Semantic Web 0 (0) 1
IOS Press

Generating and Solving Complex Transfer Type Arithmetic Word Problems: An Ontological Approach

Suresh Kumar [a,*] and P. Sreenivasa Kumar [b]
[a] *Computer Science and Engineering, Indian Institute of Technology Madras, Chennai, India*
E-mail: cs18d007@cse.iitm.ac.in
[b] *Computer Science and Engineering, Indian Institute of Technology Madras, Chennai, India*
E-mail: psk@cse.iitm.ac.in

Abstract. Most existing Arithmetic Word Problem (AWP) solvers focus on solving simple examples. Transfer-Case AWPs (TC-AWPs) involve scenarios where objects are transferred between agents. The widely used AWP datasets mainly consist of simple TC-AWPs (problems that involve single object-transfer). Current Large Language Models (LLMs) are capable of solving most of these simple TC-AWPs effectively. In this work, we focus on assessing the solving capability of LLMs (chatGPT and Gemini) for complex TC-AWPs (where multiple types of objects are transferred or more than one transfer of an object is performed). Since the popular AWP datasets contain only simple TC-AWPs, we first generate complex TC-AWPs using an ontological approach. We utilize these complex examples to assess LLMs' word-problem-solving capabilities. We observe that the accuracy of LLMs falls down rapidly as the number of object transfers is increased to 3 or 4. An approach for solving TC-AWPs using ontologies and M/L exists in the literature. We propose an extension of this approach that can handle complex TC-AWPs and find that compared to the current LLMs, the proposed solution gives better accuracy for complex TC-AWPs. We analyze the failed cases of the LLM approach and find that the reasoning capabilities of LLMs need a lot of improvement.

Keywords: Arithmetic Word Problems, Ontology, Large Language Models, Reasoning, SWRL

1. Introduction

Arithmetic Word Problems (AWPs) are elementary math problems in which numbers are dispersed in the problem-text and they can be solved by combining these numbers with basic math operations (addition, subtraction, multiplication, division). Transfer-case AWPs (TC-AWPs), a subset of AWPs, are those word problems where problem-texts involve object transfers among agents. The popular AWP datasets such as AllArith [52], MAWPS [22], and Dolphin [15] contain simple TC-AWPs (i.e., word problems involving a single object-transfer). Authors [24] focused on solving simple TC-AWPs by proposing a knowledge and learning-based approach. They developed TC-Ontology to encode domain knowledge and utilized it in the solution (i.e., automatic solver). Further, authors [25] extended the TC-Ontology and leveraged it for checking the mathematical validity of the machine-generated TC-AWPs. Our work extends the ideas proposed in these two approaches. The proposed work focuses on only the AWP-Tr domain (i.e., both simple and complex TC-AWPs expressed in English). Since the existing datasets do not contain complex TC-AWPs, we first generate such AWPs. We consider a TC-AWP complex when it involves more

[*] Corresponding author. E-mail: cs18d007@cse.iitm.ac.in.

Example of Simple TC-AWP:
Stephen has 17 books. Daniel has 10 books. Stephen gave 12 books to Daniel. How many books does Stephen have now ?

Example of Complex TC-AWP:
Stephen has 17 books and 12 pencils. Daniel has 10 books, 2 pens and 13 pencils. Mike has 15 books. Daniel gave 12 pencils to Stephen and 8 books to Mike. Mike gave 2 books to Stephen. How many books does Mike have now ?

Fig. 1. Examples of simple and complex TC-AWPs

than one object transfer of either single type or multiple types of objects. The examples of simple and complex TC-AWPs are given in the Figure 1.

In the last decade, AWP solving has been widely attempted, and the state-of-the-art (SOTA) approaches have evolved around the following ideas: rule-based solution, statistical modeling, tree-based modeling, template-based solution, incorporating domain knowledge, neural-based models, etc [73]. With the arrival of LLMs, all these models became less popular as LLMs could solve AWPs more effectively. Therefore, the proposed work focuses on assessing the AWP-solving capabilities of LLMs (chatGPT-3.5[1] and Gemini[2]). We focus on the SOTA language models that provide user interfaces to interact with and are also openly available. Therefore, we exclude the SOTA LLM models (such as- WizardMath [34], MAmmoTH [72], LLaMa-2 [61]) and non-LLM models (such as- Text2Math [66, 77]). In this context, incorrect answers are dangerous and can mislead the users. It is assumed that a general user does not have an idea about how to use prompts. Therefore, we assess LLMs as they are. Our assessment includes only complex TC-AWPs. We observed that LLMs could not solve a large proportion of these examples[3]. A few example TC-AWPs that LLMs could not solve are given in the Appendix B. The proposed ontology-based approach performs better than LLMs while solving complex TC-AWPs.

Concerning the TC-AWP domain, the existing works which adopted ontology-based modeling, the *solver* [24] and the *validity-checker* [25], focused on processing word problems sentence-wise. They identified the following four sentence-categories: Before-Transfer (eg., Stephen has 17 books), Transfer (eg., Stephen gave 5 books to Daniel), After-Transfer (eg., Now Daniel has 15 books), and Question (eg., How many books does Stephen have now ?). These four sentence categories are represented as four ontology classes (details are in Section 2). The above mentioned systems incorporated domain knowledge by developing an ontology (namely TC-Ontology). Since the proposed work leverages TC-Ontology, we include a summary in Section 2 and discuss the required extension.

In summary, the proposed work shows the importance of incorporating domain knowledge in the tasks related to the AWP domain, such as generation and solving. Our contributions are:

1. We extend the TC-Ontology to demonstrate the generation of complex TC-AWPs from the ontological representations of simple TC-AWPs. This process required enhancements to the ontology to accommodate the creation of more complex problems.
2. We utilize complex TC-AWPs to evaluate the performance of LLMs in solving these problems. Furthermore, we propose an ontological approach for solving complex TC-AWPs, which showcases enhanced reasoning capabilities and proves to be more effective in solving these complex word problems.

The remaining article is organized as follows - Section 2 briefly discusses the Background and TC-Ontology. Section 3 details the process of generating complex TC-AWPs. Section 4 explains the proposed solver. Section 4 discusses the experimental-setup and results. Section 6 details the related work. Limitations of the proposed approach and conclusions of the work are given in Sections 7 and 8, respectively.

[1] https://chat.openai.com/ (accessed in September 2023)
[2] https://gemini.google.com/ (previously called Bard, accessed in Oct/Nov 2023)
[3] All the results used in this work are from the first run.

2. Background and TC-Ontology

In this section, we discuss the background of ontologies. Since our proposed approach extends and utilizes the TC-Ontology developed in previous work, we also provide a summary of that ontology.

