SN Computer Science (2024) 5:475
https://doi.org/10.1007/s42979-024-02826-0

ORIGINAL RESEARCH

Fine-Tuned T5 Transformer with LSTM and Spider Monkey Optimizer for Redundancy Reduction in Automatic Question Generation

R. Tharaniya sairaj[1] · **S. R. Balasundaram**[1]

Received: 14 September 2023 / Accepted: 22 March 2024

Abstract
The significance of Automatic Question Generation (AQG) lies in its potential to support educators and streamline assessment processes. Notable improvements in AQG are seen with the use of language models, ranging from LSTMs to Transformers. However, there is a need for improvement in the probabilistic scoring technique employed for next word generation in the target question. In this regard, it is noted that template-based methods offer potential for enhancement, although they may result in the generation of fewer or redundant questions due to the utilization of fixed templates. This research aims to address this gap by proposing a hybrid model that combines the the advantages of template-based and Transformer-based AQG approaches. The template-based LSTM approach is explored to learn adaptable question templates. On the other side, the Transformer model is explored to reduce redundancy in the auto-generated questions. The proposed work finetunes the pipelined T5 Transformer model using the Spider Monkey Optimizer over the LSTM-generated templates. The choice of Spider Monkey Optimizer enhances the selection of the named entity in question tail (tail entity) through dynamic sub-search space division for efficient exploration and exploitation, and self-organization based on local and global scoring. This ensures that the named entity in the question tail is non-redundant (diverse) and regards both structural and contextual coherence to the auto-generated question. Experimental findings highlight improvements in how well the diversely selected named entities are relevant to the generated questions through higher precision, recall and f1-scores in the pipelining phase. Moreover, the study shows that the Spider Monkey Optimizer performs better in selecting tail entities, and it consistently outperforms other algorithms in F1-score and convergence time across all datasets, with its time complexity increasing linearly as dataset size grows. The finetuned pipelined T5 model (proposed model) exhibits improved ROUGE scores over baselines with reduced computational overhead and shorter inference time in the generative phase across datasets in linear convergence time.

Keywords LSTM · Transformer model · Generative fine-tuning · Spider monkey optimizer · Automatic question generation

Introduction

Automatic Question Generation (AQG) is vital in e-Assessment [1, 4, 6, 8, 9, 14] which widely depends on machine learning and deep learning to generate questions automatically from text. It is crucial for improving educational materials, assessment systems and chatbot interactions by generating diverse and contextually relevant questions [2, 3, 5, 10, 12, 13, 15]. In this field, LSTMs and Transformers are the prominent models to automatically transform source text into the target questions. LSTMs excel in generating template-based questions [7, 11, 14, 15, 32, 38] with higher performance score (60–65%) against the evaluation metric such as ROUGE. But limitedness in quantity of generated questions is noted depending on the number of predefined template patterns [15, 23, 32, 38]. In contrast, Transformer models introduce advanced techniques such as bidirectional encoding and fine-tuning-based decoding to auto-generate question template patterns along with greater number of questions. However, evaluating with performance metric such as ROUGE shows lower performance scores (40–45%)

This article is part of the topical collection "Emerging Applications of Data Science for Real-World Problems" guest edited by Satyasai Jagannath Nanda, Rajendra Prasad Yadav and Mukesh Saraswat.

✉ R. Tharaniya sairaj
tharaniyasairaj@gmail.com

S. R. Balasundaram
blsundar@nitt.edu

[1] Department of Computer Applications, National Institute of Technology, Tiruchirappalli, India

Published online: 23 April 2024

due to its dependability to probabilistic next word selection [1, 4, 5, 8, 23, 40].

For example, consider a source text from DBMS subject in computer science. "Atomicity in database management systems (DBMS) ensures that transactions are treated as indivisible units. It guarantees either the entire transaction's success or no changes to the database. This property prevents partial updates and maintains data integrity, ensuring a consistent database state". The possible template-based questions are “Can you demonstrate atomicity in a practical scenario?”, “summarize the main idea of atomicity.”etc. The possible LM based questions are “How does atomicity maintain data integrity in the event of system failures?”, “What does atomicity ensure in a database?” etc. In this regard, the limitations from both approaches (Table 1) must be taken care to improve the overall performance of AQG (Fig. 1).

To improve the strengths and reduce the weaknesses (Table 1) of the individual models, a hybrid model is proposed. This model aims to combine the best aspects of LSTM and Transformer architectures to improve AQG. It focuses on quantifying auto-generated questions and reducing redundancy through entity-guided next-word generation. The hybrid model utilizes a sequential pipeline where LSTM generates structural (dependency parsed) sequences followed by T5 Transformer generating questions. Additionally, the mapping phase implies the application of the modified Spider Monkey Optimizer (SMO) [4, 16, 17, 40] over the LSTM generated structural sequences to select appropriate named entities for the T5 model. In this context, the objective of achieving superior performance in selecting the named entity in question tail (tail entity) is carried out through the application of the Spider Monkey Optimizer (SMO). Tail entity selection is improved by SMO through exploration exploitation, and subdivision operations on the search space, which represents dependency sequences from LSTM-based templates. The approach of local and global scoring of the SMO results in the inclusion of inter-phrasal dependency sequences or exclusion of sub-phrasal dependency sequences (self-organization). This is based on weight updates to enhance tail entity selection. The choice of T5 transformer is inspired from baseline AQG workflows distinguishing it from other models such as BERT, BART, GPT, etc. Its distinctive bidirectional encoding and parametric fine-tuning-based decoding hold potential for generating coherent auto-generated questions [1, 5, 37–39]. Based on these observations, the proposed work models a finetuned T5 pipeline model (Flan-T5-ppl) to advance tail entity selection and entity guided AQG to reduce redundancy in target (auto-generated) questions. The contributions of the proposed work are given below.

1. Integrating strengths of LSTM and Transformer models in AQG to address individual limitations thereby improving the ROUGE score.
2. Addressing redundancy in Transformer models for AQG with tail entity selection based data pipelining.

Table 1 Need for improvement in Auto-generated Questions

Source text	Auto-generated questions	Reasons for improvement
Atomicity in database management systems (DBMS) ensures that transactions are treated as indivisible units. It guarantees either the entire transaction's success or no changes to the database. This property prevents partial updates and maintains data integrity, ensuring a consistent database state	LSTM generated Questions	
	Explain the concept of atomicity in DBMS?	Limited quantity of hand-crafted templates
	Explain the concept of transactions as indivisible units in a DBMS?	Difficulty in handling complex language structures
	Transformer generated Questions	
	How does atomicity maintain data integrity in the event of system failures?	Over-reliance on probabilistic next-word generation
	Why ensuring a consistent database state is important?	Lower n-gram Recall (ROUGE)

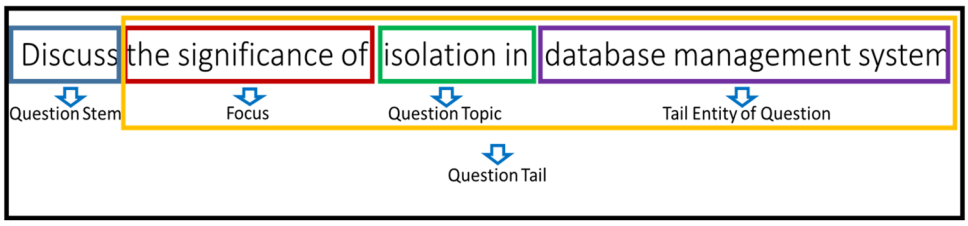

Fig. 1 Structure of an exam Question

3. Applying Spider Monkey Optimizer on LSTM for efficient tail entity selection thereby improving the F1 score and the convergence rate.

