

Entity and non-functional-requirement extraction from user stories: evidence from supervised sequence models and reproducible LLM benchmarking

Hazem Abedelrazeq Alregeb

Islamic University of Gaza

Motaz SAAD

`motazk.saad@unisalento.it`

University of Salento

Research Article

Keywords:

Posted Date: October 19th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11390767/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Entity and non-functional-requirement extraction from user stories: evidence from supervised sequence models and reproducible LLM benchmarking

Anonymous Author(s)

Abstract

User stories capture software requirements in concise natural language, but their structural elements and quality attributes are costly to identify consistently at scale. We study extraction of the actor (USER), requested capability (FUNCTION), and intended benefit (GOAL), together with five non-functional-requirement (NFR) aspects: Privacy, Performance, Usability, Scalability, and Maintainability. We report the supervised sequence-labelling results recorded for a corpus of 2,234 English user stories and a 6,980-story augmented version. Among the reported CRF, LSTM-CRF, and BiLSTM-CRF models and a custom spaCy baseline, BiLSTM-CRF achieved the highest entity F1: 0.962 on the original test split and 0.986 on the augmented test split. It also had the strongest reported per-aspect NFR F1 (0.981–0.988 on augmented data). We additionally construct a reproducible entity-only benchmark from 6,955 released stories, with a fixed 1,391-story test split shared by BiLSTM-CRF and seven locally served open-weight LLMs. Under the same normalized exact-span micro-F1 metric, BiLSTM-CRF reaches 0.953 with original training and 0.961 with 599 leakage-safe augmented variants. The best zero-shot LLM, Gemma4-12B, reaches 0.414; a Qwen3.5-4B one-shot configuration selected exclusively on validation reaches 0.434 on the held-out test set. The shared benchmark does not include validated story-level NFR labels, so it does not support an NFR comparison. The paper releases the split, configurations, model metadata, timings, and raw predictions for audit.

Keywords: User stories, Requirements engineering, Named entity recognition, Non-functional requirements, BiLSTM-CRF, Large language models

1 Introduction

User stories are widely used in agile development to express a capability from a stakeholder’s perspective [2, 6]. Their familiar “As a ... I want ... so that ...” form nominally links an actor, an action, and a benefit. This compact notation helps teams discuss value early, but it does not make the stories machine-readable by itself. In operational backlogs, stories can omit a clause, merge several capabilities, use domain-specific terminology, or deviate from the canonical template. Analysts consequently spend effort recovering the intended structure before stories can support search, quality assessment, traceability, release planning, or automated model construction. Research on user-story quality and analysis confirms both the practical value of this structure and the difficulty of deriving it reliably from natural language [16, 17, 21].

The payoff of reliable extraction is not limited to making a backlog easier to read. The actor, capability, and rationale can form a bridge from informal requirements to conceptual models and other engineering artefacts [12, 18]. They can also make it possible to identify recurring roles, compare requested capabilities, and detect stories whose value rationale is absent or ambiguous. Thus, the problem is a requirements-engineering information-extraction problem: preserve the original wording while producing a representation that can be inspected, traced, and reused. Recent work on extracting software-related information from natural-language sources further illustrates the role of automated extraction in requirements elicitation [23].

This problem has two complementary dimensions. The first is structural: identify the actor (USER), requested capability (FUNCTION), and rationale (GOAL). These elements can support story decomposition, conceptual-model extraction, and downstream engineering artefacts [9, 12, 23]. The second is qualitative: recognise non-functional requirements (NFRs), such as privacy, performance, usability, scalability, and maintainability, that influence architectural and planning decisions even when they are not stated in a standard template. NFRs are especially important because they may cross-cut multiple functional stories and affect design choices,

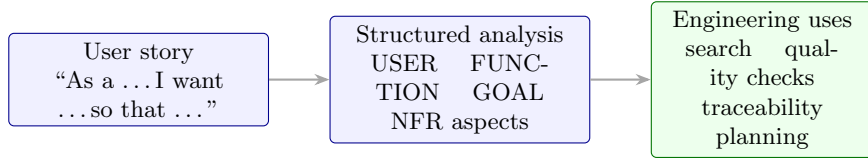


Fig. 1 Motivation for user-story analysis: structural entities and quality aspects turn free-text stories into information that can support downstream requirements-engineering activities.

acceptance criteria, and risk assessment. Automated NFR analysis has a long history in requirements engineering [4, 5, 14, 24]; privacy analysis of user stories has also been studied directly [3]. Treating structural and quality information together makes the extracted representation more useful than detecting either in isolation.