2.1. Background

Ontology is a formal framework for representing knowledge within an application domain. It provides a structured way of modeling concepts (classes), properties (roles) that detail the attributes and relationships of these concepts, and constraints on these properties. This structured representation facilitates consistent understanding, interoperability, and reasoning across systems, making it crucial in fields such as artificial intelligence, the semantic web, and information science. The Resource Description Framework Schema (RDFS) [3] and the Web Ontology Language (OWL) [2] are two widely used frameworks for computationally processing ontologies, each differing in their levels of expressive capabilities. The Resource Description Framework (RDF) [20] is a data modeling standard that facilitates effective information exchange across the web with reasoning capabilities. It serves as the foundation for building RDFS and OWL technologies. In the following, we discuss RDF, RDFS, and OWL in brief.

a) Resource Description Framework (RDF): RDF [20] serves as a foundational framework for encoding, reusing, and exchanging structured metadata. It enables the representation of various resources within a domain through statements structured as triplets (subject, predicate, object). Each component of these triplets is uniquely identified by a URI, where every Internationalized Resource Identifier (IRI) also functions as a URI. For instance, in RDF, the statement "Stephen knows Daniel" would be expressed as <http://example.org/Stephen *http://xmlns.com/foaf/0.1/knows* http://example.org/Daniel>, where the subject (Stephen's profile), predicate (the FOAF vocabulary indicating the relationship "knows"), and object (Daniel's profile) are denoted by IRIs. A URI consists of two parts: a *namespace* and a *local-name*, and it has a global scope. The *namespace* serves as a fixed prefix for the URI, while the *local-name* represents the varying suffix-component. Each URI takes the form *namespace:local-name*. Each domain's linked URIs gets a unique namespace. In RDF, namespaces can be abbreviated for convenience. For instance, using abbreviations like *foaf* for http://xmlns.com/foaf/0.1/ and *ex* for http://example.org/, we can succinctly express the earlier triplet as <ex:Stephen foaf:knows ex:Daniel>. As mentioned earlier, RDF serves as the foundation and it is used to build RDFS and OWL technologies.

b) Resource Description Framework Schema (RDFS): RDFS [3] is a language for describing vocabularies and adding schema to RDF. Where, a vocabulary is a set of classes with specific properties that use the RDF data model to provide essential elements to model a domain. It also specifies entailment-rules/axioms to infer new triples. In essence, RDFS provides a way to create domain-specific vocabulary, often referred as minimal ontology, and can be used to make and infer domain statements.

c) Web Ontology Language (OWL): OWL [2] is a family of semantic web languages designed to formally represent complex knowledge of an application domain. Since it is based on computational logic, computer programs can utilize the knowledge expressed in OWL. It expands upon RDFS by introducing additional constructs for defining classes and properties, such as- cardinality constraints, disjoint classes, etc. The World Wide Web Consortium (W3C) has standardized three variants of OWL with increasing expressive power: OWL-Lite, OWL-DL, and OWL-Full. OWL-DL, which offers maximal expressiveness while ensuring computational completeness and decidability, is particularly suited for our modeling purposes. Description Logics (DLs) [1] are decidable fragments of first-order logic (FOL), and provide a logical foundation to OWL. Since, the expressive power of OWL-DL is sufficient to model the information and the constraints required in our proposed approach, we focus on the $\mathcal{SHOIN}^{(\mathcal{D})}$ description logic and OWL-DL. Certainly, the ontology can also be represented in OWL 2 DL format.

An OWL-DL ontology can be viewed as a pair (T, A), where T represents the Terminological Box (TBox) containing definitions of concepts and properties using vocabulary terminology. The Assertional Box (ABox), represented by A, is used to provide the membership assertions - either as concepts or as properties and is the place where the details of a given concrete situation (extracted from word-problem-text) in the domain are presented.

d) Semantic Web Rule Language (SWRL): SWRL [12] is a rule language for the Semantic Web which was developed to address the limitations of OWL in making assertions using properties, which are important for practical applications [12]. While OWL has significant expressive power, it lacks the capability to represent complex logic involving properties. SWRL extends OWL with Horn-clause rules, enabling more advanced reasoning and overcoming these expressive constraints. In the context of OWL-DL, DL-safe SWRL rules [36] provide feasible reasoning. A DL-safe SWRL rule takes the form $a_1 \wedge a_2 \wedge ... \wedge a_k \rightarrow a_{k+1} \wedge a_{k+2} \wedge ... \wedge a_n$, where each a_i is an atomic unit. Theoretically, each atom represents either $C(a)$ or $P(b, c)$, where C denotes a class, P denotes a property, and a, b, and c are either individuals or variables. In the TC domain, we develop and utilize SWRL rules for several purposes, including checking the feasibility of object transfers, updating ownership of quantities after an object transfer, and other related tasks.

e) SPARQL: The W3C has endorsed SPARQL [46] as a query language for querying RDF graphs. SPARQL allows users to express queries across diverse data sources, whether the data is available as a native RDF graph or accessed through middleware. SPARQL supports four types of queries: SELECT, CONSTRUCT, ASK, and DESCRIBE. In the TC domain, we focus on SELECT queries, which are used to retrieve specific information from the RDF graph of a TC word problem. Consequently, our discussion here is limited to SELECT queries. A SELECT query has two main components: a list of variables to be retrieved and a WHERE clause that specifies the triple patterns to match. In the query, variables are denoted by a "?" followed by the variable name, such as "?x". The triple pattern itself consists of three placeholders, each of which can be either a variable or a specific keyword, used to define the search criteria. For example, the pattern *ex:Stephen foaf:knows ?person* in a SPARQL query can be used to search an RDF graph to find all the people that Stephen knows. In the TC domain, we use SELECT-type SPARQL queries to retrieve the information sought in the question of a TC word problem.

2.2. TC-Ontology

TC-Ontology, developed using OWL-DL, was initially proposed in the KLAUS-Tr system [24] and was reused (after appropriate extension) in the validity-checker system [25]. In the proposed work, we reuse the extended TC-Ontology [25] after making the appropriate changes. Authors [24] analyzed the various TC-AWPs and first devised the vocabulary of the TC-Ontology. The summary is as follows:

Concepts/Classes: Authors treat each AWP as an individual belonging to the class *Word-Problem*. They propose a categorization of the word-problem sentences into individuals belonging to the following ontology classes: *BeforeTransfer (BT)*, *Transfer (TR)*, *AfterTransfer (AT)*, and *Question (QS)*. BT class contains sentences that carry the agent-quantity and associated information *before* the object transfer. Similarly, the AT class contains the sentences that carry the post-transfer information. TR class has sentences that contain object-transfer information. QS class contains the query-sentences that seek the information from the following: before-transfer facts, after-transfer facts, or transfer being carried out. To represent the knowledge present in a sentence, the following concepts are devised: *Agent*, *TC-Quantity*, *PositiveQuantity*, *NegativeQuantity*, etc. The specific agent, like Mike, and a specific number in the problem text become individual members of the classes *Agent* and *TC-Quantity*, respectively.

Properties: Table 1 presents the essential properties (along with domain and range information) available in the TC-Ontology. A domain is a concept/class to which the subject of a Resource Description Framework (RDF[4]) statement using a given property belongs to, while range is the class of statement's object (value).

In addition to concepts and properties, TC-Ontology is equipped with axioms inferring the quantities involved in the math operations (w.r.t. the TC-AWP domain, subtraction, and addition). Appendix A presents the essential axioms of the TC-Ontology. To solve a given AWP, KLAUS-Tr system [24] utilizes ontology inferences (made by axioms) and semantic web rule language (SWRL) [12] rules that capture the knowledge about how object-transfer should affect the RDF graph of the word-problem being solved.

