiguation (Jarrar et al., 2023b; Al-Hajj and Jarrar, 2021), information extraction (Barbon Junior et al., 2024; Aljabari et al., 2024), language understanding (Khalilia et al., 2024), and interoperability (Jarrar et al., 2011)—the problem of automated conflict detection and resolution in Arabic software requirements remains largely unexplored.

To address these challenges, we introduce *AraREQ*, an *Arabized dataset*, and an end-to-end pipeline for automatic detection and resolution of conflicts in Arabic software requirements. The

dataset is constructed by adapting and extending four well-established English resources: PURE (Ferrari et al., 2017), WorldVista¹, OpenCoss (Larucea et al., 2013), and UAV (Ho et al., 2019). These datasets consist of requirement pairs labeled as either *conflict* or *neutral* (Malik et al., 2023a).

To construct *AraREQ*, we develop a semi-automated Arabization pipeline that transforms the selected English requirement pairs into high-quality Arabic counterparts. This process combines LLM-based translation using GPT-4o with expert validation by both a linguist and a software engineer to ensure linguistic accuracy and domain relevance. The resulting *AraREQ* dataset initially contains 26,462 *neutral* pairs and 83 *conflict* pairs, exhibiting severe class imbalance. To address this limitation, we introduce a targeted augmentation strategy to generate additional conflict instances. This strategy uses manually defined patterns such as negation, numerical modifications, and conditional alterations. The final dataset contains 26,462 *neutral* pairs and 547 *conflict* pairs, making *AraREQ*, to our knowledge, the largest publicly available Arabic dataset for conflict detection in requirements engineering.

We benchmark four LLMs on *AraREQ* under zero-shot and few-shot settings: Fanar (thinking mode) (Abbas et al., 2025), GPT-4o (Achiam et al., 2023), DeepSeek-Reasoner, and Llama-3.1-8B-Instruct. In addition, we develop the first end-to-end pipeline for conflict detection and resolution in Arabic software requirements. The system enables users to upload requirements in CSV format, then automatically identifies potential conflicts, and generates highlighted resolution suggestions through an interactive interface. This system demonstrates the practical applicability of LLM-based conflict analysis for Arabic requirements and is publicly available.

The main contributions of this paper are:

- *AraREQ*, the first Arabic software requirements dataset for conflict detection, consisting of 27K requirement pairs annotated with *Conflict* or *Neutral* labels.
- Benchmarking of state-of-the-art LLMs under zero-shot and few-shot settings.
- End-to-end system for detecting and resolving conflicts in software requirements.

The paper is structured as: Section 2 reviews the related work. Section 3 describes the dataset. Section 4 outlines the proposed end-to-end pipeline. Section 5 details the experimental setup and results. Section 6 discusses the findings and system behavior. Section 7 describes the End-to-End System. Finally, Section 8 concludes the paper.

¹<https://worldvista.org/Documentation>

2. Related Work

Several studies have explored automated conflict detection in software requirements using diverse methodologies. (Guo et al., 2021) propose a rule-based semantic model that detects conflicts by extracting key elements—such as operations, agents, events, and constraints—using part-of-speech tagging from CoreNLP. Experiments on the UAV, OpenCOSS, and WorldVista datasets reported accuracies above 91%, while evaluations on two private datasets (Solar Power and Telecom) achieve higher precision and recall. Similarly, Yuhana et al. (2024) integrate rule-based reasoning with clustering methods including k-means, agglomerative, and mean-shift, to enhance conflict detection. Their approach achieves F1-scores between 52% and 75% across public and private datasets.

Other approaches rely on hybrid and formal reasoning techniques. Malik et al. (2022) propose a two-stage framework combining semantic similarity thresholds with domain-specific named entity recognition, outperforming GPT-3.5 and Gemini across five annotated datasets. Similarly, Yatkin and Ovatman (2024); Gärtner and Göhlich (2024b) employ formal logic-based tools such as ANTLR4, SATSOLECS, and GPT-3 to automatically detect logical contradictions.

However, existing studies primarily focus on conflict detection in English requirements, while conflict resolution remains largely underexplored. The problem is further compounded for Arabic due to the lack of open, large-scale datasets for requirements engineering, which limits the development and benchmarking of automated methods for both conflict detection and resolution.

While several Arabic datasets have been developed for natural language processing tasks—such as named entity recognition (Hamad et al., 2025; Liqreina et al., 2023; Jarrar et al., 2023a), image captioning (Bashiti et al., 2025), dialectal morphology (Haff et al., 2022; Jarrar et al., 2023c; Nayouf et al., 2023; Jarrar and Hammouda, 2024), and semantic resources (Jarrar, 2021; Aljabari et al., 2025a)—these resources are not designed for software requirements analysis. Consequently, there remains a clear lack of publicly available Arabic datasets specifically tailored for software requirements engineering and related tasks such as conflict detection and resolution.

3. Dataset Construction

In this section, we introduce *AraREQ*, a large-scale Arabic dataset for software requirement conflict detection. The dataset is constructed through a semi-automated Arabization process that adapts and extends four existing English datasets annotated

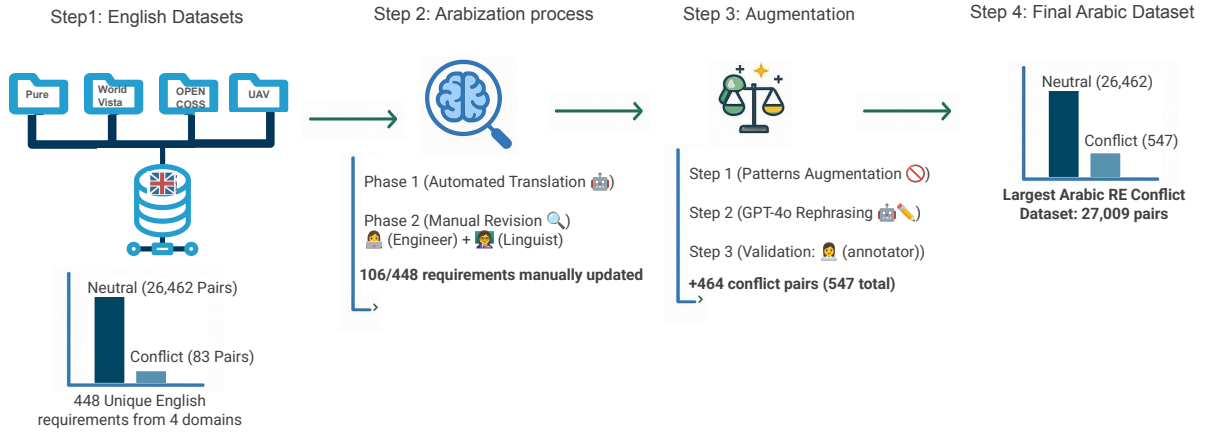


Figure 1: Dataset Construction Methodology

with conflict and neutral requirement pairs.

