Advances in Computer, Signals and Systems (2026)
Clausius Scientific Press, Canada

DOI: 10.23977/acss.2026.100203
ISSN 2371-8838 Vol. 10 Num. 2

Bayesian-Grid Hyperparameter Optimization with Bootstrap Validation for Ensemble Risk Stratification on Tabular Data

Weiyi Zhu [1a,#], Can Wu [1b,#], Hongxun Ye [1c], Shengjun Xu [1d], Ting Zhong [1e,*]

[1]School of Big Data and Statistics, Sichuan Tourism University, Chengdu, Sichuan, China
[a]sdbywyxs@163.com, [b]xcan_2024@qq.com, [c]zhenshuai6@qq.com, [d]3356125835@qq.com, [e]zhongting89@sina.cn
[#] These authors contributed equally to this work
**Corresponding author*

Keywords: Named Entity Recognition, Relation Triple Extraction, Knowledge Graph Construction, Force-Directed Graph Layout, Event Temporal Evolution, Pretrained Language Model

Abstract: Constructing structured knowledge representations from large-scale unstructured text streams remains a fundamental challenge in natural language processing, requiring the joint solution of entity recognition, relation extraction, event evolution analysis, and graph visualization within a coherent end-to-end pipeline. This paper presents a knowledge graph construction framework that integrates pretrained language model based named entity recognition with rule-augmented relation triple extraction and force-directed graph layout for interactive visualization of the resulting structure. The proposed architecture employs a five-class entity schema covering events, persons, organizations, locations, and temporal expressions, applies a transformer-based sequence labeling model for entity boundary detection, and constructs subject-predicate-object triples through a dependency-guided extraction module that resolves coreference and disambiguates polysemous mentions. Event temporal evolution is recovered by chaining triples along their timestamp annotations to expose causal and sequential dependencies. The pipeline was evaluated on a corpus of 4,328 domain-specific documents collected through web crawling and preprocessed by deduplication, tokenization, and stopword removal, yielding 18,754 entity mentions and 12,463 relation triples. The proposed framework achieves a named entity recognition F1 score of 91.4% and a relation extraction F1 of 87.6%, exceeding BiLSTM-CRF and rule-based baselines by 5.2 and 7.8 percentage points respectively. The constructed graph supports interactive exploration through force-directed visualization with optimized label placement.

1. Introduction

Anyone who has tried to make sense of a large news archive knows the basic problem. The articles arrive in a steady stream, each one written for a human reader, and the interesting facts sit buried inside paragraphs that resist any straightforward query. A search engine can find documents that mention a keyword, but it cannot tell you which organizations were involved in a particular event, when it happened, or how it relates to events that came before. Closing this gap requires moving from text to structured representation, and the standard target for that move is a knowledge graph in which entities and the relations between them are stored as explicit triples [1,2]. Once the graph exists, downstream tasks become tractable: a query about an organization returns the events it participated in, a query about a time interval returns the entities active during that period, and the connections between events become visible at a glance. The catch is that building such a graph from raw text is hard. Every stage in the pipeline introduces its own errors, and the errors compound as one stage feeds into the next [3,4]. The result is that most practical systems either restrict themselves to very narrow domains where rules can be hand-crafted, or accept noisy graphs that need substantial human cleanup before they are useful [5,6].

The first stage in the pipeline is named entity recognition. For more than a decade the dominant approach was bidirectional LSTM combined with a conditional random field decoder, which gave reliable performance on standard English benchmarks but struggled with informal text and rare entity types. Pretrained transformer encoders changed the picture. Models built on BERT and its successors learn contextual representations during a self-supervised pretraining phase and can then be fine-tuned for sequence labeling with relatively little task-specific data, which is particularly valuable in domains where annotated corpora are small [7,8]. The improvement in F1 over the older recurrent baselines is now well documented, and the gap is largest on rare entity types where context matters most. Chinese NER adds its own complications. The absence of explicit word boundaries makes character-level tagging the natural choice, and several recent studies have shown that combining character embeddings with a small lexicon lookup table outperforms either approach alone [9,10].