The organization of the rest of the paper is given in the forthcoming sections. The related work is given in Sect. "Related Work", focusing on existing techniques and identifying gaps. The methodology is described in Sect. "Experimental Study", encompassing the hybrid model's architecture, integration methods, and evaluation metrics. Experimental outcomes are presented in Sect. "Results and Discussion", comparing the hybrid model with existing approaches. The conclusion in Sect. 6 summarizes key contributions.

Related Work

The growth of AQG systems have been continuous, offering the potential in the assessment process [1–15]. This is achieved by three major approaches, namely, template-based, language model-based, and neural network-based methods. Although the implications of these approaches and the advantages they offer are high, some limitations are also noted. To understand, these limitations, literature on AQG is reviewed.

Existing AQG Approaches

Template-based methods represent an initial stage in AQG. These approaches generate predefined question syntax with placeholders [9, 10, 13, 15, 25, 39]. These placeholders are then computationally replaced with appropriate named entities to generate questions. While template-based systems provided a degree of automation, they are noted with limitations of rigid syntax, limited generative power and requirement of newer templates. Early research works focused on refining the templates for scalability to balance between automation and customization.

Neural network-based methods represent evolution of neural network architectures such as RNN, LSTM, Auto-Encoder, Transformer etc., [28, 33, 35–37] to learn question generation patterns [3, 9, 26–29]. These models can capture n-gram relationships in the source text to generate target questions that are both abundant and not domain specific. Ongoing research in this area focuses on developing specialized neural architectures tailored specifically for exam question generation tasks.

Language model-based approaches, particularly those utilizing large pre-trained Transformer models such as BART, T5 etc., have brought significant advancements to automatic question generation [1, 10, 15, 28]. These models can generate coherent and abundant questions based on the structure of given data and the parameters used. Recent research has concentrated on fine-tuning these models for educational content through effective exploitation of their generative capabilities. In addition, most language model-based approaches have shown promise in maintaining consistency of the generative output based on data fine-tuning.

Gaps and Limitations in Prior Research

While template-based, language model-based, and neural network-based approaches have made significant contributions in automatic question generation, several challenges persist. Template-based methods may struggle with handling complex question structures or generating newer question types [9, 12, 13, 30]. Neural network-based models can face difficulties in dealing with rare or out-of-distribution data. Language model-based approaches, yet powerful, may occasionally generate inaccurate questions with the need for efficient fine-tuning [18, 20, 21, 23, 31]. The current study aims to advance AQG by proposing a hybrid approach that combines template-based and language model-based methods. Based on the gaps, following research questions are addressed in the proposed work.

1. How does the integration of automatic template extraction model reduce the dependence on probabilistic next-word generation methods in AQG?
2. What is the significance of pipelined T5 Transformer in achieving improved n-gram Recall (ROUGE) and reduced Redundancy over existing AQG approaches?

Methodology

The research aims to improve the relevance of automatically generated questions with target sequence replacement technique. The proposed workflow (Fig. 2) draws insights from natural language processing (NLP), deep learning, and optimization. Here, LSTM captures question structure and the T5 model aids question generation through template alignment. The methodology involves fine-tuning the T5 transformer model with LSTM extracted question templates. Named entity mapping [19–22] using the spider monkey algorithm is employed as well towards the improvement of final AQG results.

Bi-LSTM Based Neural Question Template Extraction

The neural Bi-LSTM encoder (E) is used to examine the dependencies between individual words and word

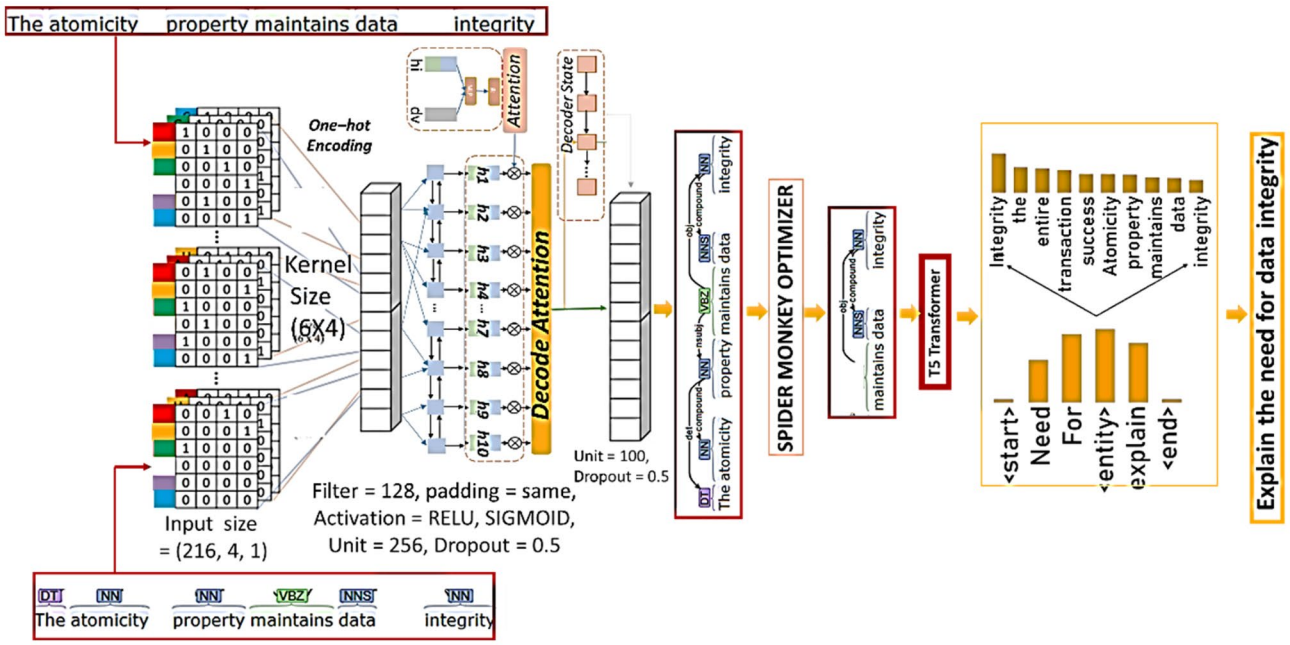

Fig. 2 Hybrid Dependency Parsing with LSTM, Pruning with SMO and Question Generation with T5 Transformer