3.1. Utilized Datasets

We utilize four publicly available English datasets from the domain of software requirements engineering: PURE, WorldVista, OPENCROSS, and UAV. These datasets contain pairs of requirements annotated as either conflict or neutral (Malik et al., 2023a). Table 1 summarizes their statistics.

- **UAV** released by the University of Notre Dame in 2021. It contains requirement specifications for the control system of unmanned aerial vehicles.
- **WorldVista** includes software requirements for a healthcare management system designed to track patient information and hospital procedures.
- **PURE** is a publicly available corpus consisting of 79 SRS documents collected from open-source software projects.
- **OPENCROSS** open platform for the evolutionary certification of safety-critical systems, particularly in railway, avionics, and automotive domains. Prior work (Malik et al., 2022, 2023a,b; Saleem et al., 2025) highlights OPENCROSS as a challenging dataset for distinguishing between conflict and neutral requirement pairs due to substantial lexical overlap between the two classes.

3.2. Dataset Arabization

To construct *AraREQ*, we Arabize the aforementioned English datasets to create a domain-specific Arabic dataset for conflict detection. The Arabization process was conducted in two phases to ensure translation quality and maintain technical accuracy across all requirement statements.

Dataset	Unique Req.	Total Pairs	Neutral Pairs	Conflict Pairs
PURE	67	2,211	2,191	20
WorldVista	148	10,878	10,843	35
OPENCROSS	117	6,786	6,776	10
UAV	116	6,670	6,652	18
Total	448	26,545	26,462	83

Table 1: Statistics of English datasets used to construct *AraREQ* dataset.

Phase 1: Automated Translation. For the initial translation, we leverage GPT-4o, which has demonstrated strong performance in English-to-Arabic translation (Alyafeai et al., 2023). Each unique requirement is translated using a carefully designed prompt (see Appendix §A.2) that instructs the model to preserve the original technical meaning, modal verbs, and domain-specific terminology. In total, 26,545 requirement pairs are translated into Arabic, forming the preliminary version of the Arabized dataset for subsequent manual revision.

Phase 2: Manual Revision. To ensure linguistic and domain accuracy, we conduct a thorough manual review of all translated texts. This process is conducted collaboratively by a linguist and a software engineer to ensure both linguistic fluency and technical accuracy. During this revision process, several types of translation issues are identified and corrected.

First, **structural issues** are observed in cases where modal verbs such as *must* and *should*, which are essential for expressing constraints in software requirements, are omitted or mistranslated during automatic translation. For example, the requirement “The system must log all user actions.” is automatically translated as “يسجل النظام جميع إجراءات المستخدم.”, where the modal verb *must* is omitted. Such cases are manually corrected to preserve the intended obligation, resulting in the revised transla-

tion". يجب على النظام تسجيل جميع إجراءات المستخدم.

Second, **abbreviation-related issues** are encountered when domain-specific abbreviations (e.g., CPT, ICD) are either left untranslated or incorrectly translated. These cases are carefully reviewed and corrected by consulting technical lexicons and relevant documentation to ensure consistency and accurate representation.

Finally, **terminology inconsistencies** are identified when domain-specific terms are translated using multiple variants. For instance, the term thermostat appears in translations as both منظم حرارة / *mnẓm ḥrārḥ* and تيرموستات / *tyrmwstāt*. To maintain consistency across the dataset, such variations are unified using domain-specific lexicons and standardized terminology.

Table 2 presents the number of requirements that are updated during this manual revision process.

Dataset	# Arabized Requirements	# Manually Modified
PURE	67	26
WorldVista	148	18
OPENCOSS	117	51
UAV	116	11
Total	448	106

Table 2: Number of manually updated requirements after machine translation across datasets.

3.3. Addressing Class Imbalance

The four English datasets are inherently imbalanced, reflecting a real-world distribution where the majority of requirements do not contradict one another (Gärtner and Göhlich, 2024a). As shown in Table 1, the Conflict class is significantly under-represented across all datasets. To mitigate this imbalance, we design a multi-step augmentation procedure that increases the number of conflict-labeled requirement pairs while preserving semantic consistency and domain relevance.

Step 1: Defining Transformation Patterns. We first define a set of linguistic transformation patterns aimed at generating conflicting counterparts for the existing requirements. Each pattern modifies a specific linguistic feature, while preserving the structure of the requirement, ensuring that the resulting pair expresses a clear semantic contradiction. The designed patterns are summarized in Table 3.

Step 2: Conflict Pair Generation. Each pattern was applied to a selected set of Neutral requirements, resulting in new, semantically contradictory pairs labeled as Conflict. This process produce 547

Pattern	Description and Example
Negation	Introducing negative phrases or replacing words with their opposites; e.g., replacing يجب (must) with لا يجب (mustn't), or إيقاف تشغيل (turn off) with تشغيل (turn on).
Numerical Changes	Modifying numerical values while avoiding semantic overlap. For example, "إدخال الأوقات بصيغة ١٢ ساعة" (entering times in 12-hour format) is changed to "إدخال الأوقات بصيغة ٢٤ ساعة" (entering times in 24-hour format).
Conditional Alterations	Changing logical conditions within the requirement, such as replacing "greater than" (>) with "less than" (<).
Color Modification	Altering color references while keeping the rest of the requirement unchanged; e.g., "يجب على النظام إضاءة الأنوار باللون الأحمر" (the system must turn on the lights in red) becomes "باللون الأصفر" (in yellow).
Source Change	Substituting the data source mentioned in the requirement; e.g., replacing "قاعدة البيانات" (database) with "ملف الإعدادات" (settings file).

Table 3: Augmentation patterns used to generate additional conflict requirement pairs.

additional requirement pairs, increasing the total number to 27,009. The number of generated instances for each dataset is summarized in Table 4.

