# *University of Ibadan Journal of Science and Logics in ICT Research (UIJSLICTR)*

***ISSN: 2714-3627***

**A Journal of the Faculty of Computing, University of Ibadan, Ibadan, Nigeria**

***Volume 17 No. 1, June 2026***

***journals.ui.edu.ng/uijslictr***
***http://uijslictr.org.ng/***
***uijslictr@gmail.com***

University of Ibadan
Journal of Science and Logics in ICT Research

# On the Construction of Knowledge Graphs from Unstructured Text using Neuro-Symbolic Networks

**[1*]Fasina E. P., [2]Sawyerr B. A., [3]Murainah A.**

*Machine Intelligence Research Group*
*Department of Computer Sciences*
*Faculty of Computing and Informatics*
*University of Lagos, Akoka, Lagos, Nigeria*
*Emails of Authors: [1*]efasina@unilag.edu.ng, [2]bsawyerr@unilag.edu.ng, [3]abdulazeez.murainah@gmail.com*

***Abstract***
In the past few years, Knowledge Graphs (KGs) have emerged as crucial tools for organizing structured information and building complex queries that support applications such as Websites, search engines, recommendation systems, LLMs, and data integration platforms. For KGs to scale with these applications, domain experts must transition from the current practice of curating KGs manually to generating them automatically from the huge volumes of unstructured text generated daily by different sources on the Internet. However, the difficulty of understanding the nuances of natural language and the need to extract precise relationships between entities in unstructured text are some of the major challenges of automatic KG generation. In this work, this challenge is tackled by combining entity and relationship extraction with neural networks (NNs) and the coherent representation of knowledge using symbolic reasoning techniques. An NN combined with a symbolic reasoning module can then be used for the accurate generation of KGs from a variety of unstructured text sources. Our methodology involves designing a generic neuro-symbolic network (GNSN) for KG extraction from a given text and then implementing a variant using open-source libraries. The implemented GNSN variant is trained and evaluated on WebNLG+2020 Text-to-RDF benchmark with a measured precision of 82.2%, recall of 82.4%, and an F1 score of 82.3%, which is competitive with other neuro-symbolic networks reported in the literature. This work aims to advance the state of the art in automatic KG generation from unstructured text.

**Keywords:** Knowledge graph, Artificial Intelligence, Symbolic AI, Neural Networks, Machine Learning, Decision-Making, Explainable AI

## 1. Introduction

The rapid growth in the adoption of artificial intelligence (AI), especially Large Language Models (LLMs), has created the need to process vast amounts of unstructured data. Unfortunately, unstructured data is not machine-readable. Knowledge graphs are new machine-readable representations of structured data that have been deployed in many applications and fields that include recommendation systems, personal assistants, information retrieval, large-scale data analysis, and search engines [1]. What is required at scale are methods that automatically generate machine-readable structured data, such as KGs, from unstructured data. A major obstacle to meeting this requirement is the difficulty of automatically generating structured data from unstructured text and placing it in machine-readable formats such as Freebase, DBpedia, Yago, Google's knowledge graph, and Wikidata.

Currently, most KGs are manually curated, and a lot of information is not represented in KGs, because the volume and speed at which information is generated makes human curation infeasible. This has resulted in a widening gap in the representation of real-world entities in KGs [02]. The construction of KGs from semi-structured and unstructured data is challenging because of the nuances and the complexity of understanding natural languages. It has therefore become imperative to develop new specialized techniques for knowledge curation and extraction that break from relying on current end-to-end Natural Language Processing (NLP) systems that require humans in the loop.

Fasina E. P., Sawyerr B. A., and Murainah A. (2026). On the Construction of Knowledge Graphs from Unstructured Text using Neuro-Symbolic Networks. *University of Ibadan Journal of Science and Logics in ICT Research (UIJSLICTR)*, Vol. 17 No. 1, pp. 121 - 145

A viable approach is adopting Neuro-Symbolic AI (NSAI) systems that hybridize neural networks and symbolic reasoning pipelines to process complex unstructured data. In these systems, structured information will be extracted with neural methods and then refined and organized with symbolic methods into valid KGs [3]. NSAI architectures combine the strong pattern-recognition capabilities of Neural Networks (NNs) with the exact reasoning capabilities of Symbolic Modules (SMs) and are therefore most likely to provide the systems that will solve the problem of automatically generating KGs from unstructured text. Neural networks are effective at discovering structure in large data sets and are best suited to extracting entities and their relationships from unstructured text [4]. The other type of modules used are symbolic reasoning modules, which ensure coherent knowledge representation and logical reasoning [5]. This integration of neural networks and symbolic reasoning pipelines will enable the design of NSAI systems for large-scale automatic generation of KGs that accurately model real-world information in unstructured data.

This paper reports on the process of automating KG construction from unstructured text using a neuro-symbolic architecture. The methodology adopted in this work begins with the design of a generic neuro-symbolic model, followed by the implementation of its variant using open-source components. The neural module is then trained to extract entities and relationships, while the symbolic module, handcrafted from open-source ontologies such as DBpedia, Wikidata, and OpenCyc, fine-tunes the knowledge graph constructed by the neural module. The goal is to overcome the limitations of neural and symbolic engines while preserving explainability, interpretability, and scalability, which are essential for applications in healthcare, finance, and law [6]. Finally, the NSAI model is evaluated using precision, recall, and F1-score to gain insights into how well the model performs at generating accurate and comprehensive knowledge graphs.

The extraction of accurate and precise KGs from large and unstructured text remains a difficult task. These difficulties arise from the intrinsic complexity of natural languages, such as ambiguity, context-dependency, and the advanced comprehension of languages required to disambiguate entities and relationships. Despite the challenges of generating Knowledge Graphs (KGs) from unstructured text, tremendous progress has been made in the extraction of accurate KGs from diverse, large, and unstructured data sources. Current approaches often fail on some of the most crucial tasks:

a) *Entity Recognition and Disambiguation*: Identifying and properly assigning the correct entity type to entities within unstructured text, but most models are weak on ambiguities and variations in entity names.

b) *Relationship Extraction:* Extracting meaningful relationships between entities, but most models fail on ambiguities and variance in entity names. Relationships are difficult to extract because they can be expressed in many different ways in natural language, and symbolic reasoning is interpretable and can reason but has lower accuracy in capturing patterns.

c) *Module Integration*: Symbolic reasoning is interpretable and can reason, but with lower accuracy in capturing patterns; neural networks are excellent in capturing patterns, but not as interpretable or capable of reasoning as symbolic approaches. On the other hand, symbolic approaches have trouble scaling up to accommodate new data.

d) *Robustness, Uncertainty Quantification and Brittleness*: A combination of a neural and a symbolic component can inherit the brittleness of both: The neural component is vulnerable to adversarial inputs and OOD data, while the symbolic component is prone to failure when the hand-crafted rules don't apply at inference time.

e) *Uncertainty Quantification*: Knowing when the system doesn't know. Uncertainty quantification is not well addressed in almost all existing Neuro-Symbolic AI (NSAI) architectures.

f) *Explainability and Trustworthiness Gaps*: Comparatively, little work has been done on the explainability of NSAI architectures, and significant gaps remain in the trustworthiness and meta-cognition of these models [7][8][9][10][11][12][13][14][15].

These shortcomings call for a new approach that combines the best of both worlds, neural networks and symbolic reasoning, to create

more robust and accurate knowledge graphs from unstructured text.

## 2. Literature Review

In this section, knowledge graphs, neural networks, symbolic reasoning architectures, neuro-symbolic architectures, and recent methods for the automatic generation of KGs from NSAI architectures are reviewed.

### *2.1 Knowledge Graphs*

KGs provide a basic mechanism for representing structured information across various domains, including search engines, recommendation systems, large-scale data analysis, and personal assistants [1]. A knowledge graph is made up of nodes that represent entities such as people or other objects, and edges that represent relationships between entities. A KG, formally defined as a set of triples (subject-relation-object) is an intuitive but powerful means for organizing data. Popular knowledge bases for storing knowledge graphs are Freebase, DBpedia, Yago or Google's Knowledge Graph [16].

KGs are flexible and can capture diverse types of relationships, making them highly valuable in fields such as AI and NLP. For instance, they can be beneficial in information retrieval by providing contextual and accurate responses to queries [17]. The ability to integrate and manage a lot of information makes KGs an important part of AI systems. Despite these benefits, KGs are mostly manually curated, which is neither scalable nor comprehensive [2].

The rapid generation of text data brought about by the wide deployment of AI applications has resulted in a significant gap between generated unstructured data and structured data manually curated for knowledge representation. Although a neural network-only approach can learn semantic similarity in data through representation learning or embeddings, it is difficult to interpret the learned knowledge. Likewise, a traditional symbolic-only approach is inflexible and insufficiently scalable to handle unstructured or semi-structured data [8].

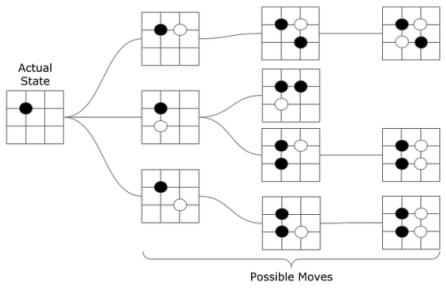


Figure 1: Symbolic Reasoning Network

### *2.2 Neuro-Symbolic AI Architectures*

NSAI has attracted significant interest for its promise of overcoming the shortcomings of both stand-alone neural networks and symbolic reasoning modules. Traditional AI consists of knowledge inference and decision-making that are based on logic and predefined rules. These systems, illustrated in Figure 1, are called symbolic reasoning systems (SRS). These systems are transparent and interpretable but are not scalable and sufficiently flexible to efficiently process large volumes of generated unstructured data [18]. In contrast, the neural networks of modern machine learning, such as in Figure 2, are very good at learning and are used extensively in pattern recognition tasks, but are “black boxes” with little explanation or logic to interpret [3].

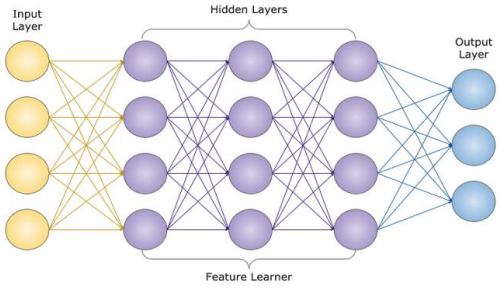


Figure 2: Neural Networks

The goal of the NSAI architecture is to bring together the good points of these two methods. NSAI combines the powerful learning capabilities of neural networks with the structured reasoning capabilities of symbolic systems, allowing systems to learn from data and reason about their learning in a human-readable and understandable manner [5]. This integration provides more flexibility in managing unstructured data, while retaining the logical consistency and transparency of symbolic systems.

Neuro-Symbolic architectures can be classified by Kautz [19] using the following taxonomy:

**Type 1: Symbolic Neuro Symbolic** – A neural network takes a symbolic input, such as a sequence of words that have been converted to vectors, and reconstructs another symbolic representation as output using a softmax operation; examples include word2vec [20], GloVe [21] and GPT-3 [22].

**Type 2: Symbolic[Neuro]** – Here, symbolic problem solvers query or employ neural networks as pattern recognition subroutines during reasoning; examples include AlphaGo [23], Ness [24], PLANS [25], VISPROG [26], HugginGPT [27], and ViperGPT [28].

**Type 3: Neuro|Symbolic** – Combines a neural system with a symbolic system in a pipeline where each module solves complementary tasks. In some architectures, the symbolic reasoning module provides feedback to the neural system; examples include Neuro-Symbolic-Vector-Architecture (NSV) [29], NeuPSL [30], NSCL [31], NeurASP [32], Abductive Learning (ABL) [33], Neural-symbolic VQA [34]. The majority of NSAI architectures, including the architecture presented in this paper, fall into this category [13].