Note that source-to-destination edge in the RDF graph can also be viewed as a triple as follows (source-individual, edge-label, destination-individual). In the discussion below, we also use triples for ease of understanding. The RDF graph is also referred as the ABox (refer A) in this paper. Figure 3 shows the RDF representation of a simple TC-AWP given in Figure 1. This RDF representation utilizes the vocabulary described above. In Section 3, we show how to use the graphical representations of the simple TC-AWPs and add more edges to the graph to obtain the representations of the complex TC-AWPs. This section provides only necessary details of the TC-Ontology. Appendix

[4]https://www.w3.org/RDF/

Property	Domain	Range
hasBT	Word-Problem	BT
hasTR	Word-Problem	TR
hasAT	Word-Problem	AT
hasQuestion	Word-Problem	QS
fromAgent	TR	Agent
toAgent	TR	Agent
hasQuant	Agent	TC-Quantity
hasLost	Agent	TC-Quantity
hasGained	Agent	TC-Quantity
quantValue	TC-Quantity	Literal
quantType	TC-Quantity	Literal

Table 1
Important properties devised in TC-Ontology

```
Agent(?a) ^ hasQuant(?a, ?q) ^ quantValue(?q, ?OldValue) ^
srwlb:add(?NewValue, ?OldValue, 5) → quantValue(?q, ?NewValue)
```

Fig. 2. This SWRL rule will make the ontology inconsistent

A provides more details on TC-Ontology and its use in the KLAUS-Tr [24] and OLGA [25] systems.

Why do we need to extend the TC-Ontology:

a) During the generation of complex TC-AWPs, we need to maintain the order of the sentences, therefore, we add a new data property *hasSequenceNumber*. The property takes domain and range as { $BT \sqcup TR \sqcup AT \sqcup QS$ } and *Integer*, respectively. Section 3 details how the generation module utilizes the *hasSequenceNumber* property.

Since the KLAUS-Tr system focused on solving simple TC-AWPs (generally triggering one subtraction and one addition), the ontology and the SWRL rules developed were sufficient. However, to solve complex TC-AWPs, the proposed system needs to perform a sequence of reasoning steps. The SWRL rule shown in Figure 2 (developed using the similar logic as proposed in KLAUS-Tr), to perform the sequence of reasoning, will make the ontology inconsistent, as it will run forever and attempt to assign an infinite number of values to *quantValue* property. Therefore, to perform the sequence of reasoning, we use two additional properties and the Owlready2 [5] python library. In the following, we explain the required additional properties.

1. We add *hasTransferSequence* data property to capture the sequence of the reasoning (i.e., sequence of the object transfers taking place). The domain and range are *TR* and *Integer*, respectively.

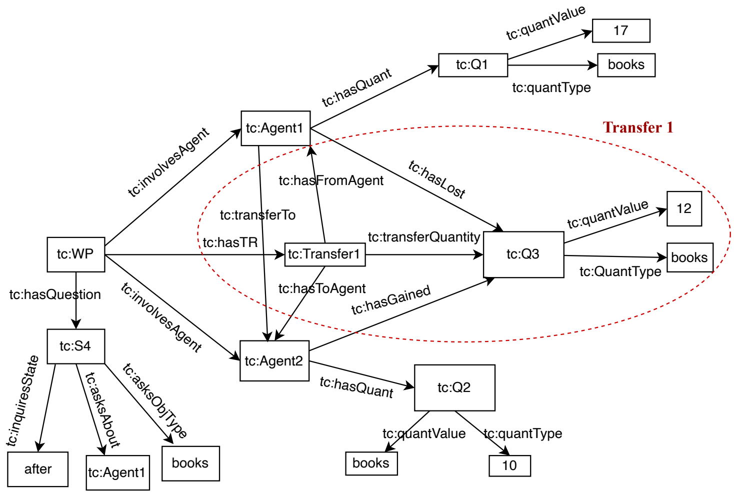
2. We add *hasUpdatedValue* data property to capture the effect of the sequential reasoning. If value of any quantity gets updated using an SWRL rule and it is required in the *transfer* that follows, it creates the sequential reasoning situation. The domain and range are *TC-Quantity* and *literal*, respectively.

Using Owlready2, we write the updated quantity value to the ontology and rerun the reasoner to perform the reasoning required by the next object transfer. Section 4 details how these properties are used in the solution design.

3. Generating complex TC-AWPs

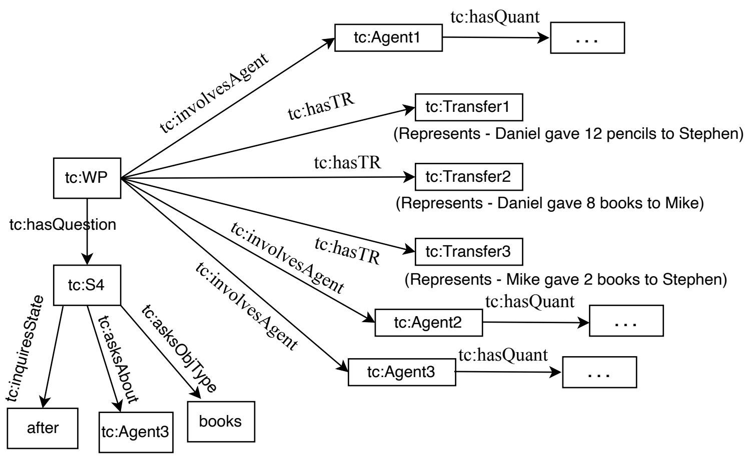
As mentioned earlier, the existing AWP datasets contain simple TC-AWPs. Therefore, we generate complex TC-AWPs as we plan to assess the solving capabilities of LLMs over these examples. The generation process makes use of the TC-Ontology. Authors [25] extended the TC-Ontology to check the mathematical validity of the machine-generated TC-AWPs. Additionally, they showed a way to convert the single-transfer TC-AWPs into two-transfer TC-AWPs. In this work, we adopt a similar idea, use the RDF representations of the simple TC-AWPs, and generate

[5] https://owlready2.readthedocs.io/



Problem Text: Stephen has 17 books. Daniel has 10 books. Stephen gave 12 books to Daniel. How many books does Stephen have now ?

Fig. 3. RDF representation of the simple TC-AWP (Agent1- Stephen, Agent2- Daniel). The red circle represents the factual details of the *transfer*. The term tc represents the namespace of the TC-Ontology.



Problem Text: Stephen has 17 books and 12 pencils. Daniel has 10 books, 2 pens and 13 pencils. Mike has 15 books. Daniel gave 12 pencils to Stephen and 8 books to Mike. Mike gave 2 books to Stephen. How many books does Mike have now ?

Fig. 4. RDF representation of the complex TC-AWP (Agent1- Stephen, Agent2- Daniel, Agent3- Mike). To make the diagram simple and understandable, we do not show the edges representing factual details about the object transfers and the quantities owned by the agents.

complex TC-AWPs (up to four object transfers).

Generation Process: Since TC-Ontology models *transfer* (of objects) as a concept, it is possible to add more individuals of this kind. For example, in the simple TC-AWP given in Figure 1, the transfer-type sentence “Stephen gave 12 books to Daniel” is an individual of the concept *transfer*. To generate complex TC-AWPs, first, we add more hasTR-type and other associated edges (equivalent to adding more individuals of type “Transfer”) to the graphical representations of the simple TC-AWPs. For example, we show a graphical representation of a complex TC-AWP in Figure 4, which is obtained after adding edges to the graph shown in Figure 3.