pairs, predicting the posterior temporal co-occurring token. Every word, represented as 'wi', undergoes a transformation, yielding an equivalent vector form $((vi)^{\rightarrow};(vi)^{\leftarrow})$. Upon this vector representation, a 32-dimensional embedding matrix is formed, encompassing factors like Entity importance, attentive context, and dependencies. This process is guided by POS tags, DEP links, NER tags and parse chunks. While encoding the source text, both forward and backward n-gram hidden states are generated. This entails combining the weight of the present token with the vector from its co-occurring token, minimizing temporal differences. In this context, the kernel size (6×4) used in experiments is pre-determined based on the neural architecture requirements. An mxn matrix is treated as kernel if it focuses on local/global feature extraction from the encoded input data. The kernel size conventionally represents context window in one dimension (m) and possible local feature vector size in other dimension (n). In our experiments, the value m = 6 represents bi-directional tri-gram encoded vector size. It is chosen based on its superiority over existing encoding techniques. The value n = 4 represents the maximum possible dependency relationships in a phrase structure. It is chosen based on demonstrative improvement in semantic encoding across existing systems. The 6×4 kernel matrix is a rectangular kernel which focuses on local feature extraction from the encoded input data. It promotes inter-phrasal dependency sequence exclusion and sub-phrasal dependency sequence inclusion based on weight updates.

The forward vector evolution is shown in Eq. 1:

$$(vi)^{\rightarrow} = \mathrm{LSTM}\,(wi, (vi-1)^{\rightarrow}) \tag{1}$$

Likewise, **Eq.** 2 showcases the backward vector's transformation:

$$(vi)^{\leftarrow} = \mathrm{LSTM}\,(wi, (vi+1)^{\leftarrow}) \tag{2}$$

In both cases, $(vi)^{\rightarrow}$ and $(vi)^{\leftarrow}$ symbolize the forward and backward vectors, reflecting the source text's representational encoding. 'w_i' stands for the word or token within the text. This two-vector approach enriches the understanding of context, considering preceding and subsequent tokens.

Furthermore, the neural Bi-LSTM decoder (D) is implemented to utilize the neural generative capabilities. This decoder is pivotal on meta-sequence patterns (hidden vectors) to acquire convergence on joint probability-based token sequence realization for AQG. It produces extra hidden states that align with two crucial aspects: hierarchical parse attention (h_c) and attention based on key elements (h_a). h_i represents the input layer neurons and each "h_i" corresponds to an individual input feature corresponding to the source vocabulary. dv represents the hidden layer stands for the activation values. h_1 to h_{10} represent individual neurons in the hidden layer, with each node computes based on the inputs received and transmits its output to the next layer. The quantification of hierarchical parse attention emerges through a linear transformation of the preceding hidden layer, striking a

balance between learned vectors and domain-specific joint representations.

The calculation of the cognitive level bias vector is encapsulated by Eq. 3:

$$hc = \max [\tanh (mx * (vi)^{\rightarrow} + b), \tanh (my * (vi)^{\leftarrow} + c)] \quad (3)$$

In this equation, 'hc' represents the hierarchical parse attention vector, while $(vi)^{\rightarrow}$ and $(vi)^{\leftarrow}$ correspond to the forward and backward vectors. The coefficients 'mx', 'my', 'b', and 'c' encapsulate weight matrix constituents of the encoding process.

On temporal basis, a context vector based on attention weights are computed. This vector is shaped by applying exponential transformation with a non-linear activation function (tanh) to the dot product of the prior two hidden layers. Integration of outputs from these layers is facilitated through a maxout pointer, culminating in an input for the final softmax layer. Guided by Eqs. 4 and 5, the softmax layer computes the likelihood of selecting a token from the source sequence towards the process of question template generation.

Equation (4) encapsulates the formation of the attention context vector:

$$ha = \max [(wi, (vi-1)^{\rightarrow} + wi, hc), (wi, (vi+1)^{\leftarrow} + wi, hc)] \quad (4)$$

Additionally, Eq. (5) outlines the computation of the probability of selecting a token from the source sequence:

$$p(wi|\ wi-k,\ wi+k) = \text{softmax}\ (hc * ha) \quad (5)$$

Generative Fine-Tuning Guided T5 Model

The subsequent phase of our methodology involves the enhancement of a T5 transformer model's ability to generate questions.

Algorithm 1: FineTune-GenT5

Input: (context, Source text) pairs "(C,S)".
Output: Generated Question List, Q.

```
Load_model(T)    // T can be T5-base or T5-small
optimizer = AdamW(T5.parameters(), lr=3.0*10-4)    //Initialize the optimizer
//Loading data in batches
data_loader = DataLoader(dataset=(C, S), batch_size=256, shuffle=True)

For each epoch do:
        for batch in data_loader:
                questions = []
                for source,context in batch:
                        (vf) =GenFVec(wi)       // forward vector
                        (vb) =GenBVec (wi)      // backward vector
                        hc = max [tanh(m*vf), tanh(m*vb)] //m is parametric matrix
                        p(wi| wi-k, wi+k) = softmax (hc)
                generated_questions = T5_generate[p(wi| wi-k, wi+k)]
                for question in generated_questions:
                        Q = Q ∪ question
                invoke_optimizer()
                lossB = get_BLoss(T5) //Backward Loss
                lossF = get_FLoss(T5) //Final Loss
                End for
        total_loss = total_loss ∪ lossF
        End for
        T5' = update_weights(T5)
Return T5' , Q
```