Dataset	# Original	# Augmented
PURE	20	79
WorldVista	35	151
OPENCOSS	10	118
UAV	18	116
Total	83	464

Table 4: Number of conflict requirements before and after augmentation across datasets.

Step 3: LLM-based Rephrasing. Because the generated contradictions may be overly explicit or syntactically simple, we further rephrase the newly created requirements using GPT-4o. This step introduces linguistic variation, making the contradictions more natural and reducing the likelihood of trivial pattern-based detection.

Step 4: Manual Verification. Finally, all generated pairs are manually reviewed by annotators. Each pair is examined to ensure that (1) it truly contradicts the original requirement, and (2) it still represents true semantic contradictions with respect to the original requirements. The final dataset statistics are presented in Table 5.

Dataset	Neutral pairs	Conflict pairs
AraPURE	2,191	99
AraWorldVista	10,843	186
AraOPENCROSS	6,776	128
AraUAV	6,652	134
Total	26,462	547

Table 5: *AraREQ* Dataset Statistics

4. Conflict Detection and Resolution

Conflict detection and resolution are critical steps in requirements engineering, as unresolved contradictions can propagate errors throughout the software development lifecycle. Traditionally, identifying such conflicts relies on extensive manual analysis by domain experts, making the process time-consuming and error-prone. To address these challenges, we leverage *AraREQ* dataset to develop the first end-to-end pipeline for automatic detection and resolution of conflicts in Arabic software requirements. The pipeline employs LLMs for both detection and resolution tasks.

4.1. Conflict Detection

We formulate conflict detection as a binary Natural Language Inference (NLI) task, aiming to determine whether a pair of requirement statements is contradictory (conflict) or non-contradictory (neutral).

To model this task, we employ LLMs in both zero-shot and few-shot settings, which have demonstrated strong generalization capabilities across reasoning and inference tasks (Madaan et al., 2025). The conflict detection pipeline consists of two main steps: **(1)** retrieval of semantically relevant examples to provide contextual guidance; and **(2)** LLM prompting to classify the input pair.

Step 1: Semantically-Guided Requirement Retrieval: To guide LLMs in conflict detection, semantically similar examples are dynamically retrieved from external knowledge for each input requirement pair. In *AraREQ*, each of the four datasets corresponds to a distinct software system; therefore, when evaluating one system, the remaining three are used as external knowledge sources to retrieve relevant examples. The semantic representation of each requirement pair is computed using a pretrained BERT model (Antoun et al., 2020), and candidate examples are ranked based on cosine similarity. The top- k most similar examples from both *conflict* and *neutral* classes are then selected to form a contextually relevant demonstration set.

Step 2: Conflict classification: A contextual prompt \mathcal{C} is constructed to guide the LLM in classifying a pair of Arabic software requirements as either *conflict* or *neutral*. Given a requirement pair

\mathcal{P} , a set of retrieved examples \mathcal{E} , and a task-specific instruction \mathcal{I} , the complete prompt is defined as:

$$\mathcal{C} = (\mathcal{I}, \mathcal{E}, \mathcal{P})$$

The LLM is then instructed to generate a textual label Y , where $Y \in \{\text{"conflict"}, \text{"neutral"}\}$.

Since modern LLMs operate as autoregressive language models, the output label is generated token-by-token. Let $Y = (y_1, y_2, \dots, y_n)$ denote the sequence of tokens forming the predicted label, where y_t represents the token generated at time step t , and n is the number of tokens in the label. The conditional probability of generating Y given the prompt context is therefore expressed as:

$$p(Y | \mathcal{P}, \mathcal{E}, \mathcal{I}) = \prod_{t=1}^n p(y_t | \mathcal{P}, \mathcal{E}, \mathcal{I}, y_{<t}) \quad (1)$$

where $y_{<t}$ denotes the sequence of previously generated tokens.

In practice, the final prediction corresponds to the generated label text (“conflict” or “neutral”). The probabilistic formulation above reflects the underlying generative mechanism of the LLM rather than a multi-token classification structure. In the zero-shot setting, the same formulation is applied without including the retrieved example set \mathcal{E} . Prompt templates for both zero-shot and few-shot configurations are provided in Appendix A.1.

4.2. Conflict Resolution

Once conflicts are detected, resolving them is necessary to maintain the internal consistency and feasibility of software requirements. To support this process, we categorize the identified conflicts into two types—*inconsistency* or *incompatibility*—as each requires a different resolution strategy.

Inconsistency arises when two requirements describe the same functionality with overlapping or partially redundant constraints. Such conflicts can typically be resolved by merging the statements into a single consistent requirement that preserves their shared intent within the given context. For example, as shown in Table 6, the inconsistency arises from the restrictive quantifier “only” in the first requirement, which conflicts with the broader set of options specified in the other.

Incompatibility, in contrast, arises when requirements define conditions that cannot be satisfied at the same time without changing their intended meaning. As shown in Table 6, resolving such conflicts requires adding contextual constraints that clarify when each requirement should apply. For instance, the requirements (يجب على النظام دعم الوضع الداكن. / The system shall support dark mode) and (يجب أن يكون اللون الأساسي لواجهة المستخدم هو الأبيض. / The primary color of the user interface shall be white) can

Conflict Type	Requirement 1 (R1)	Requirement 2 (R2)
Inconsistency	يجب أن يدعم النظام تنزيل الملفات بصيغة PDF فقط. The system shall support download files in PDF format only.	يجب أن يدعم النظام تنزيل الملفات بصيغ متعددة PDF أو DOCX أو XLSX. The system shall support download files in multiple formats, including PDF, DOCX, or XLSX.
	Resolution: يجب أن يدعم النظام تنزيل الملفات بصيغ متعددة PDF أو DOCX أو XLSX. The system shall support download files in multiple formats, including PDF, DOCX, or XLSX.	
Incompatible	لون السمة يجب أن يكون أبيض. The theme color shall be white.	لون السمة يجب أن يكون أسود. The theme color shall be black.
	R1 Resolution: يجب أن يكون لون السمة هو الأبيض بشكل افتراضي. The theme color shall be white by default. R2 Resolution: يجب أن يكون لون السمة هو الأسود عند تفعيل الوضع الداكن. The theme color shall be black when dark mode is enabled.	