The generation module takes input as ontology representation ($\mathcal{O}_{si}$) of the simple TC-AWPs and the number of ob-

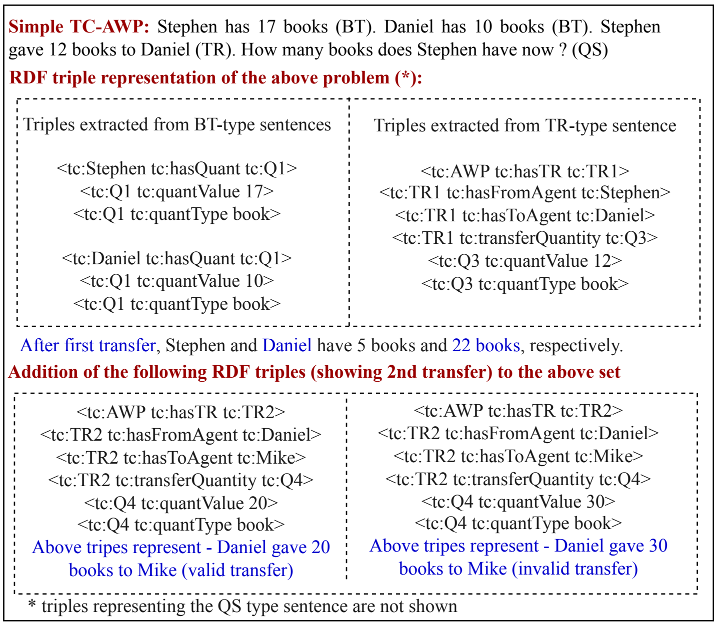


Fig. 5. How the generated TC-word-problems can become invalid

ject transfers expected (N_{tr}) in the generated word problems. *si* represents the ith simple TC-AWP. First, we generate triples representing additional BT-type sentences as they are required to create an appropriate context for multiple object transfers. This additional BT-type sentence(s) will introduce a new agent and maybe a new object type (if the existing scenario has only one object). We generate up to two new BT-type sentences (depending on how many BT-type sentences exist in the simple TC-AWP). We ensure that the generated problems contain three BT-type sentences. Based on the value of N_{tr}, the proposed system generates triples representing an additional object transfer(s). The total additional triples (T_A), that is- triples of BT-type and TR-type sentences, are then accumulated with the "triples of simple TC-AWP ($T(\mathcal{O}_{si})$)" and are taken into $T(\mathcal{O}_{ci})$, which is a triple representation of the complex TC-AWP. *ci* represents the ith complex TC-AWP.

Generating triples of BT-type sentence: A BT-type sentence (e.g., Stephen has 12 books) requires one agent-name string, a number, and one object-type string. We extract 'agent-names and object-types' information from the problem texts of simple TC-AWPs. The agent-name string and the number can be randomly initialized. However, semantic similarity checking (w.r.t. the existing object-type strings) is required to assign the object-type string. We utilize the spaCy[6] library and choose the string that is most semantically similar to the existing object-type strings (which exist in the problem text of the simple TC-AWP). For instance, a complex TC-AWP generated from a simple TC-AWP (which talks about books and pens) should omit new object types like cars, bikes, etc.

Generating triples of TR-type sentence(s): Our approach utilizes the structural information obtained from the triples of the first transfer and ABox information (such as agent-names object-type) available in the ontology and generates triples of the additional transfer. Note that the information about the number of quantities held by the agents (involved in the additional transfer) is available in the existing graph; therefore, the quantity for the additional TR-type sentence is generated appropriately.

The sentences belonging to the simple TC-AWPs are available as annotations in the ontology. We convert the additionally generated triples into sentences using a template-based program script. We use one template for each sentence-category. Note that, the sequence numbers of the sentences belonging to the "simple TC-AWPs" are learned at the preprocessing stage and maintained in the ontology. However, these sequence numbers are adjusted

[6]https://github.com/explosion/spaCy

```
P1:AWP(?wp1) ^                                  P7:TR(tr2) ^
P2:hasTR(?wp1, ?tr1) ^                          P8:hasTR(?wp1, tr2) ^
P3:fromAgent(?tr1, ?ag1) ^                      P9:fromAgent(tr2, agent3) ^
P4:toAgent(?tr1, ?ag2) ^                   →    P10:toAgent(tr2, agent1) ^
P5:involvesTRQuantity(?tr1, ?qtr1) ^            P11:involvesTRQuantity(tr2, qtr2) ^
P6:quantType(?qtr1, ?t1)                        P12:quantValue(qtr2, 12) ^
                                                P13:quantType(qtr2, ?t1)
```

The LHS of the rule examines the structure and information of the existing transfer using various predicates from the domain vocabulary. The RHS of the rule adopts a similar structure and initializes the information for the second transfer.

Fig. 6. SWRL rule showing the generation of additional object-transfer

once new sentences are generated. Newly generated BT-type sentence is placed at the beginning and TR-type sentence(s) is/are placed after the existing TR-type sentence. Finally, we arrange all the sentences using the value of *hasSequenceNumber* property and form the complex TC-AWPs.

Quality of the generated examples: The existing approaches use the following measures to assess the quality of the generated examples: *a) Language Quality Measures:* DL-based approaches primarily follow the idea of "*predict the next word given the previous few words*," and for assessing the quality of the generated text, they adopt metrics such as BLEU-4 [40], METEOR [27], and ROUGE-L [31], etc. Since the proposed approach is ontology-based and does not generate word-by-word text, we do not use the above metrics. However, we focus on another measure to assess the quality, i.e., checking mathematical validity. *b) Mathematical Validity Measure:* As mentioned in the introduction section, a generated problem is considered *valid* if it consists of sufficient information to answer the posed question. We primarily focus on the mathematical validity aspect which is discussed in the following.

The generated problems are valid: As previously discussed, the information extracted from the simple TC word problems (which are all valid) is represented as an RDF graph (i.e., the ABox of the ontology). To generate complex TC word problems, the ontological representation of a word problem needs to be expanded. This expansion involves adding more triples to represent additional BT and TR types of sentences. The generated problems are grammatically correct because the newly created sentences are template-driven. Additionally, the structure of existing triples is used to generate new, similar triples. Our generation process is supported by a domain ontology, ensuring that every generated example is valid by leveraging the structural information from the existing triples. To ensure the problem is also mathematically correct, we generate appropriate quantities for BT and TR types of sentences. Consequently, the new edges (i.e., triples) representing an additional object transfer involve only feasible object-transfer scenario. We show a conceptual illustration in Figure 5. Also, in Figure 6, we show how to generate triples denoting an additional object-transfer by leveraging information from existing triples.