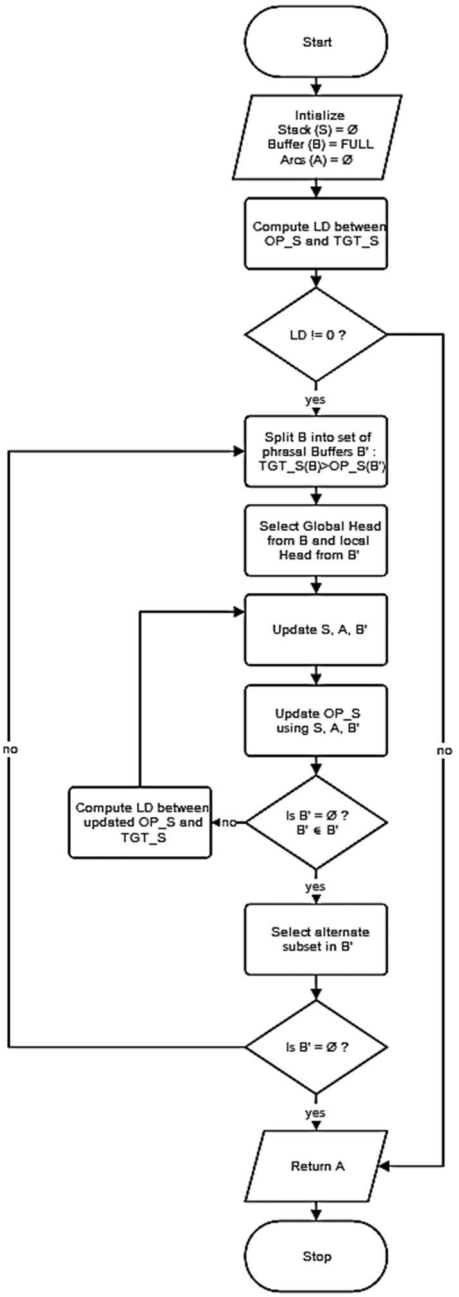

Fig. 3 Flowchart of Spider Monkey optimizer

The T5 Transformer functions as pipelined or end-to-end text transformation architecture which is not ideal to AQG. However, the generative nature of T5 can be exploited with respect to AQG (Algorithm 1). But, the proficiency of T5 model needs to be enhanced through fine-tuning, using domain-specific questions as reference points. This fine-tuning process involves replacement of questions with equivalent question templates as target text. Following this, the T5 model training for question generation is significantly

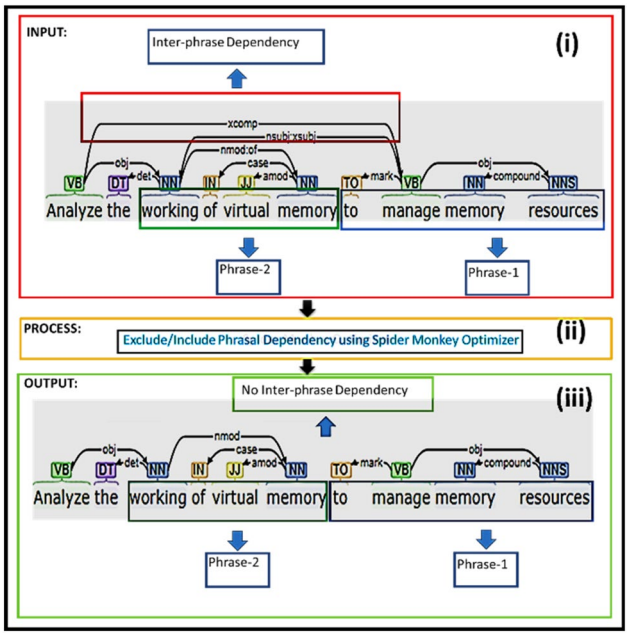

Fig. 4 Exclusion or Inclusion of phrasal Dependency using Spider Monkey Optimizer (i) Input (ii) Process (iii) Output

Table 2 Notations used in Fig. 3

Notations	Explanation
S	Stack having head or tail of dependency arc
A	Set of dependency arcs
B	List of tokens in the sentence
B'	Set of List of tokens in each phrase of the sentence
B''	Unique list in B'
TGT_S	Set of gold standard transitions
OP_S	Set of transitions made from B'' to attain B' = ∅
LD	Minimum edit distance between TGT_S and OP_S

improved. During fine-tuning, the model's learning parameters are adjusted during each iteration (Fig. 3).

SMO driven Entity—Question Template Mapping

Inspired by the behavior of spider monkeys [16–18, 31] in searching for optimal paths, SMO is adapted to find optimal mappings while considering the constraints posed by inter-phrasal dependencies. On analysis, it is noted that the integration of SMO enhances the efficiency and effectiveness of the mapping (Figs. 3 and 4). It is observed to denoise inter-phrasal dependencies in the context, eventually reducing inaccuracies in the entity ranking process. The approach introduces a refined objective to minimize the Levenstein distance in dependency arc attachment (selection) process. The notations used in Fig. 4 are described

in Table 2. This results in the generation of named entity-specific dependency graphs that provide insights into sentence structures. It is followed by ranking of Named Entities, a pivotal step where the relevance of named entities is evaluated. The Composite Lexical Similarity Metric (CLSM) combines the common lexical similarity metrics, providing a comprehensive measure of entity relevance (the suitability of entities for specific question templates).

Experimental Study

Data Sources

Various datasets have been chosen for this study. The Stanford Question Answering Dataset (SQUAD) offers a broad array of context passages and related question–answer pairs. The Blooms Cognitive Level Questions Dataset (BCLs) is used to assess cognitive levels in education and provides question templates categorized by Blooms Taxonomy. The Learning Questions (LearningQ) dataset consists of questions from educators, aiding in evaluating the quality and relevance of generated questions. Lastly, the Science Question (SciQ) dataset, curated by Allen AI, focuses on science-related questions to test the hybrid model's ability in generating analytical and reasoning-based questions.

Experimental Framework and Implementation Details

TensorFlow and the Hugging Face Transformers library are employed for preprocessing the SQuAD dataset, gearing it towards training a Transformer model. The initial steps involve loading a pre-trained tokenizer and defining the dataset structure. The subsequent preprocessing function takes charge of tokenization and encoding, accommodating both context and question pairs along with their corresponding answers. This process includes potential truncation of sequences and subsequent padding to ensure uniform lengths. The resultant preprocessed dataset is then organized and stored, setting the stage for the subsequent training of a Transformer model using TensorFlow. The reported inference time is based on the base configuration of the system given in Table 3.

Table 3 Minimum System Configuration

Resource parameters	Parametric values
CPU	Intel® Core™ i7-3770 CPU@ 3.40 GHz
GPU	NVIDIA Tesla K80
Base Speed	3.40 GHz
Cores	4
Logical Processors	8
Memory	8.0 GB DDR4

Table 4 Choice of Parameters for the Optimization Algorithms

Algorithms	Choice of Parameters	Values
Spider Monkey Optimizer (SMO)	Population Size	100
	Exploration Rate	0.3
	Exploitation Rate	0.5
	Mutation Rate	0.05
Ant Colony Optimization (ACO)	Number of Ants	100
	Alpha (pheromone)	2
	Beta (heuristic)	3
	Pheromone Decay Rate	0.5
Artificial Bee Colony (ABC)	Colony Size	100
	Maximum Trials	20
	Limit	2
	Number of Elite Bees	10
Particle Swarm Optimization (PSO)	Swarm Size	100
	Inertia Weight	0.7
	Cognitive Coefficient	1.5
	Social Coefficient	1.5
Firefly Algorithm (FA)	Population Size	100
	Attractiveness Coefficient	1.5
	Absorption Coefficient	1.0
	Mutation Rate	0.05

Choice of Parameters for Optimization Algorithms

The optimal parameters from the study's findings on optimization algorithms such as the Spider Monkey Optimizer (SMO), Ant Colony Optimizer (ACO), Artificial Bee Colony Optimizer (ABC), Particle Swarm Optimizer (PSO) and Firefly Algorithm (FA), are Tabulated (Table 4).