Named entity recognition (NER) supplies a natural formulation for the structural task: assign a type to the spans that express the actor, requested action, and benefit. Sequence models with a conditional random field (CRF) output layer remain a well-established approach for structured prediction [11, 15], while NER has been adapted to software-engineering artefacts such as Stack Overflow posts and requirements documents [19, 26, 27]. User-story-specific work has explored NER for model-based design and neural architectures that combine contextual representations with convolutional layers [1, 25]. These approaches motivate a supervised sequence-labelling baseline, but they also make the details of corpus construction, split assignment, and span scoring central to interpreting reported performance.

At the same time, instruction-tuned LLMs offer a training-free alternative for requirements tasks, including elicitation, classification, specification support, and user-story quality assistance [8, 13, 20, 22, 28, 29]. An LLM can return a plausible structured answer without task-specific training, but its free-text output can be malformed, incomplete, or more expansive than the requested span. Apparent model advantages can therefore arise from a changed prompt, a changed test subset, or a permissive post-processing rule rather than from the extraction capability itself. Fair comparison with token-oriented models requires a common held-out set, an explicit normalization and exact-span-matching rule, and a configuration-selection protocol that does not tune on the final test set.

The contribution of this manuscript is deliberately evidence-oriented. First, it consolidates the reported supervised results from the underlying thesis experiments without recreating unsupported precision/recall values. Second, it supplies a reproducible shared entity benchmark that evaluates BiLSTM-CRF and prompted LLMs on identical held-out stories with the same scorer. Third, it tests a leakage-safe augmentation condition and uses validation, rather than the test set, to select an LLM prompting configuration. Finally, it states the boundaries of historical and reproducible evidence rather than treating different corpora, label constructions, or metrics as interchangeable. This separation is important: the paper reports valuable historical NFR evidence, while making only the entity-extraction claims that can be independently audited on the released shared split.

2 Task and experimental evidence

2.1 Tasks

For a token sequence $x = (x_1, \dots, x_n)$, entity extraction assigns an IOB-style label sequence $y = (y_1, \dots, y_n)$ for USER, FUNCTION, GOAL, or outside. The NFR task concerns five aspects: Privacy, Performance, Usability, Scalability, and Maintainability. The historical supervised experiments treat these as annotations in the original experimental pipeline. The new shared benchmark evaluates only the three structural entities because its released NFR field is not a validated five-category story-level label.

2.2 Corpus and augmentation

The supervised corpus contains 2,234 English user stories collected from a requirements dataset, Scrum Alliance material, the CKAN portal, and other public sources. The source counts are shown in Table 1. The reported 80/20 split yields 1,787/447 stories for the original corpus. A reported augmented corpus contains 6,980 stories (5,584/1,396 train/test). The repository does not include the original supervised training code or per-run evaluation logs; consequently, this paper reports the recorded outcomes and does not claim a new rerun.

Table 2 illustrates both the span-level annotation format and a retained augmented training record. The examples are representative records rather than a source-variant pair. In the augmented record, the three inherited spans are retained verbatim, which is the criterion used by the reconstruction pipeline before a variant can enter the augmented training split.

The historical augmentation is described in the existing experimental record as synonym-based expansion using corpus-extender. Its exact seeds and generated training files are unavailable, so the historical augmented

Table 1 Composition of the supervised corpus.

Source	Stories
Requirements data sets [7]	1,680
Scrum Alliance	222
CKAN	63
Other public sources	269
Original corpus	2,234
Reported augmented corpus	6,980

Table 2 Examples of an original annotated user story and a leakage-safe augmented training record. Gold entities are shown as USER; FUNCTION; GOAL.

Record	User story	Gold entities
Original (story 5)	As a UI designer, I want to move on to round 2 of Homepage edits, so that I can get approvals from leadership.	UI designer; move on to round; get approvals from leadership.
Augmented (augmented:6575.0)	As a client, I want to reject a friend request, so we do not have mutual access to each other’s sunflower count.	client; reject a friend request; we do not have.