**Type 4: Neuro:Symbolic→Neuro –** In this architecture, symbolic rules are incorporated into neural networks to guide learning, i.e., knowledge or domain expertise is encoded as symbolic rules that act as constraints on the neural network's outputs; examples of this class of NSAI architecture include Logical Neural Networks [35], Deep Learning for Symbolic Mathematics [36], and DILP [37].

**Type 5: Neuro$_{\text{Symbolic}}$** – In this type of architecture, symbolic logic rules are mapped to embeddings that act as soft constraints on the neural network's loss function. Examples are the use of Tensor Product Representation [38], Logic Tensor Networks [39], Deep Ontology Networks [40], Learning Logistics Circuits [41], and Lifted Rule Injection [42].

**Type 6: Neuro[Symbolic]** – Unlike **Symbolic[Neuro]** architectures, which use neural networks as pattern recognition subroutines, these neural networks incorporate symbolic reasoning through embedded symbolic reasoning modules; examples include Neural Logic Machines [43], SATNet [44], Generate-Verify-Repair [45], Logicseg [46], and Differentiable Neural Computer (DNC) [47].

### 2.3 Constructing Knowledge Graphs with Neuro-Symbolic Architectures

Knowledge graph construction from unstructured text has benefited from the evolution of AI systems, from the 1990s to the present. Three AI paradigms emerged during this period. First-generation symbolic systems, such as WordNet and ConceptNet, developed from the 1990s through the 2010s, used hand-crafted rules and shallow linguistic patterns [9]. They achieved high precision but had limited recall. Second-generation neural or sub-symbolic systems such as BioBERT and SapBERT, developed from the 2010s to around 2022, introduced learned representations and improved named entity recognition through domain-adaptive pretraining. These systems, however, required expensive supervised training and failed to generalize beyond a predefined schema.

Later realizations of these systems relied on transformer-based extraction pipelines and entity-linking methods and offered better quality KG generation at the cost of requiring a fixed relation schema or domain-specific supervision. Third-generation neuro-symbolic systems, developed from 2022 to the present, combine large language models with symbolic reasoning components such as ontologies, logic-based reasoners, and rule engines to overcome the complementary weaknesses of the first two generations. Systems such as GRID [48] and LOKE [49] are empirical demonstrations of this approach. In this section, different neuro-symbolic architectures for creating knowledge graphs from unstructured text are reviewed.

Rahdari and Brusilovsky [50] investigated the integration of learning and reasoning by creating KGs from unstructured academic texts. They demonstrated the potential of neuro-symbolic AI to scale up the creation of KGs while preserving accuracy and relevance by using a combination of neural networks for entity extraction and symbolic techniques for knowledge representation.

McCusker [49] developed a neuro-symbolic system, LOKE-GPT, that solved what was described as the Open Knowledge Extraction (OKE) problem, which differs from the Open Information Extraction (OIE) problem, by linking extracted entities to canonical, queryable entities (e.g., Wikidata entities or nodes) in the construction of real-world KGs. The neural

component is a GPT model driven by carefully engineered prompts for entity extraction. The symbolic component is a naïve linking approach that resolves GPT's extracted raw entities to Wikidata identifiers, then generates graph edges using available entity information. LOKE-GPT consistently outperformed symbolic KG generators because of the marginal value of symbolic grounding, a significant contribution of neuro-symbolic systems to automatic KG generation.

Stutz et al. [51] develop a user-centered type 3 neuro-symbolic architecture that puts users in the loop to guide the generation of semantically structured KGs from unstructured text. They evaluated the neuro-symbolic system qualitatively and quantitatively to show that inexperienced users can generate KGs with the same high quality in a fraction of the time it will take experts and that the system was evaluated to be high on the usability scale.

Loconte et al. [52] propose a type 3 neuro-symbolic framework for question answering applications. The framework creates KGs that are grounded in ontologies by combining open-domain LLM-based extraction, embedding-based canonization of types and predicates, and targeted LLM-based correction of ontology violations. The late correction of ontology violations by LLMs eliminates repeated LLM calls, reduces the use of tokens, improves KG consistency, and preserves the quality of question answering. The authors report that their OAK+MEND method achieved ontology consistency rates of up to 98.4% for triples and 96.8% for qualifiers.

Servedio et al. [53] introduce EpisTwin, a type 3 neuro-symbolic framework with generative reasoning that is grounded in a user-centric Personal Knowledge Graph (PKG). It generates semantic triples from heterogeneous cross-application data using Multimodal Language Models. For inferencing, it uses an agentic coordinator that performs complex reasoning over the PKG by combining Graph Retrieval-Augmented Generation with Online Deep Visual Refinement to dynamically ground symbolic entities in their raw visual context. They report robust results on state-of-the-art judge models.

GRID [48] trains two extractors based on Qwen3-4B-Instruct-2507 language model, the Task-bank Reward and End2End Reward models, for the generation of ontology -guided security knowledge graphs from unstructured long-form security text such as Cyber Threat Intelligence articles, technical blogs, and threat reports. Both extractors are trained with reinforcement learning. The Task-bank Reward extractor is trained on cheap symbolic checks of four-option multiple-choice questions (for precision checks) and triple-level regex matching (for recall). The End2End Reward extractor is trained on a purely neural LLM-as-judge reward. A symbolic ontology-guided extractor constrains the inputs from the Tank-bank Reward and End2End Reward extractors for the final triple output. The symbolic Tank-bank Reward extractor consistently outperforms the neural End2End Reward extractor, confirming the observation that neuro-symbolic systems outperform purely neural systems.

An ontology-guided neuro-symbolic system is presented by Mohammadi et al. [54] for the generation of knowledge graphs from unstructured text in material science. The system uses OpenAI's GPT models, the neural component, to read unstructured scientific text, understand the context, and extract entities and their relationships. The unstructured text is processed in chunks of about 2048 tokens. The symbolic component constrains extracted entities and their relationships using EMMO, the Elementary Multiperspective Material Ontology, and a predefined schema with strict entity classes such as Material, Manufacturing, Measurement, Property, etc., and relationship types such as 'is-manufacturing-input', has-property', etc. The neural network proposes the instances while the symbolic pipeline validates, deduplicates, and commits the structure into a machine-interpretable graph. The authors reported an F1 score of 0.878 over a collection of materials science text.

By combining the strengths of neural networks and symbolic AI, neuro-symbolic architecture offers a powerful approach for extracting knowledge graphs from unstructured text. Although there are still issues, especially with scalability and transparency, these systems have the potential to fully automate KG creation and enhance data representation accuracy. Further research in this area should focus on enhancing designs, addressing current challenges, and exploring designs for more complex and realistic applications.

## 3. Neuro-Symbolic Networks for Generating Knowledge Graphs

The dual-system theory (also called the dual-process theory) is a theory of cognitive psychology that suggests that the human brain thinks and makes decisions in two modes [55]. Kahneman [55] referred to these two modes as: System 1 and System 2. System 1 processes thoughts automatically, intuitively, and unconsciously in the brain. It works immediately without effort of will. Uses abductions (heuristics) and memories to make the best of everyday life. System 2 is the conscious, deliberate, and logical way of thinking in the brain. It takes mental effort; it is turned on when we do complex, novel, and non-routine tasks.

In the division of tasks in type 3 neuro-symbolic architecture, the Neural Network Module (NNM), which is intuitive and statistical, maps onto System 1 for high-dimensional perception, while the Symbolic Reasoning Module (SRM), which is logical and analytical, maps onto System 2 for logical deduction [56]. The process of transforming noisy unstructured text into KGs requires models that capture context, followed by logical frameworks that handle consistency. The neuro-symbolic approach of combining data-driven neural extraction with constraint-based logic layers is highly effective for mitigating data gaps and noise in KGs [57]. A major challenge of building neuro-symbolic systems is bridging the continuous representation spaces of neural networks (e.g. word embeddings or contextual tensors) with the discrete symbolic sets of symbolic reasoning modules (e.g. triples) [58].

### *3.1 A Generic Type 3 Neuro-Symbolic Network Architecture for KG Generation*

Based on current literature on Neuro-Symbolic and Knowledge engineering Figure 3 is a realization of a generic Type 3 dual-module Neurosymbolic pipeline [56][57][58]. It has a neural network module, and a symbolic reasoning module that takes an unstructured input text $\mathcal{D}$ and transforms it into a formally verified Knowledge Graph $G = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ where $\mathcal{E}$ represents canonical entities, $\mathcal{R}$ represents defined relations and $\mathcal{T}$ represents the set of facts or triples $(s, p, o)$. The generic model consists of the following: 1) An NNM with three subtasks: Contextual Tokenization and Dense Embedding, Neural Entity Spotting and Span Representation, and Neural Relation Predicate Estimation. 2) A Neural Symbolic Interface and 3) an SRM with four subtasks: Ontological Alignment and Entity Resolution, Axiomatic and Rule-Based Relation Hardening, Logical Inference and Graph Coherence Verification, and Graph Serialization.

**Contextual Tokenization and Dense Embedding:** Here, the unstructured text input is digested by a deep transformer backbone, e.g., a lightweight fine-tuned bidirectional encoder or an encoder-decoder model. The text is converted into a sequence of contextualized hidden states:

$$H = \{h_1, h_2, \dots, h_n\}, h_i \in \mathbb{R}^d$$

**Neural Entity Spotting & Span Representation:** A token-classification head that uses a softmax classification over BIO (Beginning-Inside-Outside) tagging schemes identifies candidate entity spans. Once a span is identified (e.g. tokens $i$ to $j$), a pooling mechanism or a boundary-attentive network aggregates the hidden states into a single entity. This subtask outputs a set of continuous entity representation vectors:

$$e_k = Aggregate(h_i, \dots, h_j)$$

**Neural Relation Predicate Estimation:** A relational multi-head attention layer computes an interaction tensor for every valid pair of entity span $(e_1, e_2)$. This subtask outputs a dense relation logit vector $\boldsymbol{r}_{1,2}$ that represents the probabilistic strength of potential interaction between two entities in continuous space.

**Neuro-Symbolic Interface:** This unit interfaces the neural network with the symbolic reasoning module in two steps: **Vector Similarity Quantization** and **Soft-to-Hard conversion**. In the first step, the continuous entity vectors $e_k$ are passed through a vector cosine similarity or $K$-Nearest Neighbors (k-NN) indexing layer against an Ontological Vector Registry. The highest scoring matches are transformed into symbolic candidates with associated confidence scores formatted as:

$$\text{Candidate}(s, p, o) = [\text{Score}]$$

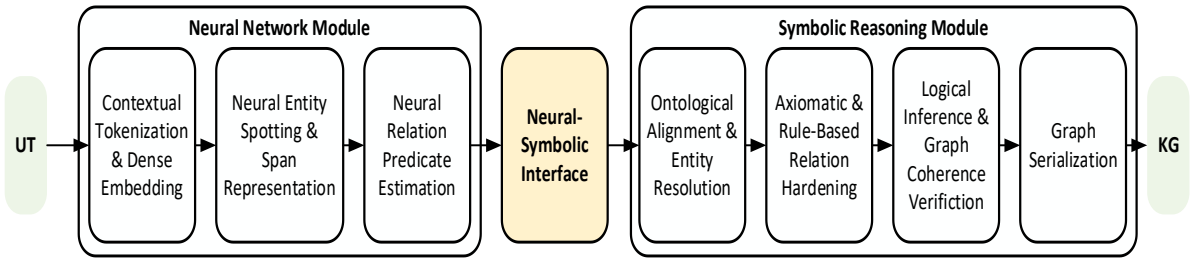


**Figure 3:** A Generic Type 3 Neuro-Symbolic Network for Generating Knowledge Graphs from Unstructured Text

**Ontological Alignment & Entity Resolution:** This sub-module cross-references entity symbols against a predefined TBox (Terminology Box/Ontology Schema). It executes string-similarity lookups and maps entities symbols to identifiers such as Custom Corporate IRI (Internationalized Resource Identifier) [59] or Wikidata ID [60]. If two distinct neural spans resolve to the same canonical URI based on deterministic rule matching, they are unified (e.g. "Artificial Intelligence" and "AI" will map to the same node under strict contextual rules).