4. Solving complex TC-AWPs

Algorithm 1 shows the pseudo-code of the proposed solver. It takes input as TC-Ontology $\mathcal{O}$ and i^{th} complex TC-AWP (WP_i). *Split-Sentences()* function (line 4) splits the word problem sentences and stores them in the set *S*. The classifier (line 6) takes each word problem sentence and predicts the class label (i.e., BT, TR, AT, or QS). Text classification module achieves 100% accuracy as we use: a) OpenAI API, which provides integration of powerful LLMs with Scikit-learn library. We leveraged the Zero-shot text classifier with OpenAI model *gpt-3.5-turbo*. b) Sentences belonging to the four categories mentioned above have different structures, and sufficient examples are available for training the classifier. Based on the type (label) of the sentence, the system first extracts relevant information from the sentences and populates it into the RDF graph (also called, ABox). This information involves the details about the agents and the quantities they own, the fine details about a transfer (such as who is losing, who is gaining, the exact amount of the quantity, etc.), and what is asked in the question. *Pop-Onto()* is a BERT-based language model trained to pick sentence-parts (agent names etc.) from the word problem text. For ABox extraction from simple TC-AWPs, OLGA [25] trained and deployed BERT-based language models (one for each sentence category). Since the number of sentence-categories are same in the simple and complex problems, we adopt the BERT-based model

Algorithm 1 : Solving complex TC-AWPs

Input: Onto. $\mathcal{O}$, WP_i ▷ WP_i: i^{th} complex TCAWP
S_i = { } ▷ normal sentences of WP_i
C_i = { } ▷ class-labeled sentences of WP_i
S ← Split-Sentences(WP_i) ▷ Let S have k sentences
for *iteration* $j = 0, 1, 2, \ldots, k-1$ **do**
  C_i ← Classifier(S_{ij}) ▷ S_{ij} : j^{th} sentence in WP_i
end for
$\mathcal{O}_i$ ← Pop-Onto(C_i , $\mathcal{O}$) ▷ populates ABox
for *iteration* $l = 1$ to n **do** ▷ n = #transfers
  $\mathcal{O}_{il}$ ← Sync-Reasoner($\mathcal{O}_{i(l-1)}$, SWRL-Rule(l))
end for

Rule	Explanation
P1: TR(?tr) ^ P2: hasTransferSequence(?tr, *l*) ^ P3: hasFromAgent(?tr, ?ag1) ^ P4: hasToAgent(?tr, ?ag2) ^ P5: hasTRQuantity(?tr, ?trq) ^ P6: hasQuant(?ag1, ?q1) ^ P7: quantValue(?q1, ?v1) ^ P8: quantType(?q1, ?t1) ^ P9: hasQuant(?ag2, ?q2) ^ P10: quantValue(?q2, ?v2) ^ P11: quantType(?q2, ?t2) ^ P12: quantValue(?trq, ?trqv) ^ P13: quantType(?trq, ?trqt) ^ P14: swrlb:equal(?t1, ?t2) ^ P15: swrlb:equal(?t1, ?trqt) ^ P16: swrlb:subtract(?vs, ?v1, ?trqv) ^ P17: swrlb:add(?va, ?v2, ?trqv) → P18: hasUpdatedValue(?q1, ?vs) ^ P19: hasUpdatedValue(?q2, ?va)	In antecedent, predicate P1 checks the existence of an object transfer and P2 checks in which sequence it is happening. Predicates P3 to P13 check the details of the object-transfer such as agents involved, quantity, value and type information, etc. Predicates P14 and P15 check whether the units of the quantities match. In consequent, predicates P18 and P19 write the updated values of the quantities owned by the agents involved in the transfer.

- swrlb:subtract(x, y, z) in P14 means x = y - z
- swrlb:add(x, y, z) in P15 means x = y + z

Fig. 7. SWRL rule to affect object transfer

proposed in OLGA and leverage it for extracting ABox information from the problem texts of complex TC-AWPs. Compound sentences (such as- Stephen has 5 books and 10 pens) are converted into two simple sentences using subject distribution at the preprocessing stage.
We develop the SWRL rule to perform reasoning in multiple object transfer situations. The value of the *hasTransferSequence* property (that is, *l*) represents the sequence of the object transfer. Figure 7 presents the SWRL rule and its explanation. $\mathcal{O}_{il}$ represents the ontology state after the lth transfer. We leverage Pellet ontology reasoner [56] in our solution. We place a SPARQL query (based on what the question sentence asks) once the $\mathcal{O}_{il}$ is updated after the nth object transfer.

5. Experimental Setup and Results

The necessary sub-tasks of this work are sentence classification and information extraction (i.e., ontology ABox) from sentences (based on the type of a sentence). Both these sub-tasks are used while generating and solving complex TC-AWPs. We discussed sentence classification in Section 4. In this Section, we discuss results of the ABox extraction and word-problem-solving.
We conducted experiments on a Mac-OS system with a 16GB RAM and Apple M1 processor. We used Google-

colab[7] services for integrating powerful LLMs such as GPT-3.5 into Scikit-learn[8] library for enhancing the performance of sentence-classification task. For ABox extraction, we use BERT-based language models. We use Python and Protégé[9] for ontology development/editing.
Since, the proposed approach is different from a typical ML/DL system, we provide details about reproducing the results in the Appendix C.

5.1. Dataset

We use the dataset AllArith-Tr [24], which contains simple TC-AWPs, and generate the complex TC-AWPs using the proposed ontological approach. We used only 200 problems (50 for each N_{tr} value, see Figure 8) for the assessment, as responses from the LLMs were manually inspected for checking NLU/reasoning failures (see Section 5.3). The average word counts are as follows: 44 (when $N_{tr} = 2$), 56 (when $N_{tr} = 3$), 63 (when $N_{tr} = 4$). The proposed system can convert a simple TC-AWP into a complex TC-AWP. Therefore, the total number of the complex-TC-AWPs the proposed system can generate depends on the number of simple-TC-AWPs given in the input.

5.2. ABox extraction

As mentioned in Section 4, we use BERT-based language models for extracting the ABox information for TC-Ontology (the idea was proposed by OLGA [25]). OLGA used 2K TC-AWP (these word problems included single object transfer only) sentences for training and achieved 76% ABox prediction accuracy (i.e., for 76% of TC-AWPs, ABox was extracted correctly). However, increasing the number of object transfers results in a longer problem text, which lowers the prediction accuracy of ABox. In Algorithm 1, we name the ABox prediction task PopOnto(), as the proposed system needs to populate the ABox into the ontology once it has extracted the information from the word problem text. On complex TC-AWPs, we achieve the following accuracy: 72%(#object-transfers=2), 66% (#object-transfers=3), and 58% (#object-transfers=4). Note that, given a correct ABox, an ontology-based solver always does the correct reasoning, provided domain knowledge is also encoded appropriately.

5.3. Solving complex TC-AWPs

The results (Figure 8) show that as we increase the complexity of the examples, the solving-accuracy of LLMs drops drastically; however, the proposed approach continues to perform well as it leverages domain knowledge while solving.
Analyzing Gemini and chatGPT results: We compare the results based on two aspects- Natural Language Understanding (NLU) and mathematical reasoning. We analyze the responses generated by LLMs for mainly semantics and pragmatic aspects. For the failed cases (examples for which LLMs gave wrong answers), chatGPT always processes the problem text appropriately; however, it could not perform the correct reasoning. In contrast, Gemini could not correctly "understand" the problem text in approximately 34% (average over $N_{tr} = 2$, 3, and 4) of the failed cases, leading to incorrect reasoning. We inspected the responses of LLM systems manually to arrive at these values. Moreover, both the LLMs gave different answers over different runs for the failed cases. Interpretation of the results in Table 2 is as follows: the first row shows that for $N_{tr} = 2$, Gemini failed to understand the problem text in 22.22% of cases, while reasoning alone led to failure in the remaining 77.77%.