Table 3 Reported entity-extraction F1.

Corpus	Model	F1
Original	spaCy	.859
Original	CRF	.934
Original	LSTM-CRF	.949
Original	BiLSTM-CRF	.962
Augmented	spaCy	.972
Augmented	CRF	.980
Augmented	LSTM-CRF	.974
Augmented	BiLSTM-CRF	.986

result should be interpreted as a reported experimental condition, not as a fully reproducible independent-data evaluation. In contrast, the new leakage-safe augmentation condition is reconstructed from the committed augmented text and is specified in Section 4.

2.3 Supervised models and metrics

The recorded comparison includes a custom spaCy pipeline, CRF, LSTM-CRF, and BiLSTM-CRF. The neural model uses 300-dimensional embeddings, a two-layer LSTM with 256 hidden units, dropout 0.5, learning rate 0.001, batch size 32, and 50 epochs. The CRF layer models dependencies across adjacent output labels, while bidirectionality supplies left and right context [10, 11, 15]. Precision, recall, and F1 are standard sequence-labelling measures; $F_1 = 2PR/(P + R)$.

3 Supervised results

Table 3 presents the entity-extraction F1 values recorded in the thesis material. We report F1 consistently for both conditions. The archived augmented precision/recall cells are inconsistent with their printed F1 values, and raw logs needed to resolve the discrepancy are absent; reporting F1 only avoids presenting incompatible statistics.

BiLSTM-CRF is the best recorded model on both conditions. The gain from .962 to .986 is associated with training and testing on the augmented corpus; it is not evidence of generalization to an untouched external test set. The original record also reports the per-aspect F1 values in Table 4. These values establish the ordering within that experiment; the new shared benchmark does not make an NFR comparison.

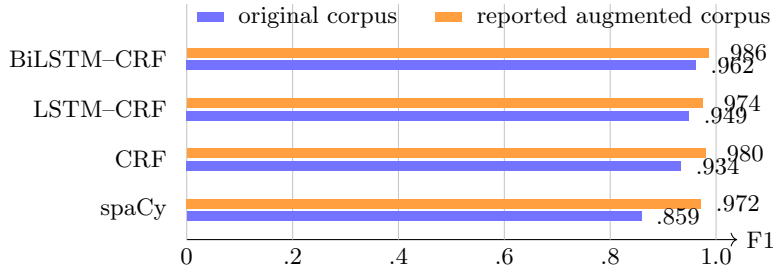


Fig. 2 Reported entity-extraction F1 by supervised model and corpus condition. These are historical results on different reported splits; the figure visualises their within-study ordering and is not a shared-test comparison.

Table 4 Reported NFR-aspect F1 for BiLSTM-CRF.

Aspect	Original	Augmented
Privacy	.962	.981
Performance	.965	.988
Usability	.968	.982
Scalability	.962	.986
Maintainability	.964	.981

4 Shared reproducible entity benchmark

The public replication package is available at an anonymized repository link supplied for review. It contains the fixed split, training and inference scripts, command-line configurations, model metadata, prompts, timing records, raw predictions, and scorer. The entity-extraction benchmark derives from 6,955 unique released story identifiers. The identifiers were shuffled once with seed 42 and partitioned at story level into 5,008 training, 556 validation, and 1,391 held-out test stories. Eleven overlapping source spans, which cannot be represented by a single-label sequence tagger, were removed using a pre-declared priority order (USER, FUNCTION, GOAL). Thus, both the BiLSTM-CRF and LLMs are evaluated against the identical sanitised test annotations.