Table 6: Examples of inconsistency and incompatible requirement conflicts with corresponding resolutions.

be reconciled by specifying different interface colors depending on the system mode.

Resolving both types of conflicts requires context-aware reasoning over the requirement statements and the system specification. To address this challenge, we propose an LLM-based approach that leverages both the conflicting requirement pair and the broader system context to generate appropriate resolutions. To the best of our knowledge, this work presents the first systematic framework for context-aware conflict resolution in requirements engineering. The proposed process consists of four steps:

- 1. Input.** The process begins with a conflicting requirement pair, along with the complete set of system requirements provided as contextual information.
- 2. Resolution Generation.** The LLM is instructed to either (a) *merge* the conflicting requirements into a single consistent statement that preserves the intent of both, when feasible, or (b) *rephrase* them into separate non-conflicting requirements when merging is not possible.
- 3. Consistency Verification.** The generated resolution is evaluated against the entire set of system requirements to ensure overall consistency. This step includes automated conflict detection to verify that no additional contradictions are introduced.
- 4. Human Validation.** The generated resolutions are presented to system analysts or stakeholders as recommendations, who retain the authority to accept, modify, or reject the proposed changes.

4.3. Evaluation Criteria for Conflict Resolution

To assess the quality of the generated conflict resolutions, we define a set of evaluation criteria for expert review. Each generated resolution is evaluated according to the following criteria:

- **Conflict Elimination:** Measures whether the generated resolution successfully removes the contradiction between the conflicting requirements. This criterion is binary: 0 indicates that the conflict remains unresolved, while 1 indicates that the conflict has been resolved.
- **Linguistic Correctness:** Evaluates the fluency and grammatical correctness of the generated requirement in formal Arabic. This criterion is rated on a 1–5 scale, where 1 indicates poor fluency or major grammatical errors and 5 indicates grammatically correct and fluent expression.
- **Ambiguity:** Assesses the clarity of the generated resolution and its adherence to requirements engineering best practices, ensuring that vague expressions such as “as needed” or “if possible” are not used. This criterion is rated on a 1–5 scale, where 1 indicates high ambiguity, and 5 indicates a fully precise and unambiguous formulation.

5. Experiments and Results

We implement and evaluate the proposed conflict detection and resolution pipeline described in §4 using the *AraREQ* dataset. We utilize four LLMs: the proprietary GPT-4o and three open-source models, Fanar, LLaMA, and DeepSeek. For conflict detection, each model is evaluated under both zero-shot and few-shot. The temperature is set to 0.0 to ensure deterministic predictions, as the task involves

Dataset	Model	Zero-shot				Few-shot			
		Micro	Macro	Conflict	Neutral	Micro	Macro	Conflict	Neutral
AraPURE	GPT4	98.5	91.0	91.8	99.5	98.5	91.1	92.4	99.5
	Deepseek	95.9	79.1	83.5	98.5	96.7	82.0	84.2	98.9
	Llama	83.9	54.1	55.5	92.4	86.0	58.5	69.7	93.3
	Fanar	94.2	74.4	84.2	97.5	94.2	74.7	84.9	97.5
AraWorldVista	GPT4	99.3	90.2	94.9	99.7	99.5	93.2	95.2	99.8
	Deepseek	97.2	74.4	90.9	98.7	97.3	75.2	91.5	98.8
	Llama	73.6	45.1	62.7	85.1	83.5	52.1	86.2	91.1
	Fanar	96.5	71.6	91.5	98.3	97.3	75.0	91.5	98.7
AraOPENCROSS	GPT4	99.5	93.6	97.2	99.8	99.5	93.9	97.6	99.8
	Deepseek	99.3	90.9	92.9	99.8	99.3	91.6	94.2	99.8
	Llama	75.2	46.3	66.7	86.2	76.7	47.4	72.0	87.0
	Fanar	96.6	71.9	87.7	98.4	96.7	72.9	89.7	98.5
AraUAV	GPT4	99.6	94.6	95.7	99.8	99.7	96.4	96.1	99.9
	Deepseek	99.2	88.9	86.9	99.8	99.2	90.0	88.3	99.8
	Llama	80.9	48.8	61.1	89.9	81.3	49.3	64.0	90.1
	Fanar	97.4	77.4	91.5	98.8	97.4	77.9	92.8	98.8

Table 7: Performance of four LLMs on the *AraREQ* datasets under zero-shot and few-shot settings. Metrics include Micro F1, Macro F1, and per-class F1 scores (Conflict and Neutral).

categorical classification rather than open-ended generation.

For conflict resolution, we adopt the best-performing model from the conflict detection experiments. We set the temperature to 0.2 to allow controlled variability in the generated resolutions. Across all experiments, the maximum token limit is 1000, top- p is 1.0, and both frequency and presence penalties are set to 0.0 to prevent bias in lexical choice or unnecessary repetition.

5.1. Conflict Detection Results

We evaluate the four LLMs using micro-F1, macro-F1, and per-class F1 metrics. The micro-F1 score serves as the primary evaluation metric, providing an overall measure of performance across all instances. However, since the dataset is class-imbalanced—with *conflict* pairs being less frequent than *neutral* ones—the macro-F1 and per-class F1 scores are also reported to better capture model behavior.

As shown in Table 7, detecting *conflict* pairs is consistently more challenging than identifying *neutral* ones, as it requires deeper semantic and contextual reasoning to capture logical inconsistencies between lexically similar requirements. Among all evaluated models, GPT-4o achieves the best performance in both zero-shot and few-shot settings, with average macro-F1 scores of 92.35 and 94.45, respectively, outperforming all open-source counterparts.

In the few-shot setting, providing examples from both classes (conflict and neutral) in the prompt improves conflict detection performance, particularly for the minority *conflict* class, while having only a

marginal effect on the *neutral* class. In addition, our results indicate that the ordering of examples within the prompt affects performance: placing the *Neutral* example before the *Conflict* example yields more stable and competitive results.

5.2. Conflict resolution results

Based on the evaluation results reported in §5.1, we select GPT-4o for conflict resolution, following the strategy outlined in §4.2. To ensure that the generated resolutions are correct and preserve the intended meaning of the original requirements, all generated resolutions—either merged or rephrased requirements—are manually reviewed by a software engineering expert using the evaluation criteria defined in § 4.3.