[7]https://colab.research.google.com/
[8]https://github.com/iryna-kondr/scikit-llm
[9]https://protege.stanford.edu/

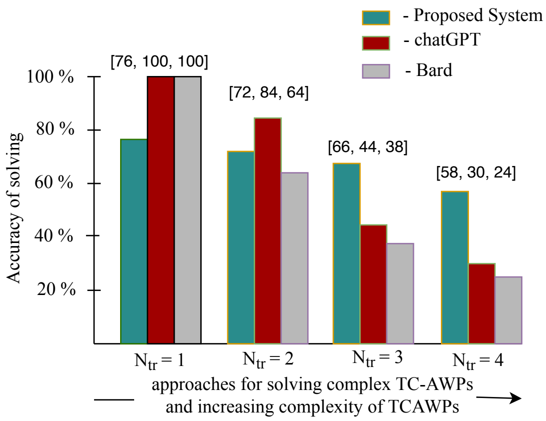


Fig. 8. Comparing accuracy-of-solving of the proposed system w.r.t. LLMs. N_{tr} represents number of object transfers in TC-AWPs.

Table 2

Gemini vs chatGPT - we analyze the failed cases (examples for which LLMs gave wrong answers) and report what percentage of these examples were failed due to NLU and reasoning.

N_{tr}	failing at NLU		failing at reasoning	
	Gemini	chatGPT	Gemini	chatGPT
2	22.22	0	77.77	100
3	35.48	0	64.51	100
4	42.10	0	57.89	100

6. Related Work

In the proposed work, we focus on both the generation and solving aspects of TC-AWPs. Various approaches have been proposed in the literature for solving word problems; however, the generation aspect of word problems is not widely attempted.

6.1. Approaches for AWP Generation

The work [70] proposes a prototype to convert OWL ontologies into word problems. The approach uses the SWAT [49] tool to convert the lexical entries into English statements. For example, the property 'hasType' is converted to 'is a kind of' natural language text. The generated sentences are then grouped together to form a word problem text. Authors [44] use answer set programming (ASP) to generate word problem text from student and teacher requirements. The work [23] proposes a theme-rewriting approach for generating algebra math word problems. The drawback of the approach is that it requires another word problem as an input, resulting in the new problem having a similar template to the input.

The neural network-based approach [75] generates word problems from equations and topics. Two RNN encoders map equations and topics to hidden-vectors and word-representations, respectively. Outputs of these two encoders are then concatenated and fed to a decoder to generate word problems. The method requires a large amount of annotated data. Authors state that the following two types of errors are observed in the generated examples: a) Problem soundness- generated examples lack semantic coherence and b) Equation matchness- template of the input equation partially correlates to the output. However, they do not mention the proportion of such examples. Given an equation-template: $z = x - y$ and topic-words: {has, apples, gave}, the system may generate the following word problem: "'Stephen has 5 apples. He gave 10 apples to Mike. How many apples does Stephen have now ?'. In KLAUS-Tr system [24], authors define the above problem as invalid, as one can not give more than one owns. The neural network-based work does not mention how appropriate domain knowledge is provided to the system.

Attempts has been made towards how to utilize commonsense-knowledge [32, 47] and improve mathematical-validity [68] during generation. Multilingual language models for word problem generation are also being explored [39].
Our proposed work generates word problems related to a specific topic, that is transfer-type AWPs. It takes input as simple TC-AWPs and generates complex TC-AWPs. Like other learning-based approaches, our approach does not require additional training examples and annotations. Since the generation process is backed by domain knowledge, the generated word problems are always valid.

6.2. Approaches for AWP Solving

Before the LLM era, the following were the popular SOTA AWP-solvers: [52, 53, 64, 66, 77]. We skip the discussion on these solvers as we primarily focus on analyzing the AWP-solving capabilities of LLM-based solvers. Authors [69] adopt and test the idea of chain-of-thought prompting in LaMDA [60], GPT-3 [4], PaLM [8], UL2 20B [59] and Codex [5] LLMs to improve the arithmetic reasoning. Word problem solving ability of LLMs is highly dependent on how well the prompts are designed [55]. Since we assess Gemini and chatGPT LLMs as they are, we do not discuss the design of the prompts and also exclude the detailed discussion on prompt-based solutions. However, it appears that the current versions of LLM's we have tested already incorporated chain-of-thought prompt based training. In the output we obtained on sample cases, intermediate statements were generated by the model before the final answer is given. In spite of this, the results on complex problems are not very good. The proposed approach provides a simple solution to this challenging problem (i.e., performing a series of intermediate reasoning steps effectively).
Authors [76] focused on solving math word problems using the GPT-3 model. They focused on analyzing the following three tasks using the GPT-3 model: classifying word problems, extracting equations from the problem text, and generating similar word problem using one given example. The work shows promising results for all these three tasks mentioned above. However, the paper states that directly applying commonsense knowledge to improve word-problem solving remains an issue. Also, scaling up the model size alone seems to be not sufficient for achieving high performance on the reasoning tasks (i.e., arithmetic, commonsense, symbolic) [48]. Therefore, the domain knowledge-based solutions should be explored further. In the proposed work, we show how to encode and utilize domain knowledge while solving word problems.

7. Limitations of the proposed approach

While encoding and utilizing domain knowledge offers several advantages, it also presents some limitations. During the generation process, the encoded domain knowledge ensures the creation of mathematically valid examples. However, because the proposed system converts ontology triples (as part of the solution) into English text using predefined templates, language-diversity may be diminished. Addressing this limitation will be a focus of our future work. Another potential limitation is the extension of the proposed approach to other types of arithmetic word problems (AWPs). This would require the development of a separate ontology for the target domain. Nonetheless, the machine learning (ML) and deep learning (DL) based sub-modules of solution are adaptable and can be extended to other types of AWPs.

8. Conclusions

We investigate the effectiveness of large language models (LLMs) in solving complex transfer-case arithmetic word problems (TC-AWPs). Although the latest state-of-the-art LLMs, such as ChatGPT and Gemini, demonstrate remarkable proficiency in comprehending natural language, they encounter significant difficulties when it comes to solving these complex TC word problems. To address these challenges, we employ a strategy that incorporates domain knowledge, which is encoded through domain ontology and Semantic Web Rule Language (SWRL) rules.

This approach not only aids in generating valid complex-word-problems (from ontological representation of simple problems) but also helps in effectively solving these complex examples. The ontology-based modeling proved effective in tackling complex word problems. The idea of utilizing domain knowledge can be extended to more sophisticated and challenging domains.

References

[1] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi and P.F. Patel-Schneider, *The Description Logic Handbook: Theory, Implementation and Applications*, 2nd edn, Cambridge University Press, USA, 2010. ISBN 0521150116.

[2] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneijder and L.A. Stein, OWL Web Ontology Language Reference, Recommendation, World Wide Web Consortium (W3C), 2004, See http://www.w3.org/TR/owl-ref/.

[3] D. Brickley and R.V. Guha, RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation, World Wide Web Consortium, 2004. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

[4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever and D. Amodei, Language Models are Few-Shot Learners, in: *Advances in Neural Information Processing Systems*, Vol. 33, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan and H. Lin, eds, Curran Associates, Inc., 2020, pp. 1877–1901. https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[5] M. Chen, J. Tworek, H. Jun, Q. Yuan, H.P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F.P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W.H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A.N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever and W. Zaremba, Evaluating Large Language Models Trained on Code, 2021.