4.1 Reproducibility protocol

The BiLSTM-CRF reproduction uses the training split for vocabulary and parameter estimation and validation F1 for checkpoint selection. Its augmentation condition adds only 599 textually changed variants whose source stories belong to the original training partition; validation and test stories remain unchanged. The committed raw augmented file contains 6,981 rows. Structural labels are inherited from the mapped source story only when every non-empty USER, FUNCTION, and GOAL label is a case-insensitive exact substring of the augmented text; 6,808 rows meet that alignment test, 147 do not, and 26 cannot be mapped. The final leakage-safe filter excludes 1,902 variants sourced from validation or test stories and 4,307 unchanged copies, admitting the remaining 599 variants. No NFR labels are propagated to this entity-only training file. The augmentation provenance does not preserve a per-row transformation operation or generation seed; therefore, we report the deterministic filtering and labels, not an unsupported claim about the linguistic transformation of each variant. Table 5 records the executable configuration and measured timing. Both neural runs used seed 42, deterministic cuDNN, disabled cuDNN benchmarking, PyTorch 2.14.1+cu130, CUDA 13.0, and one NVIDIA RTX A6000. The model uses a 3,664-word vocabulary for original training and a 3,669-word vocabulary with variants.

4.2 LLM protocol, inventory, and prompts

LLM configurations are served locally through Ollama’s chat API with native JSON output, `think:false`, a 4,096-token context limit, a maximum of 1,024 generated tokens, temperature 0, and seed 42. The recorded `top_p` and `top_k` arguments are null, so the Ollama model defaults apply. Seven models are first evaluated zero-shot. A separate validation-only sweep then evaluates 0, 1, 2, and 4 demonstrations selected deterministically from the training partition for every model; it selects one configuration before exactly one final test run. Table 6 gives the exact local artifacts used. Artifact size is the Ollama model artifact; “mean wall” is end-to-end request-wall time on the same RTX A6000 workstation, so it includes serving overhead and is not a hardware-independent throughput claim. Full 64-hex-character digests and per-story timing distributions are retained in the repository metadata.

Table 5 Reproducible BiLSTM-CRF configuration and single-workstation timing. Fit wall time includes gradient updates and validation.

Setting	Original training	+599 variants
Training stories	5,008	5,607
Validation / test stories	556 / 1,391	556 / 1,391
Seed; epochs; patience	42; 30; 5	42; 30; 5
Batch; embedding; hidden units	32; 20; 50	32; 20; 50
Learning rate	.001	.001
Gradient updates (s)	195.450	226.150
Fit wall time (s)	203.874	234.622
Test inference (s)	.685	.815
Test inference (ms/story)	.493	.586

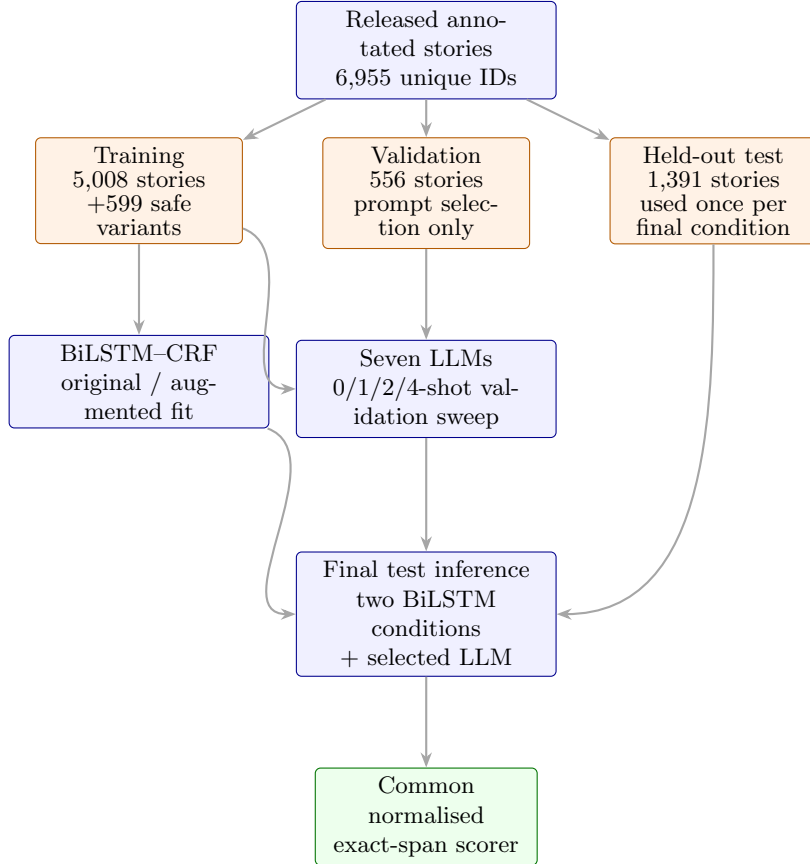


Fig. 3 Leakage-safe shared evaluation design. Training data fit BiLSTM-CRF and supply demonstrations; validation data select an LLM configuration; test data feed only final inference and are never an input to model fitting or the prompt sweep.