Table 8 summarizes the evaluation results. The first three metrics—Linguistic Correctness, Ambiguity, and Conflict Elimination—are manually assessed for *each conflict pair*. The results indicate that GPT-4o consistently generates fluent and clear requirements while resolving most detected conflicts with minimal ambiguity.

To evaluate consistency at the system level, we additionally report Accuracy, which measures whether the resolved requirements introduce new conflicts with other requirements in the system. This metric is computed automatically using GPT-4 by comparing each resolved requirement with all other requirements in the dataset. Ideally, after applying the resolution step, all resulting requirement pairs should be classified as neutral. The results show that GPT-4o effectively preserves global consistency across the requirement set. Table 9 further reports the distribution of generated resolution

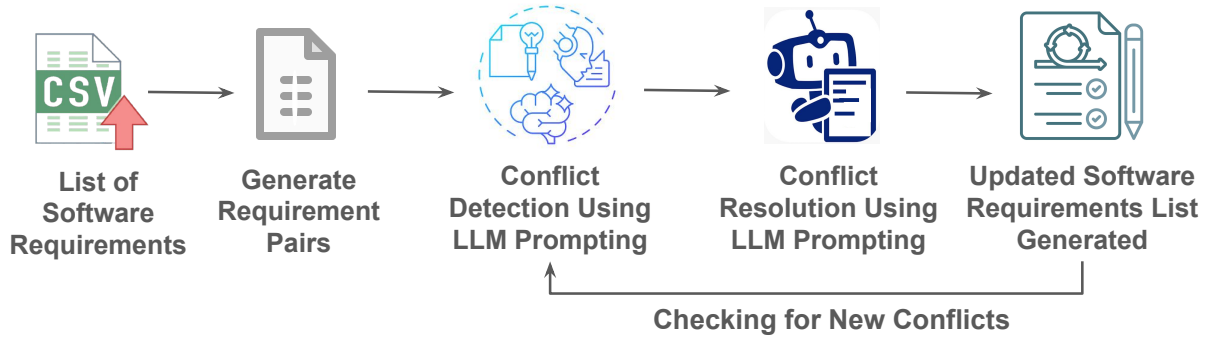


Figure 2: End-to-end conflict detection and resolution system.

Dataset	Ling. Corr.	No Ambig.	Conf. Elim.	Acc.
AraPURE	4.95	4.32	96.0	97.9
AraWorldVista	4.72	4.34	96.9	98.4
AraOPENCROSS	4.63	4.33	96.7	98.9
AraUAV	4.15	3.95	95.0	99.2

Table 8: Manual evaluation of GPT-4o conflict resolution on the *AraREQ* datasets. All values represent average expert scores.

Dataset	Conflict Pairs	Merge Pairs	Rephrase Pairs
AraPURE	99	64	35
AraWorldVista	186	41	145
AraOPENCROSS	128	14	114
AraUAV	134	14	120
Total	547	133	414

Table 9: Distribution of conflict resolution types generated by GPT-4o across the *AraREQ* dataset.

types, including merged and rephrased requirements.

6. Discussion

To better understand the experimental results, we analyze common error patterns and limitations of LLMs in conflict detection. Despite their strong overall performance, several recurring challenges were observed. First, requirements involving mathematical or numerical constraints remain difficult, particularly when logical conditions are expressed indirectly or abstractly. Second, the models exhibit limitations in handling conditional logic. Requirements containing relational operators such as “<” and “>” often lead to false conflict predictions, even when the conditions are logically compatible. Third, the models often fail to recognize the differences between modalities, such as “must” versus “will”, which significantly alter the level of obligation in a requirement. These limitations reflect the broader difficulty LLMs face in performing fine-grained logi-

cal reasoning, which is critical in software requirements engineering.

These gains validate the importance of In terms of performance, few-shot prompting led to measurable improvements across all models. example-driven prompting in reinforcing the model’s understanding of domain-specific contradictions. They also illustrate that even a small number of well-selected examples can significantly enhance performance.

7. End-to-End System

We develop a web-based platform that implements the proposed pipeline for automated conflict detection and resolution in Arabic software requirements. The system is implemented using *AraREQ* and it follows the workflow illustrated in Figure 2, and the interface is shown in Figure 3. The platform supports multiple large language models, including Fanar, DeepSeek-Reasoner, and GPT-4o, allowing users to select the model used for analysis. Also, the platform supports zero-shot or few-shot prompting settings options.

The workflow starts with requirement ingestion, where users can try set of requirements or upload a CSV file containing Arabic requirement statements. The system then automatically constructs requirement pairs using a combination-based strategy that pairs each requirement with subsequent ones, ensuring full coverage while avoiding redundant comparison.

Each requirement pair is then processed by the conflict detection module, which classifies the pair as either conflict or neutral. As depicted in Figure 2.

Finally, the detected conflicts are displayed in the platform interface, allowing users to resolve the conflict, and easily review contradictions and validate the results efficiently. Since the end-to-end system relies on the methodology described in § 4, the experimental results reported in Section 5 reflect the performance of the deployed system.

Requirement Conflict Detection and Resolution

Model
Fanar

Prompt Strategy
Zero-shot

Input Method
Type Two Requirement sets

Requirements

لون السمة يجب أن يكون أبيض
لون السمة يجب أن يكون أسود
يجب أن يدعم النظام إمكانية تنزيل ملفات بصيغة PDF
يجب أن يدعم النظام إمكانية تنزيل ملفات متعددة بأنواع PDF, EXCEL

Detect Conflict

ID	Requirement 1	Requirement 2	Result	Action
1-2	لون السمة يجب أن يكون أبيض	لون السمة يجب أن يكون أسود	:Split - Resolved Requirements لون السمة الافتراضي هو الأبيض. يمكن للمستخدم اختيار تغيير لون السمة إلى الأسود إذا رغب في ذلك.	Resolved
3-4	يجب أن يدعم النظام إمكانية تنزيل ملفات متعددة بأنواع PDF, EXCEL	يجب أن يدعم النظام إمكانية تنزيل ملفات بصيغة PDF	{resp: "Yes", "statusText": "OK", "statusCode": "0"}	Resolve

Figure 3: Snapshot of the web-based platform for automated conflict detection and resolution in Arabic software requirements.