Generative Modelling Baselines

Implementing the Bi-LSTM based neural question template extraction using libraries such as Numpy, Keras, TensorFlow etc. Hyper-parametric computations are done with batch size, max length, optimizer etc. for efficient model learning. The Hugging Face Transformers library and configurations for encoding data to the T5 model is exploited towards generative output questions. Dependency parsing using libraries such as SpaCy, Textacy etc., and SMO refines the mapping process with maxout objective. Evaluation metrics such as ROUGE scores for language model-based methods, Precision, Recall and F1 metrics for entity mapping performance are utilized.

Table 5 Results of the Proposed Work

Source text	Output questions (Naïve Approach)	Generated templates	Output questions (proposed work)
Armstrong's Axioms are set of rules to discover functional dependencies. Start with a set of known functional dependencies, we apply these rules again and again to find dependencies that are logically implied	What is the role of Armstrong's Axioms to discover functional dependencies?	Explain the significance of < Question_ topic > to < tail_ entity >	Explain the significance of Armstrong's Axioms to discover Trivial FDs
	Explain the significance of Armstrong's Axioms to discover functional dependencies		Explain the significance of transitivity rule to discover functional dependencies
	Write a note on Armstrong's Axioms and its importance to discover functional dependencies	Write a note on < Question_ topic > and its importance to < tail_ entity >	Write a note on Reflexive rule and its importance to Armstrong's Axioms
	How are Armstrong's Axioms used to discover functional dependencies?		Write a note on Armstrong's Axioms and its importance to discover functional dependencies

Table 6 Comparative study on Template Extraction Techniques

Datasets		ROUGE-1	ROUGE-2	ROUGE-3
BCLs (D1)	RNN	63.97	62.66	61.89
	LSTM	62.74	60.68	58.17
	RNN + Attn	60.16	58.79	57.02
	LSTM + Attn	**63.35**	**62.01**	**61.33**
LearningQ (D2)	RNN	64.03	62.72	61.95
	LSTM	62.8	60.74	58.23
	RNN + Attn	60.22	58.85	57.08
	LSTM + Attn	**63.41**	**62.07**	**61.39**
SQUAD (D3)	RNN	63.78	62.47	61.7
	LSTM	62.55	60.5	58
	RNN + Attn	59.98	58.61	56.85
	LSTM + Attn	**63.16**	**61.82**	**61.15**
SciQ (D4)	RNN	63.98	62.67	61.9
	LSTM	62.75	60.69	58.18
	RNN + Attn	60.17	58.8	57.03
	LSTM + Attn	**63.36**	**62.02**	**61.34**

Scores in bold indicate significant improvements of the proposed model over the baselines in terms of different performance metrics

Results and Discussion

Significance of Bi-LSTM Template Generation

The Bi-LSTM model, known for its effectiveness in capturing sequential patterns and dependencies within text data, is utilized to extract question templates that hold importance across various natural language processing tasks. By considering both past and future context, the model effectively captures long-range dependencies and intricate syntactic patterns. This allows it to accurately extract templates from questions involving diverse linguistic constructs such as multiple clauses, nested phrases etc. To train the Bi-LSTM-based question template extraction model, questions spanning across the DBMS subject domain over diverse topics such as Normalization, Concurrency control, Transaction Management etc., are considered. Each question is accompanied by its corresponding question template annotation. The model takes sequences of token as input and is subjected to supervised training, predicting the associated template by analyzing deep linguistic dependencies (Table 5).

When compared to traditional handcrafted templates, the Bi-LSTM-based approach exhibits clear advantages (Tables 5 and 6). While handcrafted templates are often constrained by the imagination and expertise of template designers, the neural model learns templates directly from data, enabling it to adapt to key topic areas and more question structures.

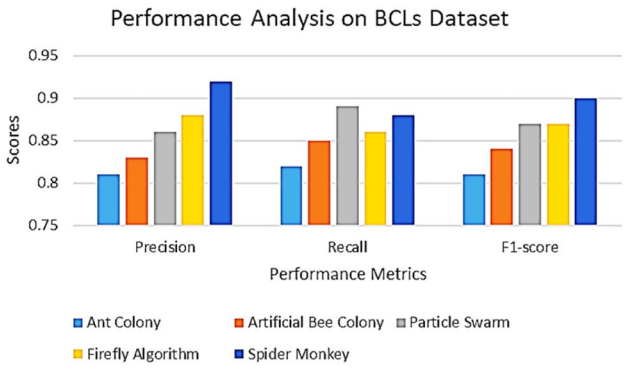

Fig. 5 Performance analysis—BCLs dataset

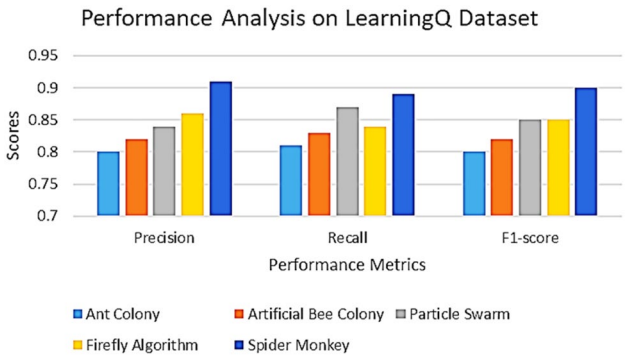

Fig. 6 Performance analysis—LearningQ dataset

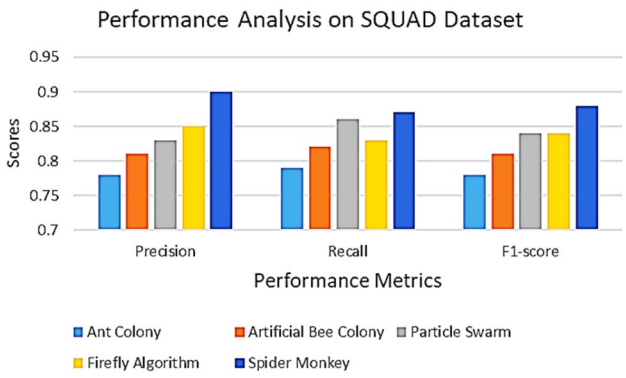

Fig. 7 Performance analysis—SQUAD dataset

The LSTM + Attn model consistently outperforms on template-based models, showing superior capability for semantic sequence analysis and attention mechanism to generate contextually relevant questions. The inference is by the criterion given below.

Performance Consistency: Evaluation criteria reveal consistent superiority of the LSTM + Attn model over RNN, LSTM, and RNN + Attn models across all datasets (D1, D2, D3, D4) and evaluation metrics (ROUGE-1, ROUGE-2, ROUGE-3).

Improvement Over Baselines: LSTM + Attn exhibits enhanced capture of higher-order of similarity between generated and reference questions compared to baseline models such as RNN, LSTM, and RNN + Attn.

Attention Mechanism Benefits: LSTM + Attn's attention mechanisms enable the model to focus on finer blocks of the input sequence which results in the production of more cohesive and contextually relevant questions.