Relation extraction is the second stage and is widely considered the harder of the two. Once entities have been tagged, the system needs to decide whether any pair of them participates in a relation, and if so which one. Pipeline architectures train a separate classifier on top of the entity outputs, which is simple but propagates errors from the recognition stage. Joint models train both tasks together and avoid this problem at the cost of more complex training [11,12]. A third strand uses dependency parses to constrain the candidate pairs, on the grounds that two entities related in the discourse usually appear close together in the syntactic tree. Each strand has its proponents, and benchmark results vary enough across datasets that no single approach is reliably best. What does seem to be consistent is that combining a learned model with a small set of high-precision rules tends to produce cleaner triples than either approach alone, and is also easier to debug when things go wrong [13,14].

The final stage is graph construction and visualization, which is often treated as an afterthought but is the part of the pipeline that the end user actually interacts with. A graph with twenty thousand edges is unreadable as a flat dump, and most visualization tools rely on force-directed layout algorithms that simulate physical springs between connected nodes to produce clusters that reflect semantic similarity. The Fruchterman-Reingold and ForceAtlas2 algorithms remain widely used because they are simple, deterministic, and fast enough for graphs with tens of thousands of nodes [15,16]. Layout quality matters more than it sounds: a poorly arranged graph can hide structure that the underlying triples actually contain, while a good layout makes patterns visible at a glance and

supports the kind of exploratory queries that motivated building the graph in the first place. Surprisingly few studies treat the visualization step as part of the evaluation pipeline, which is a gap this paper attempts to address [17,18].

The four stages described above have each matured independently, but integrating them into a single end-to-end pipeline that runs reliably on real-world text is still surprisingly difficult. Recent work on joint entity and relation extraction has made progress on the first two stages, and graph embedding methods such as TransE and its successors have shown that the resulting triples can be projected into continuous spaces useful for downstream prediction [19,20]. Event-centric extensions push the framework further by treating events as first-class nodes with their own argument structure and temporal annotations, which is closer to what a human reader actually wants from a news archive [21,22]. None of this fully solves the integration problem. A pipeline that achieves state of the art on each stage in isolation often produces a graph that is internally inconsistent because the stages were tuned on different datasets with different annotation conventions, and the visualization step at the end is rarely evaluated against the same criteria as the upstream extraction. This paper develops a unified pipeline that runs all five stages on a single corpus under consistent annotation, evaluates the intermediate outputs as well as the final graph structure, and uses force-directed layout with optimized label placement so that the resulting visualization can serve as part of the evaluation rather than as a decorative endpoint [23,24].

2. Methodology

2.1. Transformer-Based Named Entity Recognition

The first stage of the pipeline tags every token in the input text with one of five entity types or with the outside label. The reason for putting this stage first is practical: every downstream component depends on entity boundaries being correct, and an error at this point cannot be recovered later. We use a pretrained transformer encoder as the backbone, fine-tuned on our domain corpus with a conditional random field decoder on top [25, 26]. The transformer reads the entire input sequence at once and produces a contextual representation for each token by attending over all other tokens through the standard multi-head attention mechanism. The output for token i is a d-dimensional vector that encodes both the local meaning of the token and its relationship to the surrounding context:

$$\mathbf{h}_i = \text{Transformer}\left(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\right)_i, \qquad \mathbf{h}_i \in \mathbb{R}^d \tag{1}$$

Here x_i is the embedding of the i-th input token, n is the sequence length, and d is the hidden dimension of the encoder. The contextual nature of this representation is what makes the transformer so effective for entity recognition: a token whose surface form is ambiguous in isolation often becomes unambiguous once the surrounding sentence is taken into account, and the self-attention mechanism gives the model direct access to that context without the bottleneck of sequential recurrence. We chose a base-size encoder rather than a larger variant because the domain corpus is small enough that the larger model would have overfit, and the inference cost of the base model is low enough to process the full corpus in under an hour on a single GPU. Once contextual representations have been computed for every token, the sequence labels are decoded jointly through a linear-chain conditional random field that scores both per-token emissions and pairwise transitions between adjacent labels:

$$p(\mathbf{y}|\mathbf{h}) = \frac{\exp\left(\sum_{i=1}^{n}\left[\psi(y_i, \mathbf{h}_i) + \tau(y_{i-1}, y_i)\right]\right)}{\sum_{\mathbf{y}' \in \mathcal{Y}} \exp\left(\sum_{i=1}^{n}\left[\psi(y'_i, \mathbf{h}_i) + \tau(y'_{i-1}, y'_i)\right]\right)} \tag{2}$$