8. Conclusion

This paper introduced *AraREQ*, the first large-scale Arabic dataset for conflict detection and resolution in software requirements. The dataset was constructed through a semi-automated Arabization process applied to four established English requirements datasets, followed by manual validation and targeted augmentation to address class imbalance.

Using *AraREQ*, we benchmarked four LLMs: GPT-4o, Fanar, LLaMA, and DeepSeek, under both zero-shot and few-shot prompting settings for conflict detection. The results demonstrate that LLMs can effectively identify conflicts in Arabic requirements, with few-shot prompting consistently improving performance, particularly for the minority conflict class. Building on these findings, we propose an end-to-end pipeline for automated conflict detection and resolution and implement a web-based platform that operationalizes the approach.

The proposed platform demonstrates the practical applicability of our method by enabling users to upload requirements, automatically detect conflicts, and generate resolutions. Overall, *AraREQ* establishes a new benchmark for Arabic requirements engineering and supports future research on automated conflict detection and resolution. Fu-

ture work will focus on expanding the dataset, improving reasoning capabilities for complex logical conditions, and exploring hybrid approaches that integrate LLMs with formal reasoning techniques.

9. Acknowledgments

We would like to acknowledge the linguist Raghad Jayyousi for her significant effort in manually reviewing the *AraREQ* dataset.

Limitations

Despite the promising results achieved with *AraREQ*, two main limitations remain. First, the dataset is constructed from four publicly available sources, each representing specific application domains. As a result, the diversity of requirement structures and domain contexts may still be limited compared to the wide variety of requirements encountered in real-world software systems. This limitation largely stems from the scarcity of publicly available requirements datasets. Software requirements often contain sensitive information related to system functionality, business logic, and internal processes, which makes organizations reluctant to release such documents publicly.

Second, the limited availability of datasets also constrains benchmarking. The absence of widely adopted and diverse benchmark datasets makes it difficult to conduct large-scale comparative evaluations across different approaches. Consequently, although our experiments provide insights into the effectiveness of LLMs for detecting and resolving conflicts in Arabic software requirements, it remains challenging to position these results within a broader and standardized evaluation landscape.

10. Bibliographical References

- Ummar Abbas, Mohammad Shahmeer Ahmad, Firoj Alam, Enes Altinisik, Ehsannedin Asgari, Yazan Boshmaf, Sabri Boughorbel, Sanjay Chawla, Shammur Chowdhury, et al. 2025. Fannar: An arabic-centric multimodal generative ai platform. *arXiv preprint arXiv:2501.13944*.
- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altmenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Moustafa Al-Hajj and Mustafa Jarrar. 2021. [Arab-GlossBERT: Fine-Tuning BERT on Context-Gloss Pairs for WSD](#). In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 40–48, Online. INCOMA Ltd.
- Maysoon Aldekhail, Azzedine Chikh, and Djamel Ziani. 2016. Software requirements conflict identification: review and recommendations. *Int. J. Adv. Comput. Sci. Appl*, 7(10):326–335.
- Alaa Aljabari, Lina Duaibes, Mustafa Jarrar, and Mohammed Khalilia. 2024. [Event-Arguments Extraction Corpus and Modeling using BERT for Arabic](#). In *Proceedings of the Second Arabic Natural Language Processing Conference (ArabicNLP 2024)*, Bangkok, Thailand. Association for Computational Linguistics.
- Alaa Aljabari, Nagham Hamad, Mohammed Khalilia, and Mustafa Jarrar. 2025a. [WojoodOntology: Ontology-Driven LLM Prompting for Unified Information Extraction Tasks](#). In *Proceedings of the Third Arabic Natural Language Processing Conference (ArabicNLP)*, Suzhou, China. Association for Computational Linguistics.
- Alaa Aljabari, Mohammed Khalilia, and Mustafa Jarrar. 2025b. [Wojood^{Relations}: Arabic Relation Extraction Corpus and Modeling](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, Suzhou, China. Association for Computational Linguistics.
- Fakhraddin Alwajih, Samar M. Magdy, Abdellah El Mekki, Omer Nacar, Youssef Nafea, Safaa Taher Abdelfadil, Abdulfattah Mohammed Yahya, Hamzah Luqman, Nada Almarwani, Samah Aloufi, Baraah Qawasmeh, Houdaifa Atou, Serry Sibae, Hamzah A. Alsayadi, Walid Al-Dhabyani, Maged S. Al-shaibani, Aya El Aatar, Nour Qandos, Rahaf Alhamouri, Samar Ahmad, Mohammed Anwar Al-Ghrawi, Amine-tou Yacoub, Ruwa AbuHweidi, Vatimetou Mohamed Lemin, Reem Abdel-Salam, Ahlam Bashiti, Aisha Alansari, Ahmed Ashraf, Nora Alturayef, Alcides Alcoba Inciarte, Adel Ammar, Abdelrahim A. Elmadany, Mohamedou Cheikh Tourad, Ismail Berrada, Mustafa Jarrar, Shady Shehata, and Muhammad Abdul-Mageed. 2025a. [Pearl: A Multimodal Culturally-Aware Arabic Instruction Dataset](#). In *Proceedings of the Third Arabic Natural Language Processing Conference (ArabicNLP)*, Suzhou, China. Association for Computational Linguistics.
- Fakhraddin Alwajih, Abdellah El Mekki, Samar Mohamed Magdy, Abdelrahim A. Elmadany, Omer Nacar, El Moatez Billah Nagoudi, Reem Abdel-Salam, Hanin Atwany, Youssef Nafea, Abdulfattah Mohammed Yahya, Rahaf Alhamouri, Hamzah A. Alsayadi, Hiba Zayed, Sara Shatnawi, Serry Sibae, Yasir Ech-Chammakhy, Walid Al-Dhabyani, Marwa Mohamed Ali, Imen Jarraya, Ahmed Oumar El-Shangiti, Aisha Alraeesi, Mohammed Anwar Al-Ghrawi, Abdulrahman S. Al-Batati, Elgizouli Mohamed, Noha Taha Elgindi, Muhammed Saeed, Houdaifa Atou, Issam Ait Yahia, Abdelhak Bouayad, Mohammed Machrouh, Amal Makouar, Dania Alkawi, Mukhtar Mohamed, Safaa Taher Abdelfadil, Amine Ziad Ounnoughene, Rouabhia Anfel, Rwa Assi, Ahmed Sorkatti, Mohamedou Cheikh Tourad, Anis Koubaa, Ismail Berrada, Mustafa Jarrar, Shady Shehata, and Muhammad Abdul-Mageed. 2025b. [Palm: A Culturally Inclusive and Linguistically Diverse Dataset for Arabic LLM](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 0–0, Vienna, Austria. Association for Computational Linguistics.
- Zaid Alyafeai, Maged Saeed AlShaibani, Badr AlKhamissi, Hamzah Luqman, Ebrahim Alareqi, and Ali Fadel. 2023. [Taqyim: Evaluating arabic NLP tasks using chatgpt models](#). *CoRR*, abs/2306.16322.
- Wissam Antoun, Fady Baly, and Hazem M. Hajj.