[6] T. Chen and C. Guestrin, XGBoost: A Scalable Tree Boosting System, in: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 785–794–. ISBN 9781450342322. doi:10.1145/2939672.2939785.

[7] T.-R. Chiang and Y.-N. Chen, Semantically-Aligned Equation Generation for Solving and Reasoning Math Word Problems, in: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 2656–2668. doi:10.18653/v1/N19-1272. https://www.aclweb.org/anthology/N19-1272.

[8] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H.W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A.M. Dai, T.S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov and N. Fiedel, PaLM: Scaling Language Modeling with Pathways, 2022.

[9] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse and J. Schulman, Training Verifiers to Solve Math Word Problems, *CoRR* **abs/2110.14168** (2021). https://arxiv.org/abs/2110.14168.

[10] C. Fellbaum (ed.), *WordNet: An Electronic Lexical Database*, Language, Speech, and Communication, MIT Press, Cambridge, MA, 1998. ISBN 978-0-262-06197-1.

[11] Y. Goldberg and O. Levy, word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method, 2014, cite arxiv:1402.3722. http://arxiv.org/abs/1402.3722.

[12] I. Horrocks, P.F. Patel-Schneider, S. Bechhofer and D. Tsarkov, OWL rules: A proposal and prototype implementation, *Journal of Web Semantics* **3**(1) (2005), 23–40, Rules Systems. doi:https://doi.org/10.1016/j.websem.2005.05.003. http://www.sciencedirect.com/science/article/pii/S1570826805000053.

[13] M.J. Hosseini, H. Hajishirzi, O. Etzioni and N. Kushman, Learning to Solve Arithmetic Word Problems with Verb Categorization, in: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, Doha, Qatar, 2014, pp. 523–533. doi:10.3115/v1/D14-1058. https://www.aclweb.org/anthology/D14-1058.

[14] https://jena.apache.org/, Apache Jena, accessed 12 July 2021. https://jena.apache.org/documentation/query/parameterized-sparql-strings.html.

[15] D. Huang, S. Shi, C.-Y. Lin, J. Yin and W.-Y. Ma, How well do Computers Solve Math Word Problems? Large-Scale Dataset Construction and Evaluation, in: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Berlin, Germany, 2016, pp. 887–896. doi:10.18653/v1/P16-1084. https://www.aclweb.org/anthology/P16-1084.

[16] D. Huang, S. Shi, C.-Y. Lin and J. Yin, Learning Fine-Grained Expressions to Solve Math Word Problems, in: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Copenhagen, Denmark, 2017, pp. 805–814. doi:10.18653/v1/D17-1084. https://www.aclweb.org/anthology/D17-1084.

[17] D. Huang, J. Liu, C.-Y. Lin and J. Yin, Neural Math Word Problem Solver with Reinforcement Learning, in: *Proceedings of the 27th International Conference on Computational Linguistics*, Association for Computational Linguistics, Santa Fe, New Mexico, USA, 2018, pp. 213–223. https://www.aclweb.org/anthology/C18-1018.

[18] A. Joulin, E. Grave, P. Bojanowski and T. Mikolov, Bag of Tricks for Efficient Text Classification, *arXiv preprint arXiv:1607.01759* (2016).

[19] A. Khoo, Y. Marom and D. Albrecht, Experiments with Sentence Classification, in: *Proceedings of the Australasian Language Technology Workshop 2006*, Sydney, Australia, 2006, pp. 18–25. https://aclanthology.org/U06-1005.

[20] G. Klyne and J.J. Carroll, Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C, 2004. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.

[21] R. Koncel-Kedziorski, H. Hajishirzi, A. Sabharwal, O. Etzioni and S.D. Ang, Parsing Algebraic Word Problems into Equations, *Transactions of the Association for Computational Linguistics* **3** (2015), 585–597. doi:10.1162/tacl_a_00160. https://aclanthology.org/Q15-1042.

[22] R. Koncel-Kedziorski, S. Roy, A. Amini, N. Kushman and H. Hajishirzi, MAWPS: A Math Word Problem Repository, in: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, San Diego, California, 2016, pp. 1152–1157. doi:10.18653/v1/N16-1136. https://www.aclweb.org/anthology/N16-1136.

[23] R. Koncel-Kedziorski, I. Konstas, L. Zettlemoyer and H. Hajishirzi, A Theme-Rewriting Approach for Generating Algebra Word Problems, in: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Austin, Texas, 2016, pp. 1617–1628. doi:10.18653/v1/D16-1168. https://aclanthology.org/D16-1168.

[24] S. Kumar and P.S. Kumar, KLAUS-Tr: Knowledge & learning-based unit focused arithmetic word problem solver for transfer cases, *Natural Language Engineering* (2022), 1–36–. doi:10.1017/S1351324922000511.

[25] S. Kumar and P.S. Kumar, OLGA : An Ontology and LSTM-based approach for generating Arithmetic Word Problems (AWPs) of transfer type, 2022.

[26] N. Kushman, Y. Artzi, L. Zettlemoyer and R. Barzilay, Learning to Automatically Solve Algebra Word Problems, in: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Baltimore, Maryland, 2014, pp. 271–281. doi:10.3115/v1/P14-1026. https://www.aclweb.org/anthology/P14-1026.

[27] A. Lavie and A. Agarwal, METEOR: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments, in: *Proceedings of the Second Workshop on Statistical Machine Translation*, Association for Computational Linguistics, Prague, Czech Republic, 2007, pp. 228–231. https://aclanthology.org/W07-0734.

[28] C.-C. Liang, K.-Y. Hsu, C.-T. Huang, C.-M. Li, S.-Y. Miao and K.-Y. Su, A Tag-based English Math Word Problem Solver with Understanding, Reasoning and Explanation, in: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, Association for Computational Linguistics, San Diego, California, 2016, pp. 67–71. doi:10.18653/v1/N16-3014. https://www.aclweb.org/anthology/N16-3014.

[29] C. Liang, K. Hsu, C. Huang, C. Li, S. Miao and K. Su, A Tag-Based Statistical English Math Word Problem Solver with Understanding, Reasoning and Explanation, in: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, S. Kambhampati, ed., IJCAI/AAAI Press, 2016, pp. 4254–4255. http://www.ijcai.org/Abstract/16/647.

[30] C.-C. Liang, Y.-S. Wong, Y.-C. Lin and K.-Y. Su, A Meaning-Based Statistical English Math Word Problem Solver, in: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Association for Computational Linguistics, New Orleans, Louisiana, 2018, pp. 652–662. doi:10.18653/v1/N18-1060. https://aclanthology.org/N18-1060.

[31] C.-Y. Lin, ROUGE: A Package for Automatic Evaluation of Summaries, in: *Text Summarization Branches Out*, Association for Computational Linguistics, Barcelona, Spain, 2004, pp. 74–81. https://aclanthology.org/W04-1013.

[32] T. Liu, Q. Fang, W. Ding, H. Li, Z. Wu and Z. Liu, Mathematical Word Problem Generation from Commonsense Knowledge Graph and Equations, in: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, M.-F. Moens, X. Huang, L. Specia and S.W.-t. Yih, eds, Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 2021, pp. 4225–4240. doi:10.18653/v1/2021.emnlp-main.348. https://aclanthology.org/2021.emnlp-main.348.

[33] E. Loper and S. Bird, NLTK: The Natural Language Toolkit, *CoRR* **cs.CL/0205028** (2002). http://dblp.uni-trier.de/db/journals/corr/corr0205.html#cs-CL-0205028.