Zero-shot system prompt

You are an expert in requirements engineering and named entity recognition. Given a user story written in natural language, extract the three entity types below.
Return ONLY a valid JSON object with exactly one key: { "entities": [{ "type": "<USER—FUNCTION—GOAL>", "text": "<exact entity text as it appears in the story>" }] }.
Rules: Extract every occurrence of each entity type. Copy entity text verbatim; do not paraphrase. USER is the actor; FUNCTION is the action/feature the user wants; GOAL is the reason/benefit ("so that ..."). If there is no entity, use an empty list. No explanations, markdown, or extra keys.

Table 7 reports the complete validation-only prompt sweep. The selected one-shot condition prepended the single deterministic training example **story 484**, containing its story text and gold entity JSON. The complete one-shot prompt is preserved verbatim in its run metadata. Its distinct prompt digest is **eb73d19a2993**, whereas the zero-shot prompt digest is **1140a1341236**. All scores below are micro entity-level F1 under the same case- and punctuation-normalised exact-span matching rule. This is stricter than semantic or containment matching: a plausible but longer FUNCTION or GOAL phrase is counted as incorrect if it does not match the annotated span exactly. All LLM runs produced a structured prediction record for each evaluated story.

The leakage-safe augmentation improves BiLSTM-CRF from 0.953 to 0.961 overall F1, with the largest gain for GOAL (0.923 to 0.945). The original-training BiLSTM-CRF substantially outperforms every zero-shot LLM (0.953 versus 0.414 for the strongest zero-shot LLM), and the augmented model remains ahead of the validation-selected one-shot LLM (0.961 versus 0.434). Qwen3.5-4B with one demonstration won the validation

Table 6 LLM inventory and zero-shot inference configuration. All models used seed 42, temperature 0, 4,096 context tokens, at most 1,024 generated tokens, native JSON chat output, and `think:false`. Digest is a reproducibility prefix.

Ollama tag	Family	Parameters	Quant.	Context	Artifact (GiB)	Digest	Mean wall (s/story)
<code>gemma4:12b</code>	Gemma 4	11.9B	Q4_K_M	262,144	7.47	6114515d63c1	.808
<code>ministral-3:8b</code>	Mistral 3	8.9B	Q4_K_M	262,144	5.61	1922acc5827	.695
<code>qwen3.5:4b</code>	Qwen 3.5	4.2B	Q4_K_M	262,144	3.10	d8b0f5e9760	.759
<code>qwen3.5:2b</code>	Qwen 3.5	1.9B	Q8_0	262,144	2.50	0689d44085e0	.587
<code>ministral-3:14b</code>	Mistral 3	13.9B	Q4_K_M	262,144	8.46	4760c35aeb9d	1.071
<code>nemotron-3-nano:4b</code>	Nemotron H	4.0B	Q4_K_M	262,144	2.64	6cc467f05439	.459
<code>gemma4:e4b</code>	Gemma 4	7.5B	Q4_K_M	131,072	6.13	dc35e8d9c606	.363

Table 7 Validation-only prompt ablation on 556 stories. Entries are overall exact-span F1; the selected configuration is the single highest value, chosen before held-out test inference.

Model	0-shot	1-shot	2-shot	4-shot	Best
Gemma4-12B	.407	.419	.398	.400	1
Gemma4-E4B	.342	.412	.396	.394	1
Ministral-3-8B	.399	.410	.407	.402	1
Ministral-3-14B	.376	.402	.390	.382	1
Qwen3.5-2B	.373	.409	.411	.418	4
Qwen3.5-4B	.400	.424	.415	.409	1
Nemotron-3-Nano-4B	.374	.406	.412	.416	4

Table 8 Shared held-out entity-extraction benchmark (1,391 stories). All entries are exact-span F1 under the same scorer. “Selected 1-shot” was chosen using validation data only.