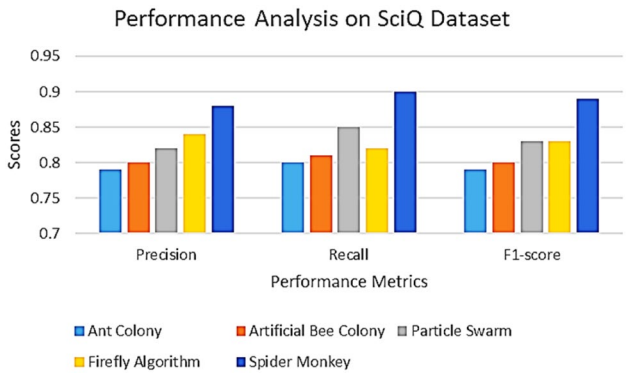

Fig. 8 Performance analysis—SciQ dataset

Data Efficient Fine-Tuning vs Baseline

The comparative study on different nature-inspired algorithms for entity mapping shows various trends in performance with respect to the data efficient fine-tuning process. The Ant Colony algorithm consistently shows the lowest results, especially when dealing with new entities. The Artificial Bee Colony method introduces bias due to four-stack weight estimation making it harder to accurately to filter or prune inter-phrasal dependencies. The Firefly and Particle Swarm algorithms encounter challenge of sub-phrasal dependencies inclusion through weight-based updates. In contrast, the Spider Monkey Optimizer (SMO) (Figs. 5, 6, 7, 8) shows better performance in inter-phrasal dependency exclusion and sub-phrasal dependency inclusion based on weight updates. This leads to improved results, representing the potential of SMO (Table 7) for improving entity mapping.

The empirical findings (Table 7) indicate that the FA and ACO algorithms exhibit a quadratic time complexity (O(n2)) increase with the increase in dataset size. This is primarily due to the quadratic rise in pairwise interactions in semantic representational space. In contrast, the time complexity of the SMO, ABC, and PSO algorithms shows a linear growth (O(n)) with dataset size, leading to consistent performance improvements across all datasets. As the dataset size increases, the operations of the SMO, ABC,

Table 7 Comparative study on Nature-inspired Entity Selection Algorithms

Datasets	Methods	Precision	Recall	F1-score	Inference Time (ms)
BCLs	Ant Colony	0.81	0.82	0.81	54.1 ± 0.63
	Artificial Bee Colony	0.83	0.85	0.84	48.5 ± 0.78
	Particle Swarm	0.86	0.89	0.87	49.7 ± 0.52
	Firefly Algorithm	0.88	0.86	0.87	56.3 ± 0.39
	Spider Monkey (Proposed)	**0.92**	**0.88**	**0.9**	**48.6 ± 0.41**
SciQ	Ant Colony	0.8	0.81	0.8	66.1 ± 0.92
	Artificial Bee Colony	0.82	0.83	0.82	49.3 ± 0.64
	Particle Swarm	0.84	0.87	0.85	48.9 ± 0.85
	Firefly Algorithm	0.86	0.84	0.85	68.2 ± 0.77
	Spider Monkey (Proposed)	**0.91**	**0.89**	**0.9**	**48.8 ± 0.53**
SQUAD	Ant Colony	0.78	0.79	0.78	70.6 ± 0.27
	Artificial Bee Colony	0.81	0.82	0.81	54.1 ± 0.64
	Particle Swarm	0.83	0.86	0.84	57.2 ± 0.52
	Firefly Algorithm	0.85	0.83	0.84	71.7 ± 0.71
	Spider Monkey (Proposed)	**0.9**	**0.87**	**0.88**	**53.4 ± 0.39**
LearningQ	Ant Colony	0.79	0.8	0.79	74.3 ± 0.56
	Artificial Bee Colony	0.8	0.81	0.8	56.8 ± 0.62
	Particle Swarm	0.82	0.85	0.83	60.7 ± 0.75
	Firefly Algorithm	0.84	0.82	0.83	76.7 ± 0.81
	Spider Monkey (Proposed)	**0.88**	**0.9**	**0.89**	**56.9 ± 0.47**

Scores in bold indicate significant improvements of the proposed model over the baselines in terms of different performance metrics

Table 8 Comparative study on Generative AQG models

Datasets	Methods	ROUGE-L	Accuracy	Inference Time (ms)
BCLs	T5-e2e [28, 37]	0.6	0.92	136.43 ± 0.41
	Flan-T5-e2e	0.61	0.94	108.78 ± 0.57
	T5-ppl [39, 39]	0.58	0.91	99.65 ± 0.63
	Flan-T5-ppl (Proposed)	**0.63**	**0.93**	**87.34 ± 0.49**
SciQ	T5-e2e [28, 37]	0.59	0.91	139.84 ± 0.64
	Flan-T5-e2e	0.57	0.92	111.35 ± 0.47
	T5-ppl [39, 39]	0.62	0.89	103.67 ± 0.52
	Flan-T5-ppl (Proposed)	**0.6**	**0.9**	**91.76 ± 0.33**
SQUAD	T5-e2e [28, 37]	0.62	0.85	164.61 ± 0.16
	Flan-T5-e2e	0.59	0.87	134.65 ± 0.56
	T5-ppl [39, 39]	0.58	0.86	123.43 ± 0.72
	Flan-T5-ppl (Proposed)	**0.61**	**0.89**	**112.12 ± 0.47**
LearningQ	T5-e2e [28, 37]	0.63	0.93	170.12 ± 0.28
	Flan-T5-e2e	0.62	0.95	139.89 ± 0.45
	T5-ppl [39, 39]	0.6	0.92	128.89 ± 0.81
	Flan-T5-ppl (Proposed)	**0.64**	**0.94**	**117.71 ± 0.37**

Scores in bold indicate significant improvements of the proposed model over the baselines in terms of different performance metrics

and PSO algorithms operate independently during optimization, resulting in linear time complexity. Although PSO demonstrates linear inference time, it requires relatively more time and yields poorer F1 scores. Conversely, ABC yields outcomes with slightly shorter inference time than the proposed SMO but at the expense of F1-score performance degradation. These findings indicate that the proposed SMO