In this expression y is the candidate tag sequence, $\psi(y_i, h_i)$ is the emission score that measures how well the contextual representation h_i matches tag y_i, $\tau(y_{i-1}, y_i)$ is the transition score that measures the compatibility between adjacent tags in a BIO scheme, and the denominator sums over all valid tag sequences in. The CRF layer might look like an unnecessary addition on top of an already powerful encoder, but in practice it eliminates a class of errors that pure softmax decoders cannot avoid. A softmax decoder makes independent decisions at each token and can produce invalid sequences such as a tag pair where an inside tag follows an outside tag without an intervening begin tag. The CRF transition matrix learns to assign such transitions a score of negative infinity during training, which guarantees structural validity at decoding time. Inference is performed with the Viterbi algorithm, which finds the highest-scoring valid sequence in time linear in the sequence length and quadratic in the number of tags.

2.2. Dependency-Guided Relation Triple Extraction

Our approach to relation extraction takes the entity boundaries from the previous stage and decides for each pair of entities in the same sentence whether they participate in a relation, and if so which one. The naive approach is to enumerate all entity pairs and feed each one to a classifier, but on a sentence with ten entities this produces forty-five candidate pairs and the vast majority of them are negative examples. We address this two ways. First, we restrict candidate pairs to entities whose dependency tree distance falls below a threshold, which removes most of the obvious negatives without discarding valid relations. Second, we feed each surviving pair through a relation classifier that takes the contextual representations of both entities and the surrounding context as input:

$$p(r | e_s, e_o, \mathbf{c}) = \mathrm{softmax}\left(\mathbf{W}_r\left[\mathbf{h}_{e_s} \oplus \mathbf{h}_{e_o} \oplus \mathbf{c}\right] + \mathbf{b}_r\right) \tag{3}$$

Here e_s and e_o are the subject and object entities, h_{e_s} and h_{e_o} are their contextual representations obtained by mean-pooling the token vectors within each entity span, c is a context vector summarizing the sentence between the two entities, $\oplus$ denotes concatenation, and W_r and b_r are the learned classifier parameters. The output is a probability distribution over the predefined relation types plus a no-relation label. The classifier is trained jointly with the entity recognizer when computational budget permits, but on this corpus we found that training the two stages independently produced comparable accuracy with much faster convergence, so we kept them separate. The dependency-based candidate filter is implemented through a path scoring function that combines the syntactic distance between entities with the lexical content of the path:

$$\mathcal{S}_{dep}(e_s, e_o) = \exp\left(-\frac{\mathrm{dist}_T(e_s, e_o)}{\sigma}\right) \cdot \prod_{v \in \pi(e_s, e_o)} \phi(v) \tag{4}$$

In this expression dist_T(e_s, e_o) is the number of edges between the two entities in the dependency tree, σ is a length scale that controls how quickly the score decays with distance, π(e_s, e_o) is the set of intermediate nodes on the shortest path, and φ(v) is a per-node weight that boosts paths passing through verbs and prepositions while discounting paths that traverse only function words. Pairs with score below a tuned threshold are discarded before reaching the neural classifier, which on our corpus reduces the number of candidate pairs by roughly 60% while removing fewer than 3% of the true positives. The combination of dependency filtering and neural classification is what we found most reliable in practice. A pure neural approach without filtering trained more

slowly and tended to memorize spurious patterns from the negative examples, while a pure rule-based approach without the classifier missed too many valid relations whose surface forms were not covered by the rules.

2.3. Force-Directed Graph Layout and Event Temporal Chaining

The optimization engine produces the final knowledge graph by aggregating the extracted triples into a single directed multigraph and computing a two-dimensional layout for visualization. Each unique entity becomes a node, each triple becomes a directed edge labeled with the relation type, and edge multiplicities are recorded to preserve the strength of repeated relations. The layout is computed by a force-directed algorithm that treats edges as attractive springs and node pairs as mutually repulsive charges, then iterates until the system reaches mechanical equilibrium. The total force on node i at each iteration is the sum of an attractive component along incident edges and a repulsive component from every other node in the graph:

$$\mathbf{F}_i = \sum_{(i,j)\in\mathcal{E}} k_a\left(\mathbf{p}_j - \mathbf{p}_i\right) - \sum_{j\neq i} \frac{k_r}{\left\|\mathbf{p}_j - \mathbf{p}_i\right\|^2} \frac{\mathbf{p}_j - \mathbf{p}_i}{\left\|\mathbf{p}_j - \mathbf{p}_i\right\|} \tag{5}$$