**Axiomatic & Rule-Based Relation Hardening:** The SRM filters the relation logits $\boldsymbol{r}_{1,2}$ using strict ontological constraints such as domain and range validations. If the neural module predicts a relation that violates the structural schema, the SRM overrides it. It uses Description Logic (DL) or Horn Clauses to check for constraints such as:

$$\forall x, y: \text{hasDirector}(x, y) \implies \text{Organization}(x) \wedge \text{Person}(y)$$

If the NNM assigns entity type "Vehicle" to $x$, the relation "hasDirector" is structurally rejected.

**Logical Inference & Graph Coherence Verification:** A symbolic reasoning engine such as Datalog, a Retes-based rule engine, or an OWL reasoner processes hardened triples to infer implicit facts and check for logical contradictions. The logic applied are:

1) *Symmetry/Transitivity:* If $A \xrightarrow{locatedIn} B$ and $B \xrightarrow{locatedIn} C$, the SRM injects $A \xrightarrow{locatedIn} C$;

2) *Consistency Checking:* If a rule dictates that an individual cannot be both a `Student` and a `FullTimeProfessor` at the same moment, and the NNM asserts both, the SRM triggers a contradiction resolution sub- routine and drops the triple with the lower neural confidence weight.

**Graph Serialization:** At this stage, validated, deduplicated, and logically enriched facts are structurally committed to a graph storage layer to generate a clean and queryable Knowledge Graph in standard deterministic formats like Resource Description Framework (RDF) triples, OWL, or Property Graph JSON-LD.

### *3.2 Proposed Neuro-Symbolic Network Architecture*

Based on GNSN and [61], a neuro-symbolic network (NSN) is proposed (**Figure 4**) consisting of three components: a neural network module, a neuro-symbolic interface, and a symbolic reasoning module. The NNM consists of three subtasks: Text Tokenization and Embedding Generation, Named Entity Recognition (NER) and Relation Extraction (RE). The SRM consists of five subtasks: Entity Normalization, Schema Constraints Validation, Triple Generation, Logical Inference and Triple Storage. It. integrates the pattern recognition capabilities of neural networks with the structured reasoning of symbolic AI module via the neuro-symbolic interface to construct knowledge graphs (KGs) from unstructured text. This hybrid architecture provides a scalable and interpretable alternative for automated KG creation by addressing the drawbacks of both methods. These components interact to extract entities and relationships from unstructured data and organize them into structured knowledge graph (KG). Table 1 compares the GNSN and the NSN neuro-symbolic pipelines.

*Neural Network Module (NNM):* The neural network module is responsible for extracting relevant entities and relationships from the input text. This module leverages pre-trained language models, such as BERT or RoBERTa, to perform Named Entity Recognition (NER) and Relation Extraction (RE). These models

have been fine-tuned on domain-specific datasets to ensure accurate identification of entities (e.g., people, organizations) and relationships (e.g., works for, located in). The neural model transforms the input text into vector representations, using the following subtasks:

**Text Tokenization**

The stream of characters from unstructured text is organized into words or subwords using a fixed vocabulary. This converts variable-length text into units that a neural network can process as inputs. This subtask transforms a raw text string $S$ of length $L$ to a sequence of discrete token IDs, and can be formalized as follows:

$$T = [t_1, t_2, \dots, t_n], \qquad t_i \in \{1, \dots, V\}$$

where the vocabulary size V is typically in the range 32,000 to 50,000 for modern tokenizers, and $N$ is the sequence length of generated token IDs. The sequence $T$ is mapped to a dense weight matrix:

$$\mathbf{E} \in \mathbb{R}^{N \times d}$$

where $d$ is the size of the model's hidden dimensions.

**Table 1**: A Comparison of the GNSN and PNSN neuro-symbolic pipelines

| Subtask | Generic Neuro-Symbolic Network (GNSN) | Proposed Neuro-Symbolic Network (PNSN) | Notes on Mapping |
|---|---|---|---|
| **1** | Contextual Tokenization & Dense Embedding | Text Tokenization | In the proposed NSN, the Tokenization subtask is followed by the Embedding Generation subtask, but both tasks are implemented in the GNSN as a single continuous pipeline that utilizes a unified transformer backbone. |
| **2** | | Embedding Generation | |
| **3** | Neural Entity Spotting and Span Representation | Named Entity Recognition | Entity boundaries are isolated by both subtasks, but the GNSN focuses on converting the token spans into aggregated, fixed-width dense vectors. |
| **4** | Neural Relation Predicate Estimation | Relation Extraction | Both generate as outputs statistical distributions over potential predicates |
| **5** | Neural-Symbolic Interface | Neural-Symbolic Interface | Both interface the neural network with the symbolic reasoning module. |
| **6** | Ontological Alignment and Entity Resolution | Entity Normalization | Both are equivalent; they normalize text variants by ensuring that they are mapped to strict ontological IRIs, i.e., the text variants of each entity are uniquely represented in the KG. |
| **7** | Axiomatic and Rule-Based Relation Hardening | Schema Constraint Validation | This subtask in the GNSN is equivalent to the Schema Constraint Validation subtask followed by the Triple Generation subtask. Relations are filtered with ontological constraints before they are committed to triples. |
| **8** | | Triple Generation | |
| **9** | Logical Inference and Graph Coherence Verification | Logical Inference | Both achieve the same goal of applying symbolic engines for resolution of transitivity and contradiction. |
| **10** | Graph Serialization | Triple Storage | Both commit triples to RDF, JSON-LD, or Property Graph databases. |

**Named Entity Recognition**
In this subtask, the embedding of tokens in context is evaluated and used to classify specific token spans as real-world entities or concepts from a set of predefined categories (e.g., Organization, Person). It does this by assigning a conditional probability to a sequence of labels $Y = [y_1, y_2, \dots, y_N]$ using a Softmax or Conditional Random Field (CRF) layer

$$P(Y|\boldsymbol{E}) = \prod_{i=1}^{N} P(y_i|\boldsymbol{h}_i)$$

The output of this subtask is a set of candidate entity spans: $\mathrm{E}_{raw} = \{(start, end, \text{label})\}$.

**Relation Extraction**
This subtask evaluates pairs of detected entity spans in the contextual vector space to predict the semantic relationship that connects them. For any two entity vectors $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ the probability distribution $\mathbf{p}_{rel}$ over a raw predicate vocabulary $\mathrm{R}_{raw}$ is computed as

$$\mathbf{p}_{rel} = \text{softmax}(\mathbf{W}_r[\mathbf{e}_1; \mathbf{e}_2] + \mathbf{b}_r) \in \mathrm{R}^{|\mathrm{R}_{raw}|}$$

The output is a dense relation logit vector $\mathbf{r}_{1,2}$ that represents the probabilistic strength of potential interactions between two entities in continuous space.

**Schema Constraint Validation**
This subtask applies ontological (domain and range) constraints to filter relation logits $\mathbf{r}_{1,2}$.

If the neural network predicts a relation that violates the structural schema, it is removed.

**Triple Generation**
Creates a discrete set of facts $\mathrm{T} = \{(s, p, o)\}$ from normalized entities and structural relations that mirror graph models of the input data, where $s, o \in O_E$, $p \in O_R$ and $O_R$ is the ontological relation schema.

**Triple Generation**
Creates a discrete set of facts $\mathrm{T} = \{(s, p, o)\}$ from normalized entities and structural relations that mirror graph models of the input data, where $s, o \in O_E$, $p \in O_R$ and $O_R$ is the ontological relation schema. These triples represent the semantic relationships between entities and are the building blocks of KGs.

**Logical Inference**
Executes deterministic rules, Horn clauses, or Description Logic over generated facts to deduce implicit relationships and flag structural contradictions based on ontological axioms. For example:

$$\forall \boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3 \in O_E: (e_1, r_{12}, e_2) \wedge (e_2, r_{23}, e_3) \Rightarrow (e_1, r_{13}, e_3)$$

The initial set T is expanded to $\mathrm{T}_{closure}$ after this subtask. $\mathrm{T}_{closure}$ is validated not to contain contradictions, and it is the deductive closure of T.

**Triple Storage**
This subtask transfers the inference-enriched and validated set of facts $\mathrm{T}_{closure}$ to physical memory using standardized semantic graph databases (or RDF/W3C serialization syntax) for indexing and querying.

### *3.3 Mathematical Foundation of the System*

The network relies on three key optimization steps:
a) First, the system minimizes the cross-entropy loss for entity recognition. For each token in the input sequence, the loss is calculated as:

$$L_{NER} = -\sum_{i=1}^{n} \log P(\hat{y}_i = y_i|w_i)$$

where $\hat{y}_i$ is the predicted label for the token $w_i$, and $y_i$ is the true label.
b) Second, the system minimizes the cross-entropy loss for relation extraction. The loss function is similarly defined as:

$$L_{RE} = -\sum_{e_i, e_j} \log P\left(\hat{r}_{i,j} = r_{i,j}|e_i, e_j\right)$$

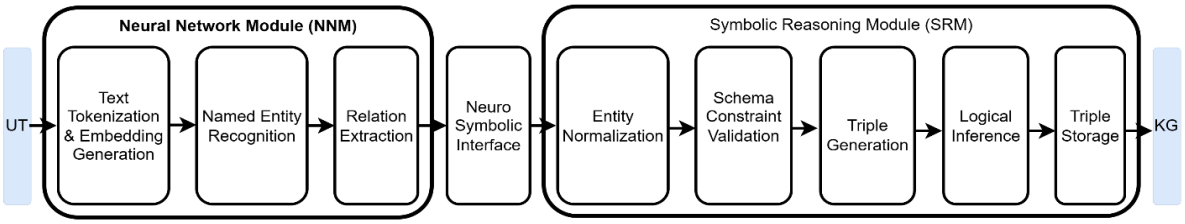


**Figure 4:** Proposed Neuro-Symbolic Architecture

where $\hat{r}_{i,j}$ is the predicted relationship and $r_{i,j}$ is the actual relationship between entities $e_i$ and $e_j$.

c) Third, to ensure logical consistency and eliminate contradictions, the symbolic reasoning module applies logical rules and constraints to generated triples. It involves checking for contradictions or incomplete relationships in KGs and applying logical inference to fill in missing information. Constraints can be formalized as logical rules:

$$\textbf{if } (e_i, r_{ij}, e_j) \textbf{ and } (e_j, r_{jk}, e_k) \textbf{ then } (e_i, r_{ik}, e_k)$$

The symbolic method ensures that substantial noise is removed from triples generated by the neural network, and also that the KG is both data-driven and logically coherent.

### *3.4 Benefits of the Neuro-Symbolic Approach*

The neuro-symbolic approach allows us to leverage the best of both worlds by combining the neural network's flexibility and scalability with the symbolic module's reasoning and logical consistency. This hybridization enhances the explainability of the system and improves the accuracy of KG construction. By combining learning and reasoning, a balance between data-driven insights and structured knowledge representation is achieved, making the system sufficiently robust to handle complex, real-world applications such as answering questions or integrating data [62].