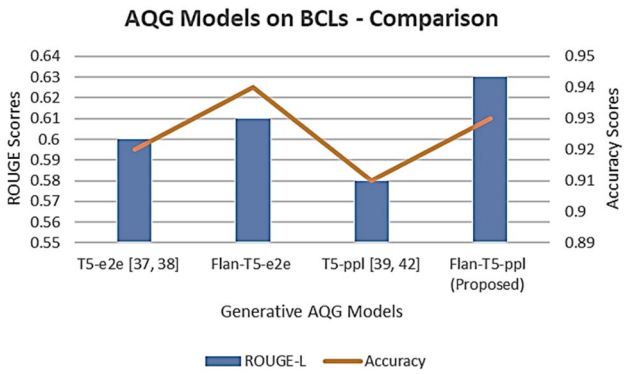

Fig. 9 QG models comparison—BCLs dataset

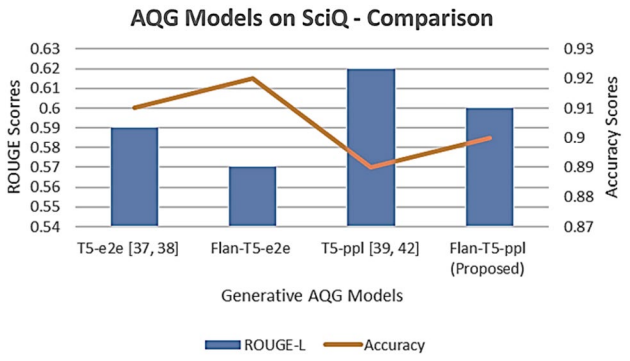

Fig. 10 QG models comparison—SciQ dataset

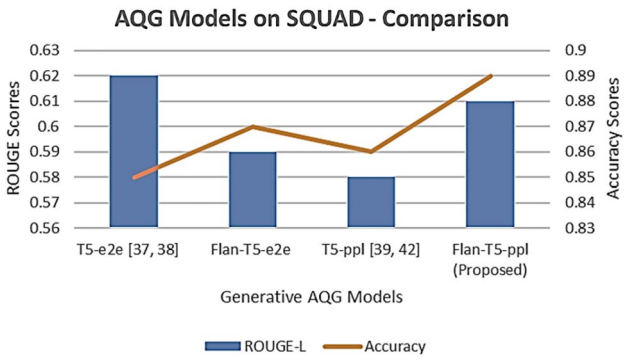

Fig. 11 QG Models comparison—SQUAD dataset

algorithm outperforms across all datasets in terms of inference time, F1-score, and linear convergence time.

Impact of Generative Fine-Tuning (Pipeline T5 Model)

The experiment's findings emphasize the effectiveness of the introduced methods for improving entity mapping. Table 8

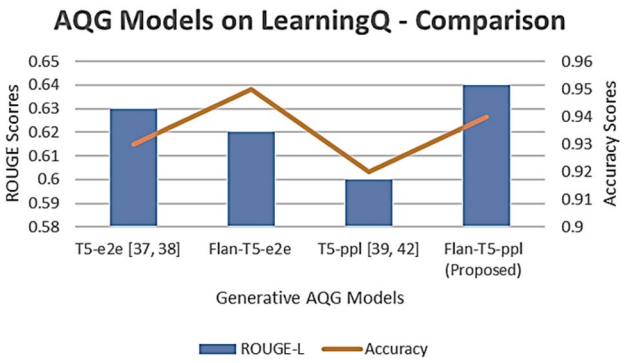

Fig. 12 QG Models comparison—LearningQ dataset

shows the comparative analysis of fine-tuned T5 model with data efficient fine-tuning technique against existing probabilistic next-word generation methods. The comparison demonstrates the superior coherence and relevance achieved by the fine-tuned model. It showcases that the hybridization leverages the T5 model's inherent understanding of text structures to produce more contextually coherent outputs. Furthermore, the implementation of the Spider Monkey Optimizer (SMO) successfully reduces the influence of inter-phrasal dependencies on entity-question template mapping. Experiments with named entity ranking through SMO show improvement in the template mapping process. This is ensured through increase in ROUGE-L scores (Figs. 9, 10, 11, 12) which not only conform structurally to the templates but also contribute contextual co-occurrence.

The end-to-end T5 (baseline) model has longer inference time and did not achieve the best accuracy in the comparative study. In contrast, the proposed finetuned pipeline T5 model performs exceptionally well (Table 8) with higher accuracy, lower computational overhead and reduced inference times across datasets.

Conclusion and Future Work

The challenge of bringing out diverse and relevant auto-generated question to reduce redundancy in AQG is addressed by the proposed LSTM + SMO + T5-ppl (Flan-T5-ppl) model through improved ROUGE scores. This improvement is attained by learning dependency structure-based templates with inter-phrasal/sub-phrasal dependency sequence exclusion/inclusion for Tail Entity (Named Entity) Selection. Experimental results show performance enhancement in named entity relevance in auto-generated questions through higher precision, recall, and F1-scores. In addition, comparative study on nature-inspired algorithms for tail entity selection shows that the Spider Monkey Optimizer (SMO) shows better performance in inter-phrasal

dependency exclusion and sub-phrasal dependency inclusion based on weight updates. It is ensured that mapped entities contribute both structurally and contextually to better auto-generated questions through improvement in n-gram Recall based analysis. The analysis also indicates that SMO, ABC, and PSO algorithms exhibit a linear increase in time complexity as dataset size increases. Among them, SMO consistently outperforms across datasets in terms of F1-score and linear convergence time. Moreover, the proposed fine-tuned pipeline T5 model (Flan-T5-ppl) for question generation shows improved accuracy with less computational overhead and lower inference times over baselines.

In future, integration of external knowledge models will be studied against existing language models for tail entity diversification-based redundancy reduction in AQG.

Acknowledgements This research was carried out at the "E-Learning And Human-Computer Interaction Lab", Department of Computer Applications, National Institute of Technology, Tiruchirappalli, Tamil Nadu, India.

Data Availability All data generated or analysed during this study are openly available.

Declarations

Conflict of Interest The authors declare that they have no conflict of interest.