Model	Condition	USER	FUNCTION	GOAL	Overall
BiLSTM-CRF	Original train	.991	.934	.923	.953
BiLSTM-CRF	+599 variants	.991	.941	.945	.961
Gemma4-12B	Zero-shot	.984	.069	.095	.414
Ministral-3-8B	Zero-shot	.988	.068	.079	.405
Qwen3.5-4B	Zero-shot	.930	.067	.089	.399
Qwen3.5-2B	Zero-shot	.943	.073	.059	.384
Ministral-3-14B	Zero-shot	.944	.063	.068	.383
Nemotron-3-Nano-4B	Zero-shot	.956	.081	.011	.375
Gemma4-E4B	Zero-shot	.896	.079	.020	.358
Qwen3.5-4B	Selected 1-shot	.991	.072	.113	.434

sweep (0.424 overall F1) and then reached 0.434 on the untouched test set. The high USER scores and low FUNCTION/GOAL scores indicate that the latter labels are the principal challenge under strict annotation-boundary matching. On the reported RTX A6000 workstation, original-training BiLSTM-CRF used 195.450 seconds for gradient updates and 0.493 ms per held-out story; the augmented condition used 226.150 seconds and 0.586 ms per story. The selected LLM’s mean request-wall time was 0.723 seconds per story. These are single-workstation measurements, not hardware-independent latency claims.

This direct entity comparison does not extend to NFRs. The shared split intentionally leaves its `nfr` lists empty because the available source field is term-level rather than a validated mapping to the five story-level categories. Consequently, the historical NFR results in Table 4 remain reported evidence only, and no LLM NFR ranking is claimed.

5 Discussion and limitations

The historical record supports the conclusion that BiLSTM-CRF was the strongest of the evaluated supervised configurations. Separately, the new shared benchmark supports a narrower, reproducible statement: its BiLSTM-CRF reproduction exceeds the tested zero-shot and validation-selected one-shot LLMs under the same exact-span scorer on the same held-out stories. The augmentation result is credible within this split because variants of validation and test stories were excluded from training. It does not establish statistical significance, hardware-independent latency, deployment cost ratios, or superiority over all possible LLM prompting strategies.

5.1 Implications for requirements practitioners

For a team that maintains a sizeable, stable backlog and can label representative stories, the reproduced BiLSTM-CRF is the practical default among the evaluated approaches. It delivers boundary-accurate structural extraction with sub-millisecond per-story inference on the measured workstation, and the leakage-safe

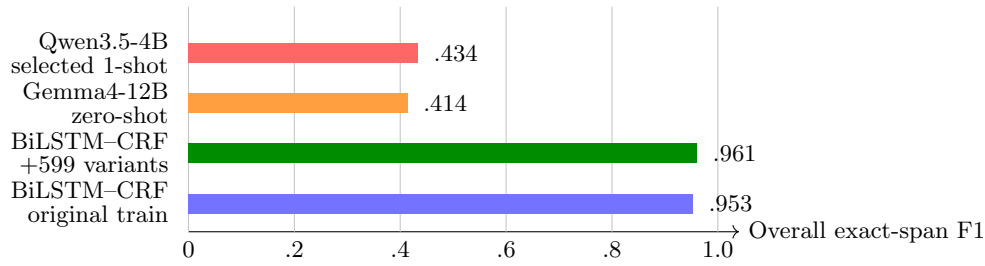


Fig. 4 Overall entity exact-span F1 on the common 1,391-story held-out test set. The LLM configurations shown are the strongest zero-shot model and the validation-selected one-shot model; the remaining zero-shot LLM results appear in Table 8.

variants provide a small but consistent improvement without exposing validation or test stories during training. The resulting USER, FUNCTION, and GOAL fields can support role and capability search, detection of stories that omit a rationale, traceability links, and release-planning views. These uses preserve the original story text while making its structure inspectable by analysts.