## 4. Experiment

In this section, experiments conducted to evaluate the performance of the proposed NSN on the WebNLG+2020 Open-domain Text-to-RDF benchmark [63] are presented. The implementation of NSN is presented in Section 4.1, followed by the WebNLG+2020 benchmark in Section 4.2, and the methodology for evaluating NSN in Section 4.3. WebNLG+2020 is a strong choice for demonstrating the proposed NSN's engineering maturity for the following reasons: Firstly, it is large enough to train the proposed NSN learnable components; Secondly, its triples are compatible with the DBpedia Knowledge Base [64], and this matches the design of NSN's symbolic reasoning module; Thirdly, it has an established leaderboard that gives NSN a direct, credible ranking against both neural and symbolic systems; Fourthly, its crowdsourced text gives more variety than templated synthetic text. Its main limitations are as follows: Firstly, it requires a full training infrastructure; Secondly, it has a narrow ontology coverage; its leaderboard is dominated by large seq2seq neural systems, which makes one-to-one comparison with a modular system such as the NSN difficult.

### 4.1 Implementation of NSN

The proposed neuro-symbolic network is implemented with the Python programming language [65], tools from its extensive libraries, and tools from open-source AI platforms such as HuggingFace [66]. Below is a description of the implementation for each subtask in the NSN pipeline using NNM and SRM labels, with its summarization in Table 2.

*NNM-1 (Text Tokenization)*: spaCy rule-based tokenizer handles sentence segmentation and surface tokenization. It feeds its output into the BertTokenizerFast high-performance tokenizer from HuggingFace's transformers library [66] for WordPiece subword tokenization, since this is what the BERT embedding layer expects.

*NNM-2 (Embedding Generation)*: Implemented using the BertModel.from_pretrained('bert-base-cased') method. Cased option is preferred because WebNLG entity names are sometimes capilatlized.

*NNM-3 (Named Entity Recognition):* The spaCy's transformer pipeline (en_core_web_trf) is fine-tuned on WebNLG's 16 seen categories using its triple-derived entity spans as training labels. For unseen categories **GLiNER** [67] may be added as a supplement because of its competitive zero-shot capability.

*NNM-4 (Relation Extraction)*: A BertForSequenceClassification head is fine-tuned as a span-pair classifier on the WebNLG's relation vocabulary, i.e., the set of dbo**:** properties appearing in the training triples. An alternative is the REBEL seq2seq extractor [68] remapped to dbo**:** for a lightweight single-pass design.

**Table 2:** Python Tools used for the Implementation of NSN Components and WebNLG+2020 Specific Roles

| Component | Subtask | Python & Open-Source Tool(s) | WebNLG+2020 Specific Role |
|---|---|---|---|
| NNM-1 | Sentence Segmentation and **Text Tokenization** | spaCy tokenizer + BERT WordPiece (transformers) | Handles crowdsourced text surface forms. |
| NNM-2 | Embedding Generation | BERT; bert-based-cased (HuggingFace) | Cased embeddings preserve entity-name signal |
| NNM-3 | Named Entity Recognition (NER) | spaCy en_core_web_trf + GLiNER (zero-shot) | GLiNER covers unseen test categories |
| NNM-4 | Relation Extraction (RE) | BERT; BertForSequenceClassification span-pair head; REBEL (alt.) | Predict dbo: property vocabulary |
| NNM/SRM Interface | Ontological Vector Registry | pyRDF2Vec (DBpedia embeddings) + FAISS (ANN index) | Bridges BERT vectors to DBpedia URI candidates |
| SRM-1 | Entity Normalization | OWL / spaCy; DBpedia Spotlight (spacy-dbpedia-spotlight) | Disambiguates to canonical DBPedia URIs |
| SRM-2 | Schema Constraint Validation | owlready2 + HermiT reasoner, DBpedia OWL ontology | Filters relations via dbo: domain/range axioms |
| SRM-3 | Triple Generation | Custom Python Assembler / rdflib.Graph.add() | Assembles validated (s, p, o) triples |
| SRM-4 | Logical Inference | OWL; owlready2 (OWL RL); Apache Jena Rule Engine (alt.) | Applies transitivity / subproperty inference |
| SRM-5 | Triple Storage | RDFLib | Exports RDF/Turtle + WebNLG flat-triple eval format |

*NNM/SRM Interface (Ontological Vector Registry)*: pyRDF2Vec [69] pre-computes embeddings for every DBpedia entity and class appearing across WebNLG's 19 categories by taking random walks over DBpedia SPARQL endpoint and training a Word2Vec-style encoder on the resulting entity-relation sequences. Generated vectors are stored in a Facebook AI Similarity Search (FAISS) index [70] for fast approximate nearest neighbor lookup.

*SRM-1 (Entity Normalization)*: The DBpedia Spotlight returns canonical DBpedia URIs with similarity scores for each entity span. Since WebNLG's gold triples are DBpedia resources, this tool disambiguates among the FAISS-retrieved candidates from the NNM/SRM interface using DBpedia's own confidence score (prominence, topical relevance, contextual ambiguity) rather than raw embedding similarity alone, giving a second, independent assessment that complements the vector registry.

*SRM-2 (Schema Constraints Validation)*: The HermiT reasoner [71] and owlready2 loaded with the DBpedia ontology (dbo: OWL file) are combined to check that the domain and range classes of each candidate relation from NNM-4 match DBpedia types of the normalized subject and object entities from SRM-1. Relations that violate dbo: domain/range axioms are filtered out before triple assembly.

*SRM-3 (Triple Generation)*: A Python function (or a direct rdflib.Graph.add() call) assembles validated triples (subject_uri, dbo:property, object_uri) from the output of SRM-1 and SRM-2.

*SRM-4 (Logical Inference)*: The owlready2's sync_reasoner() method applies OWL RL inference (transitivity, subPropertyOf, equivalentProperty axioms in DBpedia's ontology) on the triple set. A recommended alternative is Apache Jena's rule engine [72] for inference rules beyond what DBpedia OWL axioms encode and Datalog-style forward-

chaining rules that OWL DL reasoners like HermiT do not cover.

*SRM-5 (Triple Storage)*: RDFLib is used to serialize the final triple set in the Turtle/RDF format for general use and in the flat subject | predicate | object text format for the WebNLG-Text-to-triples scoring script.

### 4.2 Benchmark

WebNLG+2020 [63] is a large, crowdsourced shared-task dataset with 42,045 texts in total, split into 35,426 for training, 4,464 for development, and 2,155 for testing. The texts are paired with 15,630 RDF triple sets (13,211 training, 1,667 dev, 752 test). It comprises of two tasks:
1) RDF-to-text generation (G2T) and
2) Text-to-RDF semantic parsing (T2G).

T2G is the task relevant for the evaluation of NSN. For English, the data splits into 13,211 train, 1,667 dev, and 2.155 test (testA, semantic parsing) samples. Despite being labeled open domain, WebNLG+2020 is a closed schema, with content drawn from DBpedia across a fixed set of categories such as Astronaut, Airport, Monument, etc. Knowledge Graph generators are scored using WebNLG's Challenge evaluation scripts that compute Precision, Recall, and F1 by searching for the optimal alignment between candidate and reference triples through all possible permutations of hypothesis-reference pairs. Its scale makes it a standard benchmark for training and evaluating end-to-end neural-network knowledge-graph generators.

### 4.3 Evaluation Methodology

The methodology for evaluating the NSN (Figure 5) is grouped into six tasks performed in sequence: 1) Fetch Benchmark, 2) Preprocess Benchmark, 3) Configure Symbolic Reasoning Module; 4) Train Neuro-Symbolic Network, 5) Evaluate NSN on TestA, and 6) Run Scoring Script on NSN Predictions.

#### 4.3.1 Collect Benchmark

This task downloads the WebNLG+2020 data and the scoring script to disk by cloning the GitHub repository at github.com/WebNLG/challenge-2020, which contains raw XML files organized into categories. It contains baselines, evaluation scripts, and results for every submitted system organized by task (text2rdf vs rdf2text) and evaluation type (automatic evaluation/text-to-triples). For scoring scripts, clone github.com/WebNLG/WebNLG-Text-to-triples, which is the automatic evaluation script for the text-to-RDF task.

#### 4.3.2 Preprocess Benchmark

Preprocessing turns the raw data into 1) the input the NSN expects for training and development and 2) the gold-reference file into the format the official evaluation script requires. In the preprocessing subtask (Figure 6), the WebNLG's XML format is converted into NSN's input format, and the gold-reference file is formatted for the evaluation script. This subtask is performed in the following six steps:

1) *Parse the XML file*. Parse the entries in the cloned raw XML files, train (13,211), dev (1,667), and testA (2,155) splits, using Python's lxml.etree to generate parse entries formatted as (eid, category, text, triple-set).

2) *Structure into text-triple pairs.*
Load parsed entries into three pandas.Dataframe, then explode them on text so that each text entry mapped to multiple triples (labels) is replicated to have only one triple.

3a) *Clean and normalize entity and relation strings*. WebNLG data often encodes semantic triples with specific string artifacts that require systematic preprocessing for knowledge graph tasks. Major cleaning efforts involve removing quotes enclosing literals, replacing underscores with spaces, splitting concatenated words, and squashing capitalized letters in camel notation. Clean up accidental double spaces and trailing whitespace and apply the unicode.normalize('NFKC', text) method on free text since crowdsourced text sometimes contains inconsistent Unicode encodings for accented characters.

3b) *Separate seen from unseen categories.* Separate the entries into 16 seen vs 3 unseen categories and exclude the unseen categories from training the NSN. It checks each row's category value against the fixed 3-category unseen set (Film Scientist, MusicalWork) and derives a new column containing category tags {eid, category, category_type, split} that flows to Export NSN Input. It outputs raw triples {eid, triples} that carry the original DBpedia formatting to Export gold reference file. 4a)

*Export NSN input*. Using the row identity (eid) cleaned text is merged with category tags to produce testA, train and dev json files. 4b) *Export Gold References File:* The WebNLG-Text-to-triples script uses *Format spec* to serialize *raw triples* on disk as "one canonical triple set per eid" as expected by the official scorer.