2020. [Arabert: Transformer-based model for arabic language understanding](#). *CoRR*, abs/2003.00104.
- Sylvio Barbon Junior, Paolo Ceravolo, Sven Groppe, Mustafa Jarrar, Samira Maghool, Florence Sèdes, Soror Sahri, and Maurice Van Keulen. 2024. [Are Large Language Models the New Interface for Data Pipelines?](#) In *Proceedings of the International Workshop on Big Data in Emergent Distributed Environments*, BiDEDE '24, New York, NY, USA. Association for Computing Machinery.
- Ahlam Bashiti, Alaa Aljabari, Hadi Hamoud, Md. Rafiul Biswas, Bilal Shalash, Mustafa Jarrar, Fadi Zaraket, and George Mikros. 2025. [ImageEval 2025: The First Arabic Image Captioning Shared Task](#). In *Proceedings of the Third Arabic Natural Language Processing Conference (ArabicNLP)*, Suzhou, China. Association for Computational Linguistics.
- Kareem Darwish, Nizar Habash, Mourad Abbas, Hend Al-Khalifa, Huseein T. Al-Natsheh, Houda Bouamor, Karim Bouzoubaa, Violetta Cavall-Sforza, Samhaa R. El-Beltagy, Wassim El-Hajj, Mustafa Jarrar, and Hamdy Mubarak. 2021. [A Panoramic survey of Natural Language Processing in the Arab Worlds](#). *Commun. ACM*, 64(4):72–81.
- Alessandro Fantechi, Stefania Gnesi, Lucia Passaro, and Laura Semini. 2023. Inconsistency detection in natural language requirements using chatgpt: a preliminary evaluation. In *2023 IEEE 31st International Requirements Engineering Conference (RE)*, pages 335–340. IEEE.
- Alessio Ferrari, Giorgio Ortonzo Spagnolo, and Stefania Gnesi. 2017. Pure: A dataset of public requirements documents. In *2017 IEEE 25th international requirements engineering conference (RE)*, pages 502–505. IEEE.
- Alexander Elenga Gärtner and Dietmar Göhlich. 2024a. Automated requirement contradiction detection through formal logic and llms. *Automated Software Engineering*, 31(2):49.
- Alexander Elenga Gärtner and Dietmar Göhlich. 2024b. Towards an automatic contradiction detection in requirements engineering. *Proceedings of the Design Society*, 4:2049–2058.
- Weize Guo, Li Zhang, and Xiaoli Lian. 2021. Automatically detecting the conflicts between software requirements based on finer semantic analysis. *arXiv preprint arXiv:2103.02255*.
- Karim El Haff, Mustafa Jarrar, Tymaa Hammouda, and Fadi Zaraket. 2022. [Curras + Baladi: Towards a Levantine Corpus](#). In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2022)*, Marseille, France.
- Naghm Hamad, Mohammed Khalilia, and Mustafa Jarrar. 2025. [Konooz: Multi-domain Multi-dialect Corpus for Named Entity Recognition](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, page 7316–7331, Vienna, Austria. Association for Computational Linguistics.
- Robert K Helmecci, Mucahit Cevik, and Savas Yildirim. 2022. A prompt-based few-shot learning approach to software conflict detection. *arXiv preprint arXiv:2211.02709*.
- Florence Ho, Ruben Geraldès, Artur Gonçalves, Bastien Rigault, Atsushi Oosedo, Marc Cavazza, and Helmut Prendinger. 2019. Pre-flight conflict detection and resolution for uav integration in shared airspace: Sendai 2030 model case. *IEEE Access*, 7:170226–170237.
- Mustafa Jarrar. 2021. [The Arabic Ontology - An Arabic Wordnet with Ontologically Clean Content](#). *Applied Ontology Journal*, 16(1):1–26.
- Mustafa Jarrar, Muhammad Abdul-Mageed, Mohammed Khalilia, Bashar Talafha, AbdelRahim Elmadany, Nagham Hamad, and Alaa' Omar. 2023a. [WojoodNER 2023: The First Arabic Named Entity Recognition Shared Task](#). In *Proceedings of the 1st Arabic Natural Language Processing Conference (ArabicNLP), Part of the EMNLP 2023*, pages 748–758. ACL.
- Mustafa Jarrar, Anton Deik, and Bilal Faraj. 2011. [Ontology-based data and process governance framework -the case of e-government interoperability in palestine](#). In *Proceedings of the IFIP International Symposium on Data-Driven Process Discovery and Analysis (SIMPDA'11)*, pages 83–98.
- Mustafa Jarrar and Tymaa Hasanain Hammouda. 2024. [Qabas: An Open-Source Arabic Lexicographic Database](#). In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 13363–13370, Torino, Italy. ELRA and ICCL.
- Mustafa Jarrar, Sanad Malaysha, Tymaa Hammouda, and Mohammed Khalilia. 2023b. [SALMA: Arabic Sense-annotated Corpus and WSD Benchmarks](#). In *Proceedings of the 1st Arabic Natural Language Processing*