Prompted LLMs are better treated as interactive analyst assistance than as an unattended replacement for a trained sequence tagger in this setting. Their high USER scores can speed up role identification and their JSON output can make exploratory review convenient, but the low FUNCTION and GOAL exact-span scores mean that an analyst should verify suggested action and rationale spans before they drive downstream artefacts or quality decisions. A reflective practitioner can therefore use an LLM to propose annotations for review or to bootstrap a labelling exercise, then train and periodically re-evaluate a project-specific model once sufficient reviewed data are available. This recommendation is conditional on the common benchmark and scorer used here; teams should repeat the split, annotation, and validation protocol on their own domain before deployment.

Four limitations are central. First, the original supervised code, split files, random seeds, and raw logs are absent, preventing a clean reconstruction of the historical augmented condition and resolution of its P/R discrepancy. Second, synthetic augmentation can introduce lexical overlap between training and test data; external or project-disjoint evaluation is needed before claiming broad generalisation. Third, exact span matching penalises LLM outputs that identify the correct semantic action or rationale but use different boundaries; it is nevertheless appropriate as the primary metric here because both systems are scored identically. Fourth, the shared benchmark lacks validated story-level NFR categories, so it provides no basis for a direct NFR comparison.

Future work should regenerate the historical augmented experiment with immutable splits, add an untouched project-level holdout, compare with fine-tuned transformer encoders, and perform a pre-registered prompt search on the validation partition. Extending the shared benchmark with validated NFR labels would make NFR accuracy, latency, and cost trade-offs testable rather than inferred.

6 Conclusion

This paper reports entity and NFR extraction from user stories while distinguishing historical thesis evidence from a new reproducible comparison. In the historical record, BiLSTM-CRF attains entity F1 of .962 on the original corpus and .986 on the augmented corpus, with reported augmented NFR-aspect F1 between .981 and .988. In the new shared 1,391-story entity benchmark, BiLSTM-CRF reaches .953 with original training and .961 with 599 leakage-safe augmented variants. The strongest zero-shot LLM, Gemma4-12B, reaches .414; the Qwen3.5-4B one-shot configuration selected solely on validation reaches .434 on the final test. The benchmark is deliberately entity-only: a valid NFR comparison awaits validated story-level labels. Releasing the missing historical artifacts remains a necessary next step.

Declarations

Funding: Not applicable.

Competing interests: The authors declare no competing interests.

References

- [1] Aszani, Mulyana S (2023) Model-based design of user story using named entity recognition (ner). In: Intelligent Computing and Optimization (ICO 2023), Lecture Notes in Networks and Systems, vol 853. Springer, pp 135–144, https://doi.org/10.1007/978-3-031-50327-6_15
- [2] Beck K (1999) Extreme Programming Explained: Embrace Change. Addison-Wesley