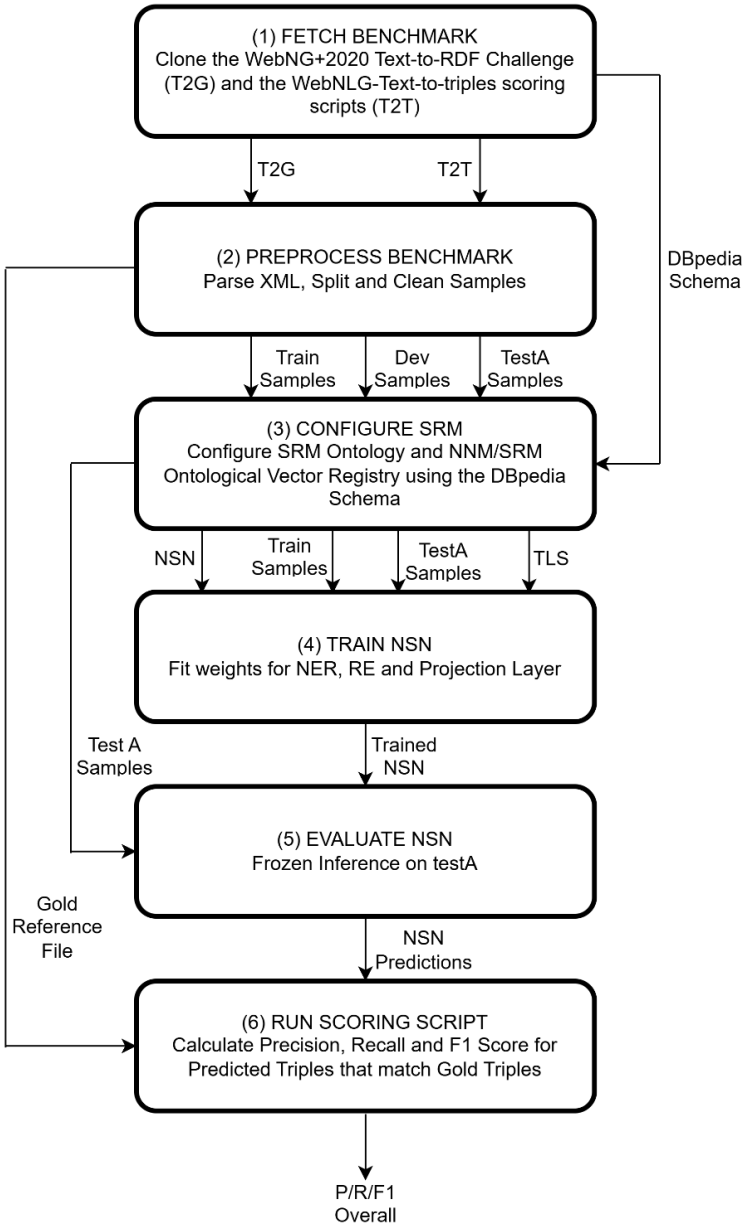


**Figure 5:** Experimental Methodology

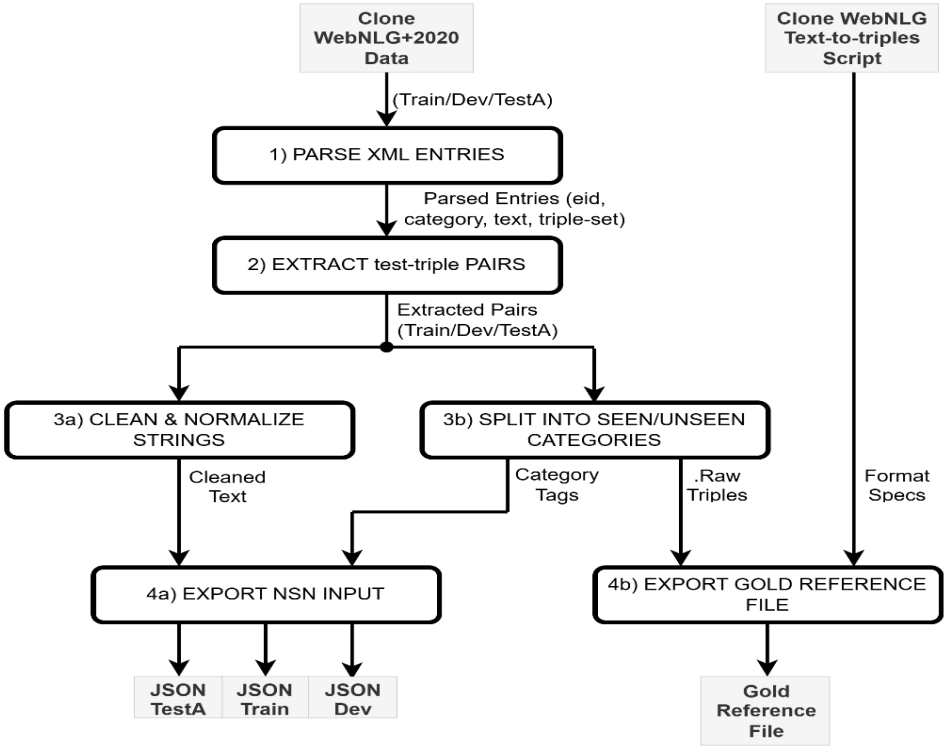


**Figure 6:** Preprocess Benchmark Subtasks

#### 4.3.3 Configure NSN, SRM Ontology, and Ontological Vector Registry

This task consists of the following five subtasks:

1) *Construct NSN*: This subtask constructs the NSN framework for later configuration with objects and functions by subtasks 3.2 and 3.3, and tasks 4 and 5. Subtask 3.2 configures SRM-1 with an ontology and DBpedia.Spotlight for ontology-grounded disambiguation; Subtask 3.3 configures the ontological vector registry with the FAISS index and the URI lookup; Task 4 configures the NER and RE classification heads, and the NNM/SRM projection layer during training; Task 5 configures NNM's tokenizer and embedder while SRM-3, SRM-4 and SRM-5 are plain operations performed on data as it flows through.

2) *Configure SRM ontology*: The ontology and vector registry are drawn from DBpedia schema resources. SRM uses the ontology class hierarchy to constrain candidate entity disambiguation. Configuring SRM-1 with the DBpedia schema takes 4 steps: **a**) Fetch and parse the DBpedia ontology OWL file with **owlready2**, then **b**) map WebNLG's 19 categories onto specific **dbo:** classes. **c**) Configure DBpedia Spotlight with the three parameters that SRM-1 needs for ontology-grounded disambiguation: confidence threshold (minimum similarity score to accept a candidate); support threshold (minimum number of DBpedia links/references candidates must have to filter out noisy matches); and a types filter and policy combination to ensure Spotlight returns candidates of the correct type. d) Add the configured DBpedia Spotlight to the spaCy pipeline.

3) *Configure the NNM/SRM Ontological Vector Registry*: This takes 4 steps: a) Collect a seed entity set restricted to instances of the 19 dbo: classes that the WebNLG's categories directly map onto. Collecting this entity set via SPARQL against the DBpedia endpoint keeps the registry's scope aligned with what SRM-1 will need for disambiguation; **b**) Each seed entity does a set of random walks over the DBpedia graph, which are converted into embeddings using Word2Vec. For a given depth, a seed entity can reach its immediate ontological neighborhood through a finite set of relations such that the embeddings encode not just the seed entity but its neighbors. This makes nearest-neighbor retrieval useful for disambiguation. **c**) To build the FAISS index over normalized vectors, cosine similarity is implemented using the IndexFlatIP on L2-normalized vectors. d) The FAISS index only returns integer positions, so a JSON lookup

table mapping each position back to its DBpedia URI persists the index-to-URI lookup and lets the registry's output be usable by SRM-1.

4) *Derive the Training Label Space (TLS) constrained by SRM Ontology*: Collect every entity type and DBpedia relation appearing in the train split's triples, filtered against the SRM ontology. This defines the fixed output vocabulary the NER and relation extraction heads will be trained to predict.

5) *Verify unseen categories are excluded from train.json and dev.json*: Reads the train.json and dev.json files and scans for unseen categories in the data. If unseen categories are detected, execution halts immediately with the offending category names printed, and Task 4 will never run.

#### 4.3.4 Train Neuro-Symbolic Network

NSN is trained in the following five steps:

1) *Fine-tune the NER head on train samples and validate on dev samples.* Run supervised token-classification training, and at each epoch, compute the training loss on train samples and the validation loss/F1 on dev samples. Save the configuration with the best dev score, not necessarily the one with the lowest training loss, which would only reward memorization.

2) *Fine-tune the relation-extraction head on train samples and validate on dev samples.* Run supervised span-pair classification on triples (subject, relation, object) from train samples, with dev samples used to pick the best checkpoint and decide when to stop training.

3) *Train the interface projection layer on train samples and validate on dev samples*. Fit the contrastive alignment loss of BERT entity embedding into the DBpedia vector-registry space against training samples entity-mention-to-gold-URI pairs and use dev loss to select the best-performing configuration.

4) *Select hyperparameters using dev samples*. Learning rate, batch size, number of epochs, and thresholds such as the FAISS top-k retrieved during interface lookup and the DBpedia Spotlight confidence cutoff should all be tuned by observing dev-split performance. 5) *Freeze all trained weights*. After step 4 lock in the configuration and terminate further parameter updates.

#### 4.3.5 Evaluate Neuro-Symbolic Network

testA.json is run once on the frozen NSN model. This evaluation is pure inferencing – feed each test input text through the full NSN pipeline and output the predicted triples in the flat subject | predicate | object format that the scoring scripts of WebNLG expect. All 2,155 samples are fed through the trained pipeline, and the resulting predictions are scored against the official leaderboard. Because testA uniquely contains the 3 unseen categories (Film, Scientist, MusicalWork), the evaluation pass also serves as a test for how well NSN can generalize.

#### 4.3.6 Run Scoring Script

The scoring script will compare candidate triples to the reference triples, then it calculates Precision, Recall, and F1 score across four distinct matching strictness levels:

1) Strict (S) requires an exact string match for each triple element with correct positional role (subject | predicate | object). Exact (E) requires an exact string match, without positional role; Partial (P) requires only partial string overlap without positional role; and Type (T) requires only partial string overlap with correct positional role.

## 5 Results and Discussion

A comparative analysis is conducted on three approaches for KG extraction from unstructured text: purely symbolic approaches (without neural extraction and reasoning), purely neural approaches (without symbolic reasoning), and neuro-symbolic approaches. Table 3 lists five benchmarks used for these comparisons.

1) *WiRe57 Open-Domain Benchmark* [73]

This is a small hand-curated reference set of 343 relational facts extracted from 57 sentences across five documents (three Wikipedia and two newswire articles). The defining feature of WiRe57 is annotation rigor, where the authors resolve coreference ambiguity and predicate granularity by hand rather than through automatic alignment. It uses token-level precision/recall scoring, where precision is the proportion of extracted words found in the reference and recall is the proportion of reference words found in the system's predictions. It is suited for evaluating rule-based and symbolic systems that need no training, and its small size makes it unsuitable for training or fine-tuning neural models.

*2) WebNLG+2020 Open-Domain Text-to-RDF Benchmark* [63]

This is a large crowdsourced shared task dataset of 42,045 texts (35,426 training, 4,464 dev, 2,155 test) paired with 15,630 RDF triple sets (13,211 training, 1,667 dev, 752 test). Its content is drawn from DBpedia across a fixed set of categories; it has a closed schema even though its surface text style is unconstrained and therefore cannot be considered open domain. It scores predictions using the WebNLG 2020 Challenge scoring script that computes Precision, Recall, and F1 by searching for the optimal alignment between candidate and reference triples through all possible permutations of hypothesis-reference pairs. The definitions of precision and recall, unlike in the WiRe57 benchmark, are based on the exact match of candidate triples and reference gold triples, and are expressed as follows [63]:

$$P\ (\text{Precision}) = \frac{TP}{TP + FP}$$

$$R\ (\text{Recall}) = \frac{TP}{TP + FN}$$

$$F1\ Score = \frac{2PR}{P + R}$$

where a triple is labeled as true positive TP if it is matched to a gold triple, a false positive FP if it does not match any gold triple, and a false negative FN if it is a gold triple without any matching prediction. Its scale makes it suitable for training and evaluating sequence-to-sequence KG generation systems.

3) *Wikipedia/TekGen Open-Domain Text-to-RDF Benchmark* [74]

TekGen is a large-scale parallel text-graph dataset built by aligning Wikidata KG to Wikipedia text and generated using distant supervision. It has 1 million aligned triple-sentence pairs covering 663 Wikidata relations, spanning 1,041 Wikidata properties in total across roughly 6 million (graph, text) pairs. It is built by automated alignment rather than by manual annotation; therefore, its quality is heuristic-dependent. The alignment heuristics assess whether the object appears in the sentence but do not verify whether predicates in the graph match appropriately with the sentence. This is the closest of all five benchmarks to true open-domain, general-knowledge text.