- Conference (ArabicNLP), Part of the EMNLP 2023, pages 359–369. ACL.
- Mustafa Jarrar, Fadi Zaraket, Tymaa Hammouda, Daanish Masood Alavi, and Martin Waahlsch. 2023c. [Lisan: Yemeni, Irqi, Libyan, and Sudanese Arabic Dialect Copora with Morphological Annotations](#). In *The 20th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*, pages 1–7. IEEE.
- Mohammed Khalilia, Sanad Malaysha, Reem Suwaileh, Mustafa Jarrar, Alaa Aljabari, Tamer Elsayed, and Imed Zitouni. 2024. [ArabicNLU 2024: The First Arabic Natural Language Understanding Shared Task](#). In *Proceedings of the Second Arabic Natural Language Processing Conference (ArabicNLP 2024)*, Bangkok, Thailand. Association for Computational Linguistics.
- Xabier Larrucea, Annie Combelles, and John Favaro. 2013. Safety-critical software [guest editors' introduction]. *IEEE Software*, 30(3):25–27.
- Haneen Liqreina, Mustafa Jarrar, Mohammed Khalilia, Ahmed Oumar El-Shangiti, and Muhammad Abdul-Mageed. 2023. [Arabic Fine-Grained Entity Recognition](#). In *Proceedings of the 1st Arabic Natural Language Processing Conference (ArabicNLP), Part of the EMNLP 2023*, pages 310–323. ACL.
- Lovish Madaan, David Esiobu, Pontus Stenetorp, Barbara Plank, and Dieuwke Hupkes. 2025. [Lost in inference: Rediscovering the role of natural language inference for large language models](#). In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 9229–9242, Albuquerque, New Mexico. Association for Computational Linguistics.
- Garima Malik, Mucahit Cevik, and Ayşe Başar. 2023a. Data augmentation for conflict and duplicate detection in software engineering sentence pairs. *arXiv preprint arXiv:2305.09608*.
- Garima Malik, Mucahit Cevik, Devang Parikh, and Ayse Basar. 2022. Supervised semantic similarity-based conflict detection algorithm: S3cda. *arXiv preprint arXiv:2206.13690*.
- Garima Malik, Savas Yildirim, Mucahit Cevik, Ayse Bener, and Devang Parikh. 2023b. Transfer learning for conflict and duplicate detection in software requirement pairs. *arXiv preprint arXiv:2301.03709*.
- Amal Nayouf, Mustafa Jarrar, Fadi zaraket, Tymaa Hammouda, and Mohamad-Bassam Kurdy. 2023. [Nàbra: Syrian Arabic Dialects with Morphological Annotations](#). In *Proceedings of the 1st Arabic Natural Language Processing Conference (ArabicNLP), Part of the EMNLP 2023*, pages 12–23. ACL.
- Mohd Hafeez Osman and Mohd Firdaus Zaharin. 2018. Ambiguous software requirement specification detection: An automated approach. In *Proceedings of the 5th International Workshop on Requirements Engineering and Testing*, pages 33–40.
- Summra Saleem, Muhammad Nabeel Asim, and Andreas Dengel. 2025. Passionnet: An innovative framework for duplicate and conflicting requirements identification. *Expert Systems with Applications*, page 128684.
- Bashar Talafha, Karima Kadaoui, Samar Mohamed Magdy, Mariem Habiboullah, Chafei Mohamed Chafei, Ahmed Oumar El-Shangiti, Hiba Zayed, Mohamedou Cheikh Tourad, Rahaf Alhamouri, Rwa Assi, Aisha Alraeesi, Hour Mohamed, Fakhraddin Alwajih, Abdelrahman Mohamed, Abdallah El Mekki, El Moatez Billah Nagoudi, Benelhadj Djelloul Mama Saadia, Hamzah A. Alsayadi, Walid Al-Dhabyani, Sara Shatnawi, Yasir Ech-chammakhy, Amal Makouar, Yousra Berrachedi, Mustafa Jarrar, Shady Shehata, Ismail Berrada, and Muhammad Abdul-Mageed. 2024. [Casablanca: Data and Models for Multidialectal Arabic Speech Recognition](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Miami, Florida, USA. Association for Computational Linguistics.
- Simge Yatkin and Tolga Ovatman. 2024. Logical analysis and contradiction detection in high-level requirements during the review process using sat-solver. *arXiv preprint arXiv:2405.00163*.
- Umi Laili Yuhana, Siti Rochimah, et al. 2024. Conflict detection of functional requirements based on clustering and rule-based system. *IEEE Access*.

A. Appendix

This section provides a full set of prompts used throughout the pipeline:

A.1. Conflict Detection Prompt

Used to classify a pair of Arabic software requirements as either conflicting or neutral.

Contradiction Detection Prompt (zero-shot)

Do the following sentences contradict each other, **yes** or **no**?

1. <requirement_1>
2. <requirement_2>

Contradiction Detection Prompt (few-shot)

Do the following sentences contradict each other, (**yes**) or (**no**):

sentence 1: <requirement_1>
sentence 2: <requirement_2>
class: <label>

Now apply the same and classify the following pair:

sentence 1: <most similar neutral requirement>
sentence 2: <most similar conflict requirement>
class: ...

A.2. Translation Prompt

Converts English software requirements into Modern Standard Arabic.

Translation Prompt

Translate the following text into Modern Standard Arabic: <requirement>

A.3. Rephrasing Prompt

Enhances clarity and formality of Arabic requirements by rewording them without altering their original meaning.

Rephrasing Prompt

Role: You are a helpful assistant that rephrases Arabic sentences to make contradictions less obvious while preserving meaning.

Given the following sentence that expresses a logical contradiction to an original statement, rephrase it to make the conflict less obvious or straightforward, using varied vocabulary and sentence structure, while preserving the core opposing meaning: <requirement>

A.4. Conflict Resolution Prompt

Prompts the model to resolve conflicting requirements by merging them into one or rephrasing them into two non-conflicting statements.

Conflict Resolution Prompt

Role: You are a requirements engineer. You are given two potentially conflicting requirements.

Requirement 1: <requirement_1>
Requirement 2: <requirement_2>

Below are other relevant requirements from the same system:

<List of related requirements from the same system.>

Task: Resolve the conflict in Arabic using a hybrid strategy:

1. If possible, **combine both requirements into a single non-conflicting and concise requirement** that preserves the intent of both.
2. If they cannot be merged clearly, **rewrite them as two separate non-conflicting requirements**, possibly by adding conditions, default cases, or clarifying their scope.

Important Constraints:

- Avoid vague expressions such as "as needed", "if possible", or "may".
- Only merge requirements if they can be resolved clearly and deterministically.

Expected Output Format:

[Resolution Type]: Merge/Rephrase
Resolved Requirement(s):
- ...