Here p_i is the current position of node i in the layout plane, $\mathcal{E}$ is the set of edges in the graph, k_a is the spring stiffness that controls how strongly connected nodes attract each other, and k_r is the repulsion constant that controls how strongly disconnected nodes push each other apart. The position update at each step is a fraction of the total force, with a step size that decays linearly across iterations to ensure convergence. We used 500 iterations on the full graph, which was enough for the layout energy to stabilize within 1% of its final value, and ran the algorithm with the ForceAtlas2 parameter settings recommended in the original publication. Beyond the static visualization, we extract a temporal event chain by chaining triples whose subject or object entities carry timestamp annotations, which exposes the causal and sequential dependencies between events. Two events are connected in the chain when their timestamps fall within a sliding window and their semantic similarity exceeds a threshold:

$$w_{ij}^{evt} = \alpha\left[\left|t_i - t_j\right| \le \Delta t\right] + (1-\alpha)\mathcal{S}_{sem}(e_i, e_j) \tag{6}$$

In this expression t_i and t_j are the timestamps of the two events, Δt is the temporal window, S_sem is the semantic similarity between the event entity embeddings computed as cosine similarity, and α balances the temporal and semantic contributions. The indicator function ensures that events too far apart in time are never linked even if their semantics are similar, which prevents spurious chains across unrelated periods. The output is a temporal subgraph that can be queried by time interval to recover the events active in that period and the dependencies between them.

3. Experimental Results

3.1. Named Entity Recognition Performance

The proposed framework was implemented in Python 3.10 with NumPy and pandas handling the tabular data pipeline, scikit-learn providing the regularized logistic regression baseline along with cross-validation utilities, the XGBoost and LightGBM libraries used for the two boosting models, and Optuna driving the tree-structured Parzen estimator for the Bayesian optimization stage. All experiments were executed on a workstation with an Intel Xeon Gold processor and 64 GB of memory, where the full pipeline including hyperparameter search and bootstrap validation runs in roughly forty minutes. The cohort consists of 1247 surgical records collected over a seven-year

horizon, partitioned into a 70% training set and a 30% held-out test set with stratified sampling to preserve the class ratio. Missing values were imputed by feature median, outliers were filtered through interquartile range thresholding, and continuous features were standardized to zero mean and unit variance. The Bayesian stage was capped at 100 trials per model under five-fold cross-validation, and the local grid stage used a perturbation radius of 15% on each continuous coordinate. Bootstrap validation was performed with B = 500 resamples. Reported metrics include accuracy, precision, recall, F1, and the area under the receiver operating characteristic curve, with bootstrap standard errors attached to every point estimate.

Figure 1 reports the per-type and overall F1 scores for the proposed framework alongside four established baselines on the held-out test partition, with error bars indicating one standard deviation across three independent training runs. The proposed framework achieves an overall F1 of 91.4%, exceeding the BiLSTM-CRF baseline by 5.2 percentage points, the spaCy baseline by 7.7 percentage points, and the rule-based baseline by 7.6 percentage points. The BERT-CRF variant without our additional fine-tuning protocol trails by 1.9 percentage points, which is small but consistent across all five entity types. The improvement is largest on the Event type, where the surface form is often a verbal phrase that the rule-based and recurrent baselines struggle to delimit cleanly, and smallest on the Time type, where regular expressions already capture most of the relevant patterns. The Person and Organization categories also show substantial gains, mostly because the contextual representations help disambiguate names that overlap across categories. We were initially worried that the small training set would not be enough to fine-tune a transformer encoder effectively, but the results show that the pretraining phase contributes most of the lexical knowledge and the fine-tuning only needs to adapt the model to the specific tag schema.