- [3] Casillo F, Deufemia V, Gravino C (2022) Detecting privacy requirements from user stories with nlp transfer learning models. *Information and Software Technology* 146:106853. <https://doi.org/10.1016/j.infsof.2022.106853>
- [4] Chung L, Nixon BA, Yu E, et al (2000) *Non-Functional Requirements in Software Engineering*. Springer
- [5] Cleland-Huang J, Settini R, Zou X, et al (2007) Automated classification of non-functional requirements. *Requirements Engineering* 12(2):103–120. <https://doi.org/10.1007/s00766-007-0045-1>
- [6] Cohn M (2004) *User Stories Applied: For Agile Software Development*. Addison-Wesley
- [7] Dalpiaz F (2018) Requirements data sets (user stories). Mendeley Data, V1, <https://doi.org/10.17632/7zbk8zsd8y.1>
- [8] El-Hajjami A, Fafin N, Salinesi C (2024) Which AI technique is better to classify requirements? an experiment with SVM, LSTM, and ChatGPT. In: *Joint Proceedings of the REFSQ-2024 Workshops, CEUR Workshop Proceedings*, vol 3672. CEUR-WS.org
- [9] Herwanto GB, Quirchmayr G, Tjoa AM (2022) From user stories to data flow diagrams for privacy awareness: A research preview. In: *Requirements Engineering: Foundation for Software Quality (REFSQ), Lecture Notes in Computer Science*, vol 13216. Springer, pp 148–155
- [10] Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Computation* 9(8):1735–1780
- [11] Huang Z, Xu W, Yu K (2015) Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:150801991*
- [12] Kochbati T, Li S, Gérard S, et al (2021) From user stories to models: A machine learning empowered automation. In: *Proceedings of the 9th International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, pp 28–40
- [13] Krishna M, Gaur B, Verma A, et al (2024) Using LLMs in software requirements specifications: An empirical evaluation. In: *Proceedings of the 32nd IEEE International Requirements Engineering Conference (RE)*. IEEE, pp 475–483, <https://doi.org/10.1109/RE59067.2024.00060>
- [14] Kurtanović Z, Maalej W (2017) Automatically classifying functional and non-functional requirements using supervised machine learning. In: *Proceedings of the 25th IEEE International Requirements Engineering Conference (RE)*, pp 490–495, <https://doi.org/10.1109/RE.2017.36>
- [15] Lafferty JD, McCallum A, Pereira FCN (2001) Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: *Proceedings of the 18th International Conference on Machine Learning (ICML)*, pp 282–289
- [16] Lucassen G, Dalpiaz F, van der Werf JMEM, et al (2015) Forging high-quality user stories: Towards a discipline for agile requirements. In: *Proceedings of the 23rd IEEE International Requirements Engineering Conference (RE)*, pp 126–135, <https://doi.org/10.1109/RE.2015.7320415>
- [17] Lucassen G, Dalpiaz F, van der Werf JMEM, et al (2016) Improving agile requirements: The quality user story framework and tool. *Requirements Engineering* 21(3):383–403
- [18] Lucassen G, Robeer M, Dalpiaz F, et al (2017) Extracting conceptual models from user stories with visual narrator. *Requirements Engineering* 22(3):339–358. <https://doi.org/10.1007/s00766-017-0270-1>
- [19] Malik G, Cevik M, Khedr Y, et al (2021) Named entity recognition on software requirements specification documents. In: *Proceedings of the 34th Canadian Conference on Artificial Intelligence (Canadian AI)*
- [20] Marques N, Silva RR, Bernardino J (2024) Using ChatGPT in software requirements engineering: A comprehensive review. *Future Internet* 16(6):180. <https://doi.org/10.3390/fi16060180>
- [21] Raharjana IK, Siahaan D, Fatichah C (2021) User stories and natural language processing: A systematic literature review. *IEEE Access* 9:53811–53826. <https://doi.org/10.1109/ACCESS.2021.3070606>

- [22] Ronanki K, Berger C, Horkoff J (2023) Investigating ChatGPT’s potential to assist in requirements elicitation processes. In: Proceedings of the 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE, pp 354–361, <https://doi.org/10.1109/SEAA60479.2023.00062>
- [23] Siahaan D, Raharjana IK, Fatichah C (2023) User story extraction from natural language for requirements elicitation: Identify software-related information from online news. *Information and Software Technology* 158:107195. <https://doi.org/10.1016/j.infsof.2023.107195>
- [24] Slankas J, Williams L (2013) Automated extraction of non-functional requirements in available documentation. In: Proceedings of the 1st International Workshop on Natural Language Analysis in Software Engineering (NaLRE) at ICSE
- [25] Syaifudin MF, Rakhmawati NA, Indraswari R (2024) Named entity recognition on user requirement analysis with bert-cnn. In: 2024 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT). IEEE, pp 295–299, <https://doi.org/10.1109/3ICT64318.2024.10824318>
- [26] Tabassum J, Maddela M, Xu W, et al (2020) Code and named entity recognition in StackOverflow. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL), pp 4913–4926
- [27] Ye D, Xing Z, Foo CY, et al (2016) Software-specific named entity recognition in software engineering social content. In: Proceedings of the 23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER), pp 90–101
- [28] Zhang Y, Li Y (2024) The classification of software requirements using large language models: Emerging results and future directions. In: Proceedings of the 2024 Brazilian Symposium on Software Components, Architectures, and Reuse (SBCARS). IEEE, <https://doi.org/10.1109/CBASE64041.2024.10824444>
- [29] Zhang Z, Rayhan M, Herda T, et al (2024) LLM-based agents for automating the enhancement of user story quality: An early report. In: International Conference on Agile Software Development (XP), Lecture Notes in Business Information Processing, vol 512. Springer, pp 117–126, https://doi.org/10.1007/978-3-031-61154-4_8