4) *Cyber Threat Intelligence Benchmarks* [48]

The Cyber Threat Intelligence (CTI) benchmark is a unified evaluation set of 219 CTI articles taken from five sources: GRID, CASIE, CTINexus, MalKG, and SecureNLP. Each source has different threat reporting conventions and formats, a feature that makes CTI distinctly heterogeneous and makes the benchmark metrics an average across sources. Although CTI is domain-focused, it tests for cross-source generalization.

**Table 3: Pure Symbolic OIE Systems on the WiRe57 Open-Domain Benchmarks**

| Benchmark | Domain scope | Scale | Construction method |
|---|---|---|---|
| **WiRe57** [73] | True open domain (Wikipedia + newswire) | Very small — 57 sentences, 343 tuples | Manual, hand-curated reference |
| **WebNLG+2020** [63] | Closed-schema, open surface style (DBpedia categories) | Large — 42,045 texts, 15,630 triple sets | Crowdsourced text generation from fixed DBpedia triples |
| **Wikipedia / TekGen** [74] | True open domain (general Wikipedia knowledge) | Very large — 16M aligned triple-sentence pairs (full corpus) 13,474-sentence ontology subset (Text2KGBench) | Automated distant-supervision alignment of Wikidata to Wikipedia sentences |
| **Cyber Threat Intelligence (GRID)** [48] | Domain-focused, multi-source (security text) | 249 articles across 5 sources | Traceable article-graph alignment + scripted task-bank supervision |
| **Materials Science Literature** [54] | Domain-focused (technical scientific text) | Not disclosed in available excerpts | LLM extraction + domain ontological guidance + manual verification |

5) *Materials Science Literature Benchmark* [54] This benchmark evaluates KG extraction from complex materials science literature, comparing hierarchical versus fixed-size text chunking strategies, with results aligned against a curated ground truth dataset through manual verification to ensure semantic correctness. While this collection of text may be described as unstructured, it is domain-focused, unlike the Wikipedia/TekGen benchmark that can be described as truly open domain.

### *5.1 Comparative Analysis of Pure Symbolic Approaches*

Table 4 shows the benchmark results of seven variants of the Open Information Extraction (OIE) system (ReVerb [75], OLLIE [76], ClausIE [77], Stanford OpenIE [78], OpenIE 4 [79], PropS [80], MinIE [81]) on the WiRe57 open domain benchmark [73]. OIE's are purely symbolic systems designed for the automatic extraction of structured relations and entities from open-domain unstructured text without pre-specified target relations or labeled training data.

For the WiRe57 benchmark, the precision and recall of a system's predicted tuples were computed at the token level [73]. Precision is defined as the proportion of extracted words found in the reference, and recall is defined as the proportion of reference words found in the system's predictions [73]. Using these formal definitions of precision, recall, and F1 score, the best-performing OIEs are MinIE, with a recall of 0.323 and an F1 score of 0.358, and Reverb with a precision of 0.569.

### *5.2 Comparative Analysis of Neuro-Symbolic Approaches*

Three neuro-symbolic systems, LOKE-GPT [49], GRID [48], and the Ontology-Guided Hierarchical Extractor [54], are compared on three benchmarks in Table 5: Wikipedia/TekGen [74], Cyber Threat Intelligence [48], and Material Science Literature [54]. LOKE-GPT processor [49] is based on adding a symbolic entity layer on top of the OpenAI text-davinci-003 model using Completion APIs. It investigates the use of GPT models and prompt engineering for knowledge graph construction with the Wikidata knowledge graph. Two processors are compared in [49]: LOKE-GPT, which is the link-corrected version of OKE-GPT, and OKE-GPT, which is uncorrected. As shown in Table 5, LOKE-GPT outperforms OKE-GPT with an F1 score of 0.218 compared to 0.148 for OKE-GPT. McCusker [49] attributes the superior performance of LOKE-GPT to the fact that it is link-corrected. GRID [48] reports two metrics on the CTI benchmark for its twin reinforcement learning KG extractors: an F1 score of 0.685 for its Task Bank reward pipeline and 0.581 for its End2End reward pipeline. The ontology-guided hierarchical KG extractor of Mohammadi et al. [54] returns a Precision of 0.934, a Recall of 0.830, and an F1 score of 0.878 on the Material Science Literature benchmark.

On the very large Wikipedia/TekGen benchmark, both LOKE-GPT and OKE-GPT outperform Open IE4, a pure symbolic KG generator, with 31-fold and 21-fold improvements, although with modest F1 scores of 0.218 and 0.148, respectively. Other neuro-symbolic systems report higher F1 scores on smaller domain-focused benchmarks. The performance results of the three neuro-symbolic KG generators demonstrate that while the complementary combination of neural networks and symbolic reasoning pipelines outperforms pure symbolic reasoning systems on most benchmarks, there is still a lot of room for improvement on truly open-domain benchmarks.

### *5.3 Comparative Analysis of Symbolic, Neural Network, and Neuro-Symbolic Approaches*

Melnyk et al. [82] compared four neural networks BT5 [83], CycleGT [84], ReGen [85] and Grapher [82], and two symbolic networks Stanford OpenIE [86] and Amazon AI [87] on the WebNLG+ 2020 Text-to-RDF Challenge [63]. On the WebNLG+2020 challenge, the Stanford Open IE performed poorly with an exact-match F1 score of 0.158 compared to 0.198 on the much smaller WiRe57 benchmark. These results confirm the pattern documented in the literature that symbolic KG generators are not flexible and do not scale. However, Amazon AI (Shanghai), the closest to a pure symbolic pipeline, is ranked fourth in Table 6, and it is the official winner of the WebNLG 2020 challenge with an F1 score of 0.689. With a pipeline like SRM-1 (entity normalization), SRM-2, and SRM-3 (triple generation) in NSN.

**Table 4: Pure Symbolic OIE Systems on the WiRe57 Open-Domain Benchmarks**

| System | Year | Extractions | Precision | Recall | F1 Score |
|---|---|---|---|---|---|
| ReVerb | 2011 | 79 | **0.569** | 0.121 | 0.200 |
| OLLIE | 2012 | 145 | 0.347 | 0.175 | 0.239 |
| ClausIE | 2013 | 223 | 0.401 | 0.298 | 0.342 |
| Stanford OpenIE | 2015 | 371 | 0.210 | 0.188 | 0.198 |
| Open IE4 | 2016 | 101 | 0.501 | 0.182 | 0.267 |
| PropS | 2016 | 184 | 0.222 | 0.162 | 0.187 |
| MinIE | 2017 | 252 | 0.400 | **0.323** | **0.358** |

**Table 5: Neuro-Symbolic Systems on Open-Domain Unstructured Text Benchmarks**

| System | Year | Text Domain | Precision | Recall | F1 Score |
|---|---|---|---|---|---|
| Open IE4 | 2016 | Wikipedia/TekGen (Open Domain) Text-to-RDF | 0.005 | 0.009 | 0.007 |
| OKE-GPT | 2023 | Wikipedia/TekGen (Open Domain) Text-to-RDF | 0.101 | 0.280 | 0.148 |
| LOKE-GPT | 2023 | Wikipedia/TekGen (Open Domain) Text-to-RDF | 0.248 | 0.195 | 0.218 |
| GRID – Task-Bank Reward (TBR) | 2026 | Cyber Threat Intelligence | 0.846 | 0.649 | 0.685 |
| GRID – End2End Reward (E2ER) | 2026 | Cyber Threat Intelligence | 0.769 | 0.539 | 0.581 |
| Ontology-Guided Hierarchical Extraction | 2026 | Materials Science Literature | **0.934** | **0.830** | **0.878** |

**Table 6: Pure Symbolic OIE vs Pure Neural Network on the WebNLG+2020, Open-domain Text-to-RDF Benchmarks**

| System | Year | Method | Precision | Recall | F1 Score |
|---|---|---|---|---|---|
| Stanford OpenIE | 2015 | Lexico-Syntactic Patterns | 0.154 | 0.164 | 0.158 |
| Amazon AI (Shanghai) [87] | 2020 | Symbolic (Heuristic Entity Linking + DBpedia Database Query) | 0.689 | 0.690 | 0.689 |
| BT5 (T5 – Linearized Graph Generation) [83] | 2020 | Neural Network | 0.670 | 0.701 | 0.682 |
| CycleGT [84] | 2020 | Neural Network | 0.338 | 0.349 | 0.342 |
| ReGen (T5 + RL) | 2021 | Neural Network | 0.714 | 0.738 | 0.723 |
| Grapher (T5 – 770M) | 2022 | Neural Network | 0.715 | 0.733 | 0.722 |
| Proposed NSN | 2026 | Neuro-Symbolic Network | **0.822** | **0.824** | **0.823** |

It outperforms BT5, a fine-tuned T5 system, and CycleGT, a neural network without a pretrained backbone. Amazon AI is described as a heuristic-based approach that does entity linking to match entities in the input text with the DBpedia ontology, then queries the DBpedia database to extract the relationship between them.

CycleGT [84] is an unsupervised method for text-to-graph and graph-to-text generation that places third in the WebNLG 2020 Challenge and sixth in Table 6. The KG construction combines an off-the-shelf entity extractor that identifies all the entities in the input text and a multi-label classifier trained via unsupervised cycle-consistency loss between text-to-graph and graph-to-text directions that predicts the relation

between pairs of entities. The absence of a large-scale pretrained model backbone is the most plausible explanation for its low F1 score of 0.342.

All the other neural network models, BT5, ReGen, and Grapher, use the T5 pretrained model [88] as their backbone. T5 is an encoder-decoder pretrained transformer that consists of self-attention, feedforward, and cross-attention blocks. The final decoder block's output feeds into a dense layer with a softmax. BT5 utilizes a large pretrained T5 model to generate KG in linearized form, where the object-predicate-subject triples are concatenated, and the entire text-to-graph problem is viewed as sequence-to-sequence modeling. Grapher is a T5 pre-trained language model with text-based node generation and a classification edge head. ReGen also leverages the T5 pretrained model and reinforcement learning for bidirectional graph-to-text and text-to-graph generation.

The two neural sequence-to-sequence and transformer-based models outperform the symbolic Amazon AI system. Overall, ReGen has an F1 score of 0.723, a marginal improvement over Grapher's 0.722. Both neural network-based KG generators outperform the Stanford Open IE by a factor of 4.5 and Amazon AI by a factor of 1.05. Although the purely neural approaches, Grapher and ReGen, worked well, they lacked the symbolic reasoning module's logical consistency. The quantity and complexity of unstructured text presented challenges for the purely symbolic Amazon AI, despite its logical soundness. By fusing rule-based reasoning with data-driven insights, the NSN neuro-symbolic approach achieved the greatest F1 score of 0.823, outperforming the symbolic Amazon AI by a factor of 1.19 and the Grapher neural network by a factor of 1.14.

T5, not BERT, is the backbone of all three systems (BT5, Grapher, ReGen) in the top five positions in Table 6. This is not incidental, as there are no systems in the published leaderboard of WebNLG+2020 that have BERT as their backbone. The reason is that BERT is an encoder and does not have the mechanism to generate variable-length output sequences like T5, which is an encoder-decoder and can directly generate linearized graph strings. A separate mechanism must be connected to BERT that converts its outputs – NER and relation-extraction span-pair classifications – for variable-length output sequence generation. BT5, Grapher, and ReGen all depend on T5's ability to generate linearized graph strings. For NSN, it utilizes BERT for NNM and not T5; then, for graph assembly, it utilizes the symbolic triple-generation and inference stages of SRM. The results obtained clearly highlight the advantages of NSN's neuro-symbolic integration.