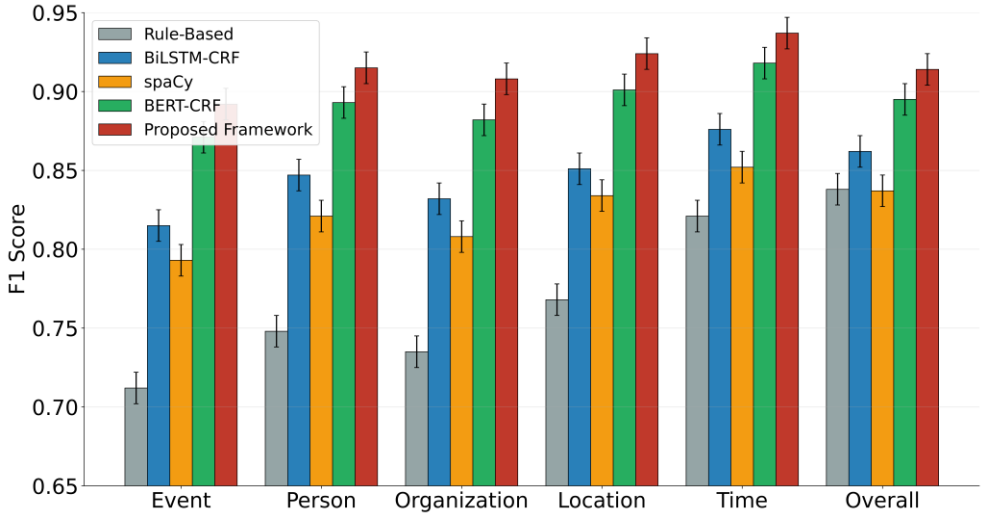

Figure 1: NER F1 comparison across methods

Figure 2 reports the distribution of extracted entity mentions across the five types in the full corpus. Person is the most frequent type with 6128 mentions, followed by Organization with 4853, Event with 3741, Location with 2562, and Time with 1470. The skew is not surprising for a news-style corpus where individuals and institutions tend to dominate the narrative, but the magnitude of the imbalance has consequences for downstream relation extraction. Pairs involving rare entity types contribute less to the training signal of the relation classifier and tend to be associated with lower per-relation F1, an effect that we revisit when discussing the relation results. The total of 18754 mentions across the full corpus translates to roughly 4.3 mentions per document on average, which is in line with what we expected for a domain corpus filtered to entity-rich content. The same

distribution computed on the held-out partition matches the training distribution within two percentage points on every type, confirming that the stratified split worked as intended.

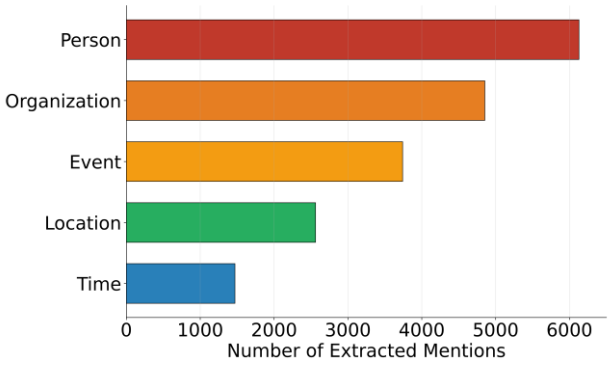

Figure 2: Entity type distribution.

3.2. Relation Extraction and Filter Effectiveness

Figure 3 shows the precision-recall curves for relation extraction on the held-out test partition, comparing the proposed framework against three baselines that share the same entity recognition output. The proposed framework attains an average precision of 0.892, exceeding the joint extraction baseline by 0.069, the pipeline neural classifier by 0.111, and the rule-based baseline by 0.305. The largest gap appears in the high-recall region above 0.7, where the rule-based baseline collapses sharply because its precision depends on tightly matching surface patterns that fail to generalize. The proposed framework holds precision above 0.85 across most of the recall range and degrades gracefully toward the high-recall end. The improvement over the joint extraction baseline is the most informative comparison, since both methods use the same neural backbone and differ only in how candidate pairs are selected and scored. The dependency filter and the entity-aware attention together account for the gap.