References

1. Grover K, Kaur K, Tiwari K, Rupali KP. Deep learning based question generation using T5 transformer. In: Garg D, Wong K, Sarangapani J, Gupta SK (eds) Advanced computing. IACC 2020. Communications in Computer and Information Science, vol 1367. Springer, Singapore. 2021. https://doi.org/10.1007/978-981-16-0401-0_18.
2. Perkoff E, Bhattacharyya A, Cai J, Cao J. Comparing neural question generation architectures for reading comprehension. 2023;556–566. https://doi.org/10.18653/v1/2023.bea-1.47.
3. Rathod M, Tu T, Stasaski K. Educational multi-question generation for reading comprehension. 2022;216–223. https://doi.org/10.18653/v1/2022.bea-1.26.
4. Thabet B, Zanichelli N, Zanichelli F. Q&A generation for flashcards within a transformer-based framework. 2023. https://doi.org/10.1007/978-3-031-29800-4_59.
5. Fuadi M, Wibawa A. Automatic question generation from Indonesian texts using text-to-text transformers. 2022;84–89. https://doi.org/10.1109/IEIT56384.2022.9967858.
6. Wolf T, Debut L, Sanh V, Chaumond J, Delangue C, Moi A, Cistac P, Rault T, Louf R, Funtowicz M, Davison J, Shleifer S, Platen P, Ma C, Jernite Y, Plu J, Xu C, Scao T, Gugger S, Rush A. Transformers: state-of-the-art natural language processing. 2020;38–45. https://doi.org/10.18653/v1/2020.emnlp-demos.6.
7. Kumar V, Ramakrishnan G, Li Y-F. Putting the horse before the cart: a generator-evaluator framework for question generation from text. 2019;812–821. https://doi.org/10.18653/v1/K19-1076.
8. Yuan X, Wang T, Gulcehre C, Sordoni A, Bachman P, Zhang S, Subramanian S, Trischler A. Machine comprehension by text-to-text neural question generation. 2017;15–25. https://doi.org/10.18653/v1/W17-2603.
9. Serban I, García-Durán A, Gulcehre C, Ahn S, Chandar S, Courville A, Bengio Y. Generating factoid questions with recurrent neural networks: the 30M factoid question-answer corpus. 2016;588–598. https://doi.org/10.18653/v1/P16-1056.
10. Chan Y-H, Fan Y-C. A recurrent BERT-based model for question generation. 2019;154–162. https://doi.org/10.18653/v1/D19-5821.
11. Subramanian SW, Tong Y, Xingdi Z, Saizheng T, Adam BY. Neural models for key phrase extraction and question generation. 2018;78–88. https://doi.org/10.18653/v1/W18-2609.
12. Das RR, Antariksha M, Souvik DD. A rule based question generation framework to deal with simple and complex sentences. 2016;542–548. https://doi.org/10.1109/ICACCI.2016.7732102.
13. Mulla N, Gharpure P. Automatic question generation: a review of methodologies, datasets, evaluation metrics, and applications. Prog Artif Intell. 2023;12. https://doi.org/10.1007/s13748-023-00295-9.
14. Li L, Zhang L, Zhu C, Mao Z. QGAE: an end-to-end answer-agnostic question generation model for generating question-answer pairs. JUSTC. 2023;53.1. https://doi.org/10.52396/JUSTC-2023-0002.
15. Leite B, Lopes CH. Towards enriched controllability for educational question generation. 2023. https://doi.org/10.1007/978-3-031-36272-9_72.
16. Patel V, Vishwamitra L. Dynamic kernel clustering by spider monkey optimization algorithm. J Classificat. 2023;40. https://doi.org/10.1007/s00357-023-09439-x.
17. Agrawal AG, Deepika S, Rachita SA. Optimum redundancy allocation using spider monkey optimization. Soft Comput. 2023;1–14. https://doi.org/10.1007/s00500-023-08746-0.
18. Sarkar A. Neural coordination through spider monkey optimization-guided weight synchronization. Multimed Tools Appl. 2023;1–30. https://doi.org/10.1007/s11042-023-14443-9.
19. Deußer TH, Lars B, Christian SR. Informed named entity recognition decoding for generative language models. 2023.
20. Yan Y, Cai B, Song S. Nested named entity recognition as building local hypergraphs. Proc AAAI Conf Artif Intell. 2023;37:13878–86. https://doi.org/10.1609/aaai.v37i11.26625.
21. Chen Y, Huang R, Pan L, Huang R, Zheng Q, Chen P. A Controlled attention for nested named entity recognition. Cogn Comput. 2023;15:1–14. https://doi.org/10.1007/s12559-023-10112-z.
22. Yu J, Chen Y, Zheng Q, Wu Y, Chen P. Full-span named entity recognition with boundary regression. Connect Sci. 2023;35:1–27. https://doi.org/10.1080/09540091.2023.2181483.
23. Nanda SJ. Band selection in hyperspectral image with chaotic binary MOCLONAL algorithm. SN Comput Sci. 2022;3:410. https://doi.org/10.1007/s42979-022-01314-7.
24. Nanda SJ, Yadav RP, Gandomi AH, Saraswat M (eds) Data science and applications. In: ICDSA. Lecture notes in networks and systems, vol 820. Springer, Singapore. 2023. https://doi.org/10.1007/978-981-99-7817-5
25. Yadav RS, Ila M, Ravi S, Chitrakant J, Amit S, Sarthak NS. Third international conference on paradigms of communication, computing and data sciences (PCCDS 2022) at MNIT Jaipur in Virtual Mode, JULY 05–07. 2022.
26. Ghosh S, Chopra A, Naskar SK. Learning to rank hypernyms of financial terms using semantic textual similarity. SN Comput Sci. 2023;4:610. https://doi.org/10.1007/s42979-023-02134-z.
27. Sharma S, Srivastava S, Verma P, et al. A comprehensive analysis of indian legal documents summarization techniques. SN Comput Sci. 2023;4:614. https://doi.org/10.1007/s42979-023-01983-y.
28. Dash A, Awachar M, Patel A, et al. Open-domain long-form question-answering using transformer-based pipeline. SN Comput Sci. 2023;4:595. https://doi.org/10.1007/s42979-023-02039-x.

29. Das S, Deb N, Cortesi A, et al. Sentence embedding models for similarity detection of software requirements. SN Comput Sci. 2021;2:69. https://doi.org/10.1007/s42979-020-00427-1.
30. Zhang Z, Song X. An exploratory study on utilising the web of linked data for product data mining. SN Comput Sci. 2023;4:15. https://doi.org/10.1007/s42979-022-01415-3.
31. Shekhawat SS, Shringi S, Sharma H. Twitter sentiment analysis using hybrid Spider Monkey optimization method. Evol Intel. 2021;14:1307–16. https://doi.org/10.1007/s12065-019-00334-2.
32. Zhang Y, Wang Y, Yang J. Lattice LSTM for chinese sentence representation. IEEE Trans Audio Speech Lang Process. 2020. https://doi.org/10.1109/TASLP.2020.2991544.
33. Al-Smadi BS. DeBERTa-BiLSTM: a multi-label classification model of Arabic medical questions using pre-trained models and deep learning. Comput Biol Med. 2024;170: 107921.
34. Naidu MSR, Anilkumar B, Yugandhar D. An exact segmentation of affected part in breast cancer using spider monkey optimization and recurrent neural network. Multimed Tools Appl. 2024;1–19.
35. Dong C, Shen Y, Lin S, Lin Z, Deng Y. A unified framework for contextual and factoid question generation. IEEE Trans Knowl Data Eng. 2023.
36. Goyal R, Kumar P, Singh VP. Automated question and answer generation from texts using text-to-text transformers. Arab J Sci Eng. 2023;1–15.
37. Mulla N, Gharpure P. Genetic algorithm optimized topic-aware transformer-based framework for conversational question generation. Proc Comput Sci. 2023;230:914–22.
38. Manasa P, Malik A, Batra I. Detection of twitter spam using GLoVe vocabulary features, bidirectional LSTM and convolution neural network. SN Comput Sci. 2024;5:206. https://doi.org/10.1007/s42979-023-02518-1.
39. Buzaaba H, Amagasa T. Question answering over knowledge base: a scheme for integrating subject and the identified relation to answer simple questions. SN Comput Sci. 2021;2:25. https://doi.org/10.1007/s42979-020-00421-7.
40. Sharma H, Hazrati G, Bansal JC. Spider monkey optimization algorithm. In: Bansal J, Singh P, Pal N (eds) Evolutionary and swarm intelligence algorithms. Studies in Computational Intelligence, vol 779. Springer, Cham. 2019. https://doi.org/10.1007/978-3-319-91341-4_4.