### *5.4 Error Analysis and Limitations*

Even with the system's effectiveness, there are still certain issues with the proposed NSN: 1) *Entities with overlapping roles*: Occasionally, the relation extraction approach had trouble with complicated or unclear relationships, especially when several entities had overlapping or unclear roles. Furthermore, some entities (such as less well-known names or acronyms) were not correctly represented by the ontology; hence, entity normalization was not always flawless. 2) *Ambiguity in Relationships:* Due to unclear wording in the text, the relation extraction model occasionally made inaccurate relationship predictions. For instance, because there is no context, the model may incorrectly identify a relationship that "works with" as "works for". 3) *Entity Normalization:* Occasionally, entities with similar names were not resolved by the symbolic module, resulting in duplicate entries in the knowledge graph.

### *5.5. Discussion on Scalability and Real-World Applications*

The performance of the neuro-symbolic approach was found to be much more accurate and complete than that of purely neural or symbolic approaches. Moreover, the explainability of the system was improved by the introduction of symbolic reasoning, which is key for regulated sectors such as finance and healthcare. The influence of the comprehensiveness of ontologies used for symbolic reasoning and the quality of the underlying neural models on the performance of the system was significant.

Scalability remains a key problem in applications. The performance of the existing system depends on the computational resources available for training large language models, even with its ability to process massive datasets. For future work, ways to enhance the scalability of the system will be explored, such as the combination of more efficient symbolic

reasoning algorithms and efficient neural networks.

## 6. Conclusions

Using the advantages of both neural networks and symbolic AI, a neuro-symbolic method IS proposed in this research for creating knowledge graphs from unstructured text. While the symbolic reasoning module guaranteed the logical integrity and completeness of the created knowledge graph, the neural network module demonstrated efficacy in extracting entities and relationships from a variety of text sources, attaining high precision and recall. Experiments indicated that KG construction can be automated using the neuro-symbolic technique, as it yields better results in terms of accuracy and coverage compared to just neural or symbolic approaches.

Two of the more significant achievements of the work are the dynamic availability of the data sources and the integration of logical inference with data-driven approaches, where the system is able to cope with the complexity of natural language while being interpretable. The results also highlighted some drawbacks, such as the lack of clarity in relation extraction and entity normalization issues. In spite of these challenges, the neuro-symbolic approach proves to be a valuable method for the creation of KGs, and it shows great promise for practical applications, such as in the medical field, information retrieval systems, and decision-making systems for AI.

The scalability of the system will be improved in the future, lightweight neural architectures will be considered, the ontology will be expanded to include additional complex and diverse entity types. Further, more complex contextual models can enhance the system's ability to handle uncertain interactions.

## 7. Recommendations

Based on the results of the study, the following suggestions are given to advance and utilize neuro-symbolic systems to build knowledge graphs: 1) *Improving Relation Extraction:* To increase relation extraction accuracy, particularly when dealing with ambiguous or overlapping relationships, future iterations of the model should include more sophisticated context-aware techniques, such as GPT-based methods. 2) *Improving Entity Normalization:* Errors in entity normalization could be reduced by implementing more sophisticated ontologies or dynamic entity linking techniques when dealing with uncommon entities or entity names that have multiple versions. 3) *Optimizing Scalability:* Optimizing both the neural and symbolic components to enable more effective processing of huge datasets should be the main goal in order to make this system more suitable for large-scale industrial applications. To decrease computational overhead, methods such as knowledge distillation and model pruning can be considered. 4) *Application to Domain-Specific Use Cases:* This method could be adapted for particular sectors, like healthcare or finance, where domain-specific ontologies and logical rules can improve the precision and applicability of knowledge graphs. Collaboration with industry partners would assist the adaptation of the system to meet practical needs.

## References


[01] Noy, N.F., Gao, Y., Jain, A., et al., 2019. Industry-Scale Knowledge Graphs: Lessons & Challenges. *Communications of the ACM*, 62(8), 36-43. https://doi.org/10.1145/3331166

[02] Cheng, M., Wang, J., Li, S., & Zhang, H. (2021). Neural-Symbolic Methods for Knowledge Graph Reasoning. *eScholarship*.

[03] Hitzler, P., Betz, A., & Bock, P., 2022. Neuro-Symbolic AI: The Path Forward. *Frontiers in AI*, 5, 1-7.

[04] Devlin, J., Chang, M.W., Lee, K., & Toutanova, K. (2019). BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*, 4171–4186.

[05] Garcez, A.d'A., Lamb, L.C., & Gabbay, D.M., (2019). Neural-Symbolic Cognitive Reasoning. *Springer*.

[06] Arrieta, A.B., Díaz-Rodríguez, N., Del Ser, J., et al. (2020). Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities & Challenges Toward Responsible AI. *Information Fusion*, 58, 82–115.

[07] Garcez, A. d'A., & Lamb, L. C. (2023). Neurosymbolic AI: The 3rd wave. *Artificial Intelligence Review*, 56(11), 12387–12406. https://doi.org/10.1007/s10462-023-10448-w

[08] Futia, G., & Vetrò, A. (2020). On the Integration of Knowledge Graphs into Deep Learning Models for a More Comprehensible AI—Three Challenges for Future Research. *Information*, 11(122), 1–10. https://doi.org/10.3390/info11020122

[09] Colelough, B. C., & Regli, W. (2025). Neuro-symbolic AI in 2024: A Systematic Review. *arXiv preprint arXiv:2501.05435*. https://arxiv.org/abs/2501.05435

[10] Xu, J., Zhang, Z., Friedman, T., Liang, Y., & Van den Broeck, G. (2018). A Semantic Loss Function for Deep Learning with Symbolic Knowledge. *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, 5502–5511. https://arxiv.org/abs/2411.04383

[11] Hamilton, K., Nayak, A., Božić, B., & Longo, L. (2024). Is Neuro-Symbolic AI Meeting Its Promises in Natural Language Processing? A Structured Review. *Semantic Web*, 1–43. https://doi.org/10.3233/SW-223228

[12] Wan, Z., Liu, C. K., Yang, H., Li, C., You, H., Fu, Y., ... & Raychowdhury, A. (2024). Towards Cognitive AI Systems: A Survey and Prospective on Neuro-Symbolic AI. *arXiv preprint arXiv:2401.01040*. https://doi.org/10.48550/arXiv.2401.01040

[13] Wan, Z., Liu, C. K., Yang, H., Raj, R., Li, C., You, H., ... & Raychowdhury, A. (2024). Towards Efficient Neuro-Symbolic AI: From Workload Characterization to Hardware Architecture. *IEEE Transactions on Circuits and Systems for Artificial Intelligence*, 1(1), 53-68. https://arxiv.org/abs/2409.13153

[14] Sarker, M. K., Zhou, L., Eberhart, A., & Hitzler, P. (2021). Neuro-Symbolic Artificial Intelligence: Current Trends. *AI Communications*, 34(3), 197–209. https://doi.org/10.3233/AIC-210084

[15] Alkhouri, I., Tavabi, L. A., & Shakarian, P. (2025). A Comprehensive Review of Neuro-Symbolic AI for Robustness, Uncertainty Quantification, and Intervenability. *Arabian Journal for Science and Engineering*. https://doi.org/10.1007/s13369-025-10887-3

[16] Singhal, A., 2012. Introducing the Knowledge Graph: Things, Not Strings. *Google Blog*.

[17] Paulheim, H. (2017). Knowledge Graph Refinement: A Survey of Approaches and Evaluation Methods. *Semantic Web* 8, 3 (2017), 489–508. URL http://ub-madoc. bib. uni-mannheim. de/41515.

[18] Besold, T., Garcez, A., & Lamb, L., (2017). Neural-Symbolic Learning & Reasoning: Contributions & Challenges. In *Proceedings of AAAI Spring Symposium*, 1–5. https://doi.org/10.13140/2.1.1779.4243

[19] Kautz, H. (2022) The Third AI Summer: AAAI Robert S. Engelmore Memorial Lecture, *AI Magazine* 43(1), 105-125. http://doi.org/10.1002/aaai.12036

[20] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*. https://doi.org/10.48550/arXiv.1301.3781

[21] Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* 1532-1543

[22] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, 33, 1877-1901. https://doi.org/10.48550/arXiv.2005.14165

[23] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Hassabis, D. (2016). Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529(7587), 484–489.

[24] Chen, X., Liang, C., Yu, A. W., Song, D., & Zhou, D. (2020). Compositional Generalization via Neural-Symbolic Stack Machines. *Advances in Neural Information Processing Systems*, 33, 1690-1701. https://dl.acm.org/doi/10.5555/3495724.3495867

[25] Dang-Nhu, R. (2020). Plans: Neuro-Symbolic Program Learning from Videos. *Advances in Neural Information Processing Systems*, 33, 22445-22455.

[26] Gupta, T., & Kembhavi, A. (2023). Visual Programming: Compositional Visual Reasoning without Training. In *Proceedings of the IEEE/CVF Conference on Computer*

*Vision and Pattern Recognition* 14953–14962.

[27] Shen, Y., Song, K., Tan, X., Li, D., Lu, W., & Zhuang, Y. (2023). Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36, 38154-38180.

[28] Surís, D., Menon, S., & Vondrick, C. (2023). Vipergpt: Visual Inference via Python Execution for Reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 11888-11898.

[29] Hersche, M., Zeqiri, M., Benini, L., Sebastian, A., & Rahimi, A. (2023). A Neuro-Vector-Symbolic Architecture for Solving Raven's Progressive Matrices. *Nature Machine Intelligence*, 5(4), 363-375. https://doi.org/10.1038/s42256-023-00630-8

[30] Pryor, C., Dickens, C., Augustine, E., Albalak, A., Wang, W., & Getoor, L. (2022). NeuPSL: Neural Probabilistic Soft Logic. *arXiv preprint arXiv:2205.14268*.

[31] Mao, J., Gan, C., Kohli, P., Tenenbaum, J. B., & Wu, J. (2019). The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences from Natural Supervision. *arXiv preprint arXiv:1904.12584*.

[32] Yang, Z., Ishay, A., & Lee, J. (2023). NeurASP: Embracing Neural Networks into Answer Set Programming. *arXiv preprint arXiv:2307.07700*.

[33] Dai, W. Z., Xu, Q., Yu, Y., & Zhou, Z. H. (2019). Bridging Machine Learning and Logical Reasoning by Abductive Learning. *Advances in Neural Information Processing Systems*, 32.

[34] Yi, K., Wu, J., Gan, C., Torralba, A., Kohli, P., & Tenenbaum, J. (2018). Neural-symbolic VQA: Disentangling Reasoning from Vision and Language Understanding. *Advances in Neural Information Processing Systems*, 31. https://dl.acm.org/doi/10.5555/3326943.3327039

[35] Riegel, R., Gray, A., Luus, F., Khan, N., Makondo, N., Akhalwaya, I. Y., ... & Srivastava, S. (2020). Logical Neural Networks. *arXiv preprint arXiv:2006.13155*.

[36] Lample, G., & Charton, F. (2019). Deep Learning for Symbolic Mathematics. *arXiv preprint arXiv:1912.01412* https://doi.org/10.48550/arXiv.1912.01412

[37] Shakarian, P., Baral, C., Simari, G. I., Xi, B., & Pokala, L. (2023). Neuro Symbolic Learning with Differentiable Inductive Logic Programming. In *Neuro Symbolic Reasoning and Learning* 75-87. Cham: Springer Nature Switzerland.

[38] Smolensky, P., Lee, M., He, X., Yih, W. T., Gao, J., & Deng, L. (2016). Basic Reasoning with Tensor Product Representations. *arXiv preprint arXiv:1601.02745*. https://doi.org/10.48550/arXiv.1601.02745

[39] Serafini, L., & Garcez, A., 2016. Logic Tensor Networks: Deep Learning & Logical Reasoning from Data & Knowledge. In *Proceedings of NIPS*, 1-9.