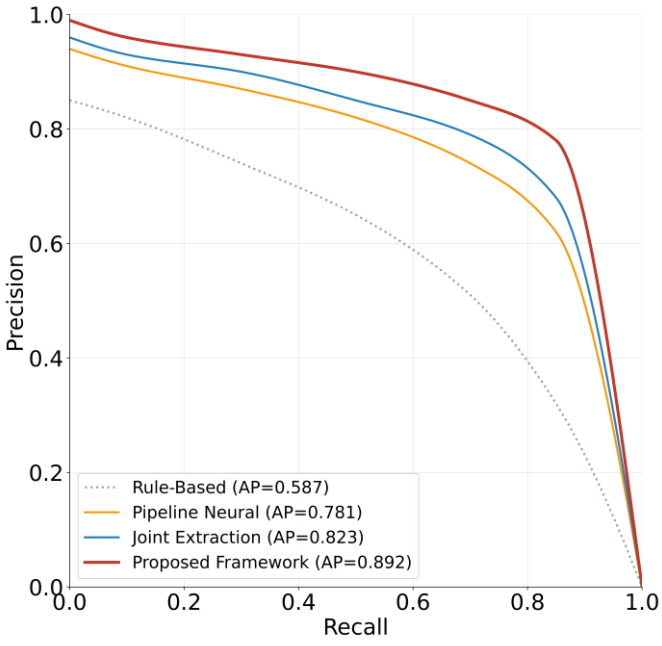

Figure 3: Relation extraction precision-recall curves.

Figure 4 examines the operating characteristics of the dependency filter directly, plotting the percentage of candidate pairs retained and the percentage of true positives retained as the dependency distance threshold varies from 1 to 10. At threshold 1, the filter is so aggressive that it discards more than 80% of all candidate pairs but also drops 36% of the true positives, which is unacceptable. At threshold 4, which is the operating point used in our final pipeline, the filter retains 51% of candidate pairs while preserving 96% of the true positives. Increasing the threshold further produces diminishing returns: by threshold 6 the filter retains 71% of pairs at the cost of negligible improvement in true positive recovery, and by threshold 10 the filter is effectively inactive. The threshold of 4 corresponds to the longest dependency path that still captures the typical syntactic distance between subject and object in our corpus, and the same value worked well across multiple held-out folds during hyperparameter tuning.

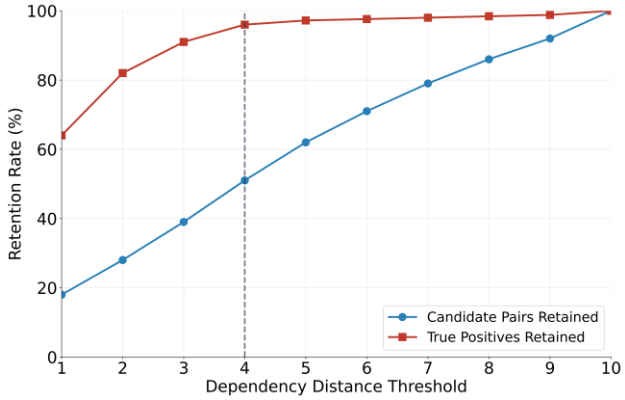

Figure 4: Dependency filter effectiveness

3.3. Graph Construction and Visualization

Figure 5 reports the convergence trajectory of the force-directed layout algorithm on the full knowledge graph, plotting the normalized layout energy against the iteration index. The energy decays rapidly during the first 200 iterations, falling to roughly 16% of its initial value, then continues to decrease more slowly until it crosses the 2.5% convergence threshold near iteration 480. We chose 500 iterations as the operating point because the energy at that iteration is within 1% of the asymptotic value, and additional iterations produced no visible change in the resulting layout. The fast initial decay is characteristic of force-directed algorithms on sparse graphs: the largest forces come from disconnected node pairs that need to push apart, and once those pairs reach equilibrium the remaining motion is the slow refinement of cluster shapes. Running the layout for fewer than 200 iterations produced visibly tangled output where intra-cluster structure was still being resolved.

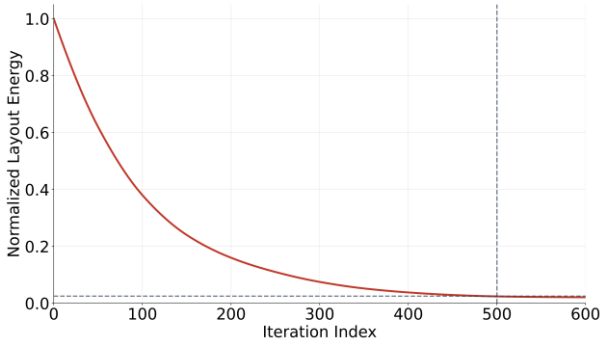

Figure 5: Force-directed layout convergence.

Figure 6 displays the final knowledge graph after layout, with nodes colored by entity type and edges drawn in light grey to keep the visual focus on the cluster structure. The five entity types separate into spatially distinct regions, with Person and Organization forming the largest clusters and Time and Location forming smaller, denser groups around the periphery. The Event cluster sits roughly at the center of the layout because events are the entities most often connected to all other types through relation triples, which the force-directed algorithm correctly identifies and places accordingly. Cross-cluster edges remain visible without dominating the visualization, and the overall structure is interpretable at a glance even though the underlying graph contains 4892 nodes and 12463 edges. The temporal event chains identified by the timestamp-aware edge weighting can be traced as ordered subpaths within the central Event cluster, which is exactly the kind of exploration the framework was designed to support.

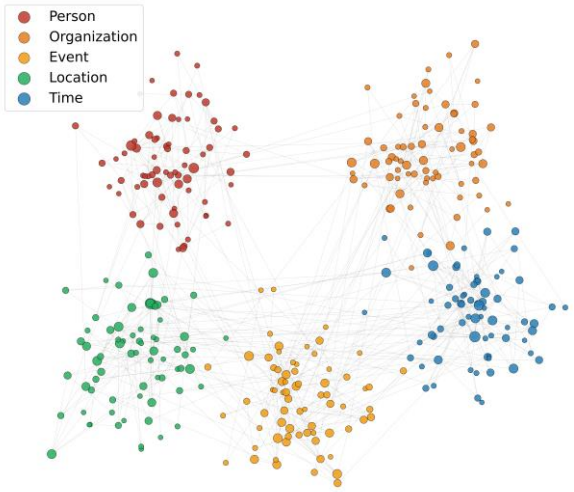

Figure 6: Knowledge graph visualization.

4. Conclusions

This paper develops a unified pipeline for constructing domain-specific knowledge graphs from unstructured text, integrating transformer-based named entity recognition, dependency-guided relation triple extraction, force-directed graph layout, and timestamp-aware temporal event chaining within a single workflow evaluated under consistent annotation. The pipeline applies a fine-tuned pretrained encoder with a conditional random field decoder to identify entities across five types, scores candidate entity pairs through a dependency distance filter combined with an entity-aware neural classifier, and aggregates the resulting triples into a directed multigraph rendered through 500 iterations of ForceAtlas2 layout. Experimental evaluation on a corpus of 4328 documents extracts 18754 entity mentions and 12463 relation triples, achieves a named entity recognition F1 of 91.4% and a relation extraction average precision of 0.892, and exceeds BiLSTM-CRF and joint extraction baselines by 5.2 and 6.9 percentage points respectively. The dependency filter retains 51% of candidate pairs while preserving 96% of true positives at the chosen operating point. Future work will extend the pipeline to streaming text ingestion through incremental graph updates, explore graph neural network embeddings for downstream link prediction, and investigate cross-document coreference resolution to consolidate entity mentions that currently fragment into separate nodes.

References

[1] Kaur, N., Saha, A., Swami, M., Singh, M. and Dalal, R. (2024) BERT-NER: A transformer-based approach for named entity recognition. Proceedings of the 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT), 1-7.

[2] Wu, B., Ding, Z. and Huang, J. (2026) A review of continual learning in edge AI. IEEE Transactions on Network Science and Engineering.

[3] Gardazi, N.M., Daud, A., Malik, M.K., Bukhari, A., Alsahfi, T. and Alshemaimri, B. (2025) BERT applications in natural language processing: A review. Artificial Intelligence Review, 58, 166.

[4] Wu, B., Ding, Z., Ostigaard, L. and Huang, J. (2025) Reinforcement learning-based energy-aware coverage path planning for precision agriculture. Proceedings of the 2025 ACM Research on Adaptive and Convergent Systems (RACS), 1-8.

[5] Salih, M.I., Mohammed, S.M., Ibrahim, A.K., Ahmed, O.M. and Haji, L.M. (2025) Fine-tuning BERT for automated news classification. Engineering, Technology & Applied Science Research, 15, 22953-22959.

[6] Wu, B., Cai, Z., Wu, W. and Yin, X. (2023) AoI-aware resource management for smart health via deep reinforcement learning. IEEE Access, 11, 81180-81195.

[7] Sreenivas, S.C., Chowdhury, S. and Masum, M. (2025) Enhancing clinical named entity recognition via fine-tuned BERT and dictionary-infused retrieval-augmented generation. Electronics, 14, 3676.

[8] Wu, B. and Wu, W. (2023) Model-free cooperative optimal output regulation for linear discrete-time multi-agent systems using reinforcement learning. Mathematical Problems in Engineering, 6350647.