[40] Hohenecker, P., & Lukasiewicz, T. (2020). Ontology Reasoning with Deep Neural Networks. *Journal of Artificial Intelligence Research*, 68, 503-540. https://doi.org/10.1613/jair.1.11661

[41] Liang, Y., & Van den Broeck, G. (2019). Learning Logistic Circuits. In Proceedings of the AAAI Conference on Artificial Intelligence 33, (1), 4277-4286

[42] Demeester, T., Rocktäschel, T., & Riedel, S. (2016, November). Lifted Rule Injection for Relation Embeddings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 1389–1399.

[43] Dong, H., Mao, J., Lin, T., Wang, C., Li, L., & Zhou, D. (2019). Neural logic machines. *arXiv preprint arXiv:1904.11694*.

[44] Wang, P. W., Donti, P., Wilder, B., & Kolter, Z. (2019). Satnet: Bridging Deep Learning and Logical Reasoning using a Differentiable Satisfiability Solver. In *International Conference on Machine Learning* 6545–6554. PMLR.

[45] Cadenas, R. (2026). Convergent Correctness in Stochastic Code Generation: A Generate-Verify-Repair Architecture for Deterministic Validation of Autonomous AI Coding Agents in Regulated Environments. Available at SSRN 6754899.

[46] Li, L., Wang, W., & Yang, Y. (2023). Logicseg: Parsing Visual Semantics with Neural Logic Learning and Reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4122–

4133. https://doi.org/10.1109/ICCV51070.2023.00381

[47] Graves, A., Wayne, G., & Danihelka, I., 2016. Neural Turing Machines. In *Proceedings of NIPS*, 1-10.

[48] Huang, L., Liu, Z., Shao, F., Ma, S., Zhang, M., Chen, Z., ... & Xiao, X. (2026). GRID: Graph Representation of Intelligence Data for Security Text Knowledge Graph Construction. *arXiv preprint arXiv:2605.16714*.

[49] McCusker, J. (2023). LOKE: Linked Open Knowledge Extraction for Automated Knowledge Graph Construction. *arXiv preprint arXiv:2311.09366* https://doi.org/10.48550/arXiv.2311.09366

[50] Rahdari, B., & Brusilovsky, P., 2019. Building a Knowledge Graph for Recommending Experts. In *Proceedings of the DI2KG 2019*, Anchorage, Alaska, pp.1-8.

[51] Stütz, J. D., Karras, O., Oelen, A., & Auer, S. (2025). A User-Centered Neuro-Symbolic Approach for Knowledge Graph Creation from Text. In *International Knowledge Graph and Semantic Web Conference*, 9-24. Cham: Springer Nature Switzerland.

[52] Loconte, L., Hospedales, T., & Cornelio, C. (2026). Better Later Than Sooner: Neuro-Symbolic Knowledge Graph Construction via Ontology-grounded Post-extraction Correction. *arXiv preprint arXiv:2605.29168*.

[53] Servedio, G., Aghilar, P., Mattiace, A., Carmosino, G., Musicco, F., Conte, G., Anelli, V. W., Tommaso Di Noia, & Donini, F. M. (2026). The EpisTwin: A Knowledge Graph-Grounded Neuro-Symbolic Architecture for Personal AI. *arXiv preprint arXiv:2603.06290*.

[54] Mohammadi, N., Dreger, M., Collarana, D., Eslamibidgoli, M. J., Malek, K., & Eikerling, M. (2026). Knowledge Graph Construction from Materials Science Literature using Large Language Models and Advanced Data Preprocessing. *ChemRxiv* https://doi.org/10.26434/chemrxiv.10001546/v1

[55] Kahneman, D. (2011). Thinking, Fast and Slow. *Macmillan*.

[56] Yu, D., Yang, B., Liu, D., Wang, H., & Pan, S. (2023). A Survey on Neural-Symbolic Learning Systems. *Neural Networks*, 166, 105-126.

[57] DeLong, L. N., Mir, R. F., & Fleuriot, J. D. (2024). Neurosymbolic AI for Reasoning Over Knowledge Graphs: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 36(5), 7822-7842. https://doi.org/10.1109/TNNLS.2024.3420218

[58] Cheng, K., Ahmed, N. K., Rossi, R. A., Willke, T., & Sun, Y. (2025). Neural-Symbolic Methods for Knowledge Graph Reasoning: A Survey. *ACM Transactions on Knowledge Discovery from Data*, 18(9), 1-44. https://doi.org/10.1145/3686806

[59] Dürst, M., & Suignard, M. (2005). Internationalized Resource Identifiers (IRIs) (No. rfc3987).

[60] Sardo, L., & Bianchini, C. (2022). Wikidata: A New Perspective Towards Universal Bibliographic Control. *JLIS: Italian Journal of Library, Archives and Information Science = Rivista Italiana di Biblioteconomia, Archivistica e Scienza Dell'informazione:* 13, 1, 2022, 291-311.

[61] Anyaegbunam, C. F. (2024) Constructing Knowledge Graphs from Unstructured Text using Neuro-Symbolic Computing. MIT Thesis, *Department of Computer Sciences, School of Postgraduate Studies, University of Lagos, Lagos, Nigeria*

[62] Chen, S., Liu, H., & Yang, J. (2023). Exploring Knowledge Graph-based Neural-Symbolic System from an Application Perspective. *arXiv preprint* arXiv:2302.05019.

[63] Ferreira, T. C., Gardent, C., Ilinykh, N., Van Der Lee, C., Mille, S., Moussallem, D., & Shimorina, A. (2020). The 2020 Bilingual, Bi-Directional WebNLG+ Shared Task: Overview and Evaluation Results (WebNLG+ 2020). In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, 55-76. https://aclanthology.org/2020.webnlg-1.7/

[64] Mendes, P. N., Jakob, M., & Bizer, C. (2012). DBpedia: A Multilingual Cross-Domain Knowledge Base (pp. 1813-1817). *European Language Resources Association (ELRA)*.

[65] Rossum, G. V. (2018). The Python Language Reference: Release 3.6. 4. https://docs.python.org/3/reference/index.html

[66] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A. M. (2019). Huggingface's Transformers: State-of-the-Art Natural Language Processing. *arXiv preprint arXiv:1910.03771*. https://doi.org/10.48550/arXiv.1910.03771

[67] Zaratiana, U., Tomeh, N., Holat, P., & Charnois, T. (2024). GLiNER: Generalist Model for Named Entity Recognition using Bidirectional Transformer. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (vol. 1: Long papers) pp. 5364-5376 https://doi.org/10.18653/v1/2024.naacl-long.300

[68] Cabot, P. L. H., & Navigli, R. (2021, November). REBEL: Relation Extraction by End-To-End Language Generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021* pp. 2370-2381. https://doi.org/10.18653/v1/2021.findings-emnlp.204

[69] Steenwinckel, B., Vandewiele, G., Agozzino, T., & Ongenae, F. (2023, May). pyRDF2Vec: A Python Implementation and Extension of RDF2Vec. *In European Semantic Web Conference* (pp. 471-483). Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-33455-9_28

[70] Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P. E., ... & Jégou, H. (2025). The Faiss Library. *IEEE Transactions on Big Data*. https://doi.org/10.1109/TBDATA.2025.3618474

[71] Glimm, B., Horrocks, I., Motik, B., Stoilos, G., & Wang, Z. (2014). HermiT: An OWL 2 Reasoner. *Journal of Automated Reasoning*, 53(3), 245-269. https://doi.org/10.1007/s10817-014-9305-1

[72] Siemer, S. (2019). Exploring the Apache Jena Framework. *Georg-August University*: Göttingen, Germany. https://www.dbis.informatik.uni-goettingen.de/Teaching/Theses/PDF/FPrakt-Siemer-MSc-jun-2019.pdf

[73] Léchelle, W., Gotti, F., & Langlais, P. (2019). WiRe57: A Fine-Grained Benchmark for Open Information Extraction. In *Proceedings of the 13th Linguistic Annotation Workshop*, 6-15. https://arxiv.org/abs/1809.08962

[74] Agarwal, O., Ge, H., Shakeri, S., & Al-Rfou, R. (2021). Knowledge Graph Based Synthetic Corpus Generation for Knowledge-Enhanced Language Model Pre-Training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 3554-3565. https://doi.org/10.18653/v1/2021.naacl-main.278

[75] Fader, A., Soderland, S., & Etzioni, O. (2011). Identifying Relations for Open Information Extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 1535-1545 https://aclanthology.org/D11-1142.pdf

[76] Schmitz, M., Soderland, S., Bart, R., & Etzioni, O. (2012). Open Language Learning for Information Extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (pp. 523-534). https://aclanthology.org/D12-1048.pdf

[77] Del Corro, L., & Gemulla, R. (2013). Clausie: Clause-Based Open Information Extraction. In *Proceedings of the 22nd International Conference on World Wide Web*, 355-366. https://doi.org/10.1145/2488388.2488420

[78] Angeli, G., Premkumar, M. J. J., & Manning, C. D. (2015). Leveraging Linguistic Structure for Open Domain Information Extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* (Vol. 1: Long Papers) 344-354 https://aclanthology.org/P15-1034.pdf

[79] Mausam, M. (2016). Open Information Extraction Systems and Downstream Applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 4074-4077. https://dl.acm.org/doi/abs/10.5555/3061053.3061220

[80] Soderland, S., Gilmer, J., Bart, R., Etzioni, O., & Weld, D. S. (2013). Open Information Extraction to KBP Relations in 3 Hours. In *TAC*. https://tac.nist.gov/publications/2013/participant.papers/UWashington.TAC2013.proceedings.pdf

[81] Gashteovski, K., Gemulla, R., & Del Corro, L. (2017). MinIE: Minimizing Facts in Open Information Extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2630-2640. https://doi.org/10.18653/v1/D17-1278

[82] Melnyk, I., Dognin, P., & Das, P. (2022). Knowledge Graph Generation from Text. In *Findings of the Association for Computational Linguistics*: EMNLP 2022 1610-1622. https://doi.org/10.18653/v1/2022.findings-emnlp.116

[83] Agarwal, O., Kale, M., Ge, H., Shakeri, S., & Al-Rfou, R. (2020). Machine Translation Aided Bilingual Data-to-Text Generation and Semantic Parsing. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, 125-130. https://aclanthology.org/2020.webnlg-1.13/

[84] Guo, Q., Jin, Z., Qiu, X., Zhang, W., Wipf, D., & Zhang, Z. (2020). CycleGT: Unsupervised Graph-To-Text and Text-To-Graph Generation via Cycle Training. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*. 77-88. https://aclanthology.org/2020.webnlg-1.8.pdf

[85] Dognin, P., Padhi, I., Melnyk, I., & Das, P. (2021). ReGen: Reinforcement Learning for Text and Knowledge Base Generation Using Pretrained Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 1084-1099. https://doi.org/10.18653/v1/2021.emnlp-main.83

[86] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., & McClosky, D. (2014, June). The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations* 55-60. https://aclanthology.org/P14-5010.pdf

[87] Guo, Q., Jin, Z., Dai, N., Qiu, X., Xue, X., Wipf, D., & Zhang, Z. (2020). $\mathcal{P}$2: A Plan-and-Pretrain Approach for Knowledge Graph-to-Text Generation. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)* 100-106. https://aclanthology.org/2020.webnlg-1.10/

[88] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the Limits of Transfer Learning with a Unified Text-To-Text Transformer. *Journal of Machine Learning Research*, 21(140), 1-67. http://jmlr.org/papers/v21/20-074.html