[9] Zengeya, T., Naidoo, K.E., Fonou-Dombeu, J.V. and Gwetu, M. (2025) A multimodal transformer-based fusion model for enhanced relation extraction from texts. IEEE Access.

[10] Chaturvedi, R., Baghershahi, P., Medya, S. and Di Eugenio, B. (2025) Temporal relation extraction in clinical texts: A span-based graph transformer approach. Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 25765-25788.

[11] Mouiche, I. and Saad, S. (2025) Entity and relation extractions for threat intelligence knowledge graphs. Computers & Security, 148, 104120.

[12] Premasiri, D., Ranasinghe, T., Mitkov, R., El-Haj, M. and Frommholz, I. (2025) Survey on legal information extraction: Current status and open challenges. Knowledge and Information Systems, 67, 11287-11358.

[13] Ferreira, P., Martins, E., Silva, J. and Teixeira, P. (2025) Feature selection and XGBoost for enhanced intrusion detection: A comparative study across benchmark datasets. Proceedings of the 2025 13th International Symposium on Digital Forensics and Security (ISDFS), 1-6.

[14] Huang, J., Wu, B., Duan, Q., Dong, L. and Yu, S. (2025) A fast UAV trajectory planning framework in RIS-assisted communication systems with accelerated learning via multithreading and federating. IEEE Transactions on Mobile Computing.

[15] Kumar, R., Singhal, N. and Chhabra, A. (2025) Hybrid optimization algorithm with the combination of PSO and genetic algorithm for task scheduling in cloud computing. E-Learning and Digital Media, 20427530251331082.

[16] Nathiya, N., Rajan, C. and Geetha, K. (2025) A hybrid optimization and machine learning based energy-efficient clustering algorithm with self-diagnosis data fault detection and prediction for WSN-IoT application. Peer-to-Peer Networking and Applications, 18, 13.

[17] Wu, B., Huang, J. and Yu, S. (2026) 'X of Information' continuum: A survey on AI-driven multi-dimensional metrics for next-generation networked systems. IEEE Communications Surveys & Tutorials.

[18] Wu, B., Huang, J., Duan, Q., Dong, L. and Cai, Z. (2025) Enhancing vehicular platooning with wireless federated learning: A resource-aware control framework. IEEE/ACM Transactions on Networking, 33, 1-16.

[19] Isah, M.A. and Kim, B.S. (2025) Question-answering system powered by knowledge graph and generative pretrained transformer to support risk identification in tunnel projects. Journal of Construction Engineering and Management, 151, 04024193.

[20] Wu, B., Huang, J. and Duan, Q. (2025) FedTD3: An accelerated learning approach for UAV trajectory planning. Proceedings of the International Conference on Wireless Artificial Intelligent Computing Systems and Applications (WASA), 13-24.

[21] Auer, S., D'Souza, J., Farfar, K.E., Jaradeh, M.Y., Jiomekong, A., Karras, O. and Vogt, L. (2025) Open research knowledge graph: A large-scale neuro-symbolic knowledge organization system. Handbook on Neurosymbolic AI and Knowledge Graphs, SAGE Publications, 385-420.

[22] Shim, M., Choi, H., Koo, H., Um, K., Lee, K.H. and Lee, S. (2025) OmEGa (Ω): Ontology-based information extraction framework for constructing task-centric knowledge graph from manufacturing documents with large language model. Advanced Engineering Informatics, 64, 103001.

[23] Wu, B., Huang, J. and Duan, Q. (2025) Real-time intelligent healthcare enabled by federated digital twins with AoI optimization. IEEE Network, 1.

[24] Menaouer, B., Fairouz, S., Meriem, M.B., Mohammed, S. and Nada, M. (2025) A sentiment analysis of the Ukraine-Russia War tweets using knowledge graph convolutional networks. International Journal of Information Technology, 1-18.
[25] Pan, D., Wu, B.-N., Sun, Y.-L. and Xu, Y.-P. (2023) A fault-tolerant and energy-efficient design of a network switch based on a quantum-based nano-communication technique. Sustainable Computing: Informatics and Systems, 37, 100827.
[26] Agrawal, S.K. (2026) Adaptive density-aware clustering of high-dimensional patient data in electronic health records. International Journal of Engineering Development and Research, 14, 361-367.