[34] H. Luo, Q. Sun, C. Xu, P. Zhao, J. Lou, C. Tao, X. Geng, Q. Lin, S. Chen and D. Zhang, WizardMath: Empowering Mathematical Reasoning for Large Language Models via Reinforced Evol-Instruct, 2023.

[35] A. Mitra and C. Baral, Learning To Use Formulas To Solve Simple Arithmetic Problems, in: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Berlin, Germany, 2016, pp. 2144–2153. doi:10.18653/v1/P16-1202. https://www.aclweb.org/anthology/P16-1202.

[36] B. Motik, U. Sattler and R. Studer, Query Answering for OWL-DL with Rules, *Web Semant.* **3**(1) (2005), 41–60–. doi:10.1016/j.websem.2005.05.001.

[37] A. Mukherjee and U. Garain, A review of methods for automatic understanding of natural language mathematical problems, *Artif. Intell. Rev.* **29**(2) (2008), 93–122. doi:10.1007/s10462-009-9110-0.

[38] R. Navigli, Natural Language Understanding: Instructions for (Present and Future) Use, in: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, International Joint Conferences on Artificial Intelligence Organization, 2018, pp. 5697–5702. doi:10.24963/ijcai.2018/812.

[39] K. Niyarepola, D. Athapaththu, S. Ekanayake and S. Ranathunga, Math Word Problem Generation with Multilingual Language Models, in: *Proceedings of the 15th International Conference on Natural Language Generation*, S. Shaikh, T. Ferreira and A. Stent, eds, Association for Computational Linguistics, Waterville, Maine, USA and virtual meeting, 2022, pp. 144–155. doi:10.18653/v1/2022.inlg-main.12. https://aclanthology.org/2022.inlg-main.12.

[40] K. Papineni, S. Roukos, T. Ward and W.-J. Zhu, Bleu: a Method for Automatic Evaluation of Machine Translation, in: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, 2002, pp. 311–318. doi:10.3115/1073083.1073135. https://aclanthology.org/P02-1040.

[41] A. Patel, S. Bhattamishra and N. Goyal, Are NLP Models really able to Solve Simple Math Word Problems?, in: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, Online, 2021, pp. 2080–2094. doi:10.18653/v1/2021.naacl-main.168. https://aclanthology.org/2021.naacl-main.168.

[42] D.K. Pattipati, R. Nasre and S.K. Puligundla, OPAL: An extensible framework for ontology-based program analysis, *Softw. Pract. Exp.* **50**(8) (2020), 1425–1462. doi:10.1002/spe.2821.

[43] J. Pennington, R. Socher and C. Manning, GloVe: Global Vectors for Word Representation, in: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1532–1543. doi:10.3115/v1/D14-1162. https://www.aclweb.org/anthology/D14-1162.

[44] O. Polozov, E. O'Rourke, A.M. Smith, L. Zettlemoyer, S. Gulwani and Z. Popovic, Personalized Mathematical Word Problem Generation, in: *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, AAAI Press, 2015, pp. 381–388–. ISBN 9781577357384.

[45] protege.stanford.edu, Protégé, 2012. http://protege.stanford.edu/.

[46] E. Prud'hommeaux and A. Seaborne, SPARQL Query Language for RDF, 2008. http://www.w3.org/TR/rdf-sparql-query/.

[47] L. Qin, J. Liu, Z. Huang, K. Zhang, Q. Liu, B. Jin and E. Chen, A Mathematical Word Problem Generator with Structure Planning and Knowledge Enhancement, in: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, Association for Computing Machinery, New York, NY, USA, 2023, pp. 1750–1754–. ISBN 9781450394086. doi:10.1145/3539618.3591937.

[48] J.W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, H.F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, E. Rutherford, T. Hennigan, J. Menick, A. Cassirer, R. Powell, G. van den Driessche, L.A. Hendricks, M. Rauh, P. Huang, A. Glaese, J. Welbl, S. Dathathri, S. Huang, J. Uesato, J. Mellor, I. Higgins, A. Creswell, N. McAleese, A. Wu, E. Elsen, S.M. Jayakumar, E. Buchatskaya, D. Budden, E. Sutherland, K. Simonyan, M. Paganini, L. Sifre, L. Martens, X.L. Li, A. Kuncoro, A. Nematzadeh, E. Gribovskaya, D. Donato, A. Lazaridou, A. Mensch, J. Lespiau, M. Tsimpoukelli, N. Grigorev, D. Fritz, T. Sottiaux, M. Pajarskas, T. Pohlen, Z. Gong, D. Toyama, C. de Masson d'Autume, Y. Li, T. Terzi, V. Mikulik, I. Babuschkin, A. Clark, D. de Las Casas, A. Guy, C. Jones, J. Bradbury, M.J. Johnson, B.A. Hechtman, L. Weidinger, I. Gabriel, W. Isaac, E. Lockhart, S. Osindero, L. Rimell, C. Dyer, O. Vinyals, K. Ayoub, J. Stanway, L. Bennett, D. Hassabis, K. Kavukcuoglu and G. Irving, Scaling Language Models: Methods, Analysis & Insights from Training Gopher, *CoRR* **abs/2112.11446** (2021). https://arxiv.org/abs/2112.11446.

[49] S. Robert, M. James, W. Sandra, P. Richard and T. Allan, Automating generation of textual class definitions from OWL to English, in: *Journal of Biomedical Semantics*, 2011, pp. 2(2):S5–. doi:https://doi.org/10.1186/2041-1480-2-S2-S5.

[50] S. Roy and D. Roth, Solving General Arithmetic Word Problems, in: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 1743–1752. doi:10.18653/v1/D15-1202. https://aclanthology.org/D15-1202.

[51] S. Roy and D. Roth, Illinois Math Solver: Math Reasoning on the Web, in: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, Association for Computational Linguistics, San Diego, California, 2016, pp. 52–56. doi:10.18653/v1/N16-3011. https://www.aclweb.org/anthology/N16-3011.

[52] S. Roy and D. Roth, Unit Dependency Graph and Its Application to Arithmetic Word Problem Solving, in: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, AAAI Press, 2017, pp. 3082–3088–.

[53] S. Roy and D. Roth, Mapping to Declarative Knowledge for Word Problem Solving, *Transactions of the Association for Computational Linguistics* **6**(0) (2018), 159–172. https://transacl.org/ojs/index.php/tacl/article/view/1319.

[54] S. Roy, T. Vieira and D. Roth, Reasoning about Quantities in Natural Language, *Transactions of the Association for Computational Linguistics* **3** (2015), 1–13. doi:10.1162/tacl_a_00118. https://www.aclweb.org/anthology/Q15-1001.

[55] F. Shi, X. Chen, K. Misra, N. Scales, D. Dohan, E. Chi, N. Schärli and D. Zhou, Large language models can be easily distracted by irrelevant context, in: *Proceedings of the 40th International Conference on Machine Learning*, ICML'23, JMLR.org, 2023.

[56] E. Sirin, B. Parsia, B.C. Grau, A. Kalyanpur and Y. Katz, Pellet: A practical OWL-DL reasoner, *Web Semant.* **5**(2) (2007), 51–53–. doi:10.1016/j.websem.2007.03.004.

[57] S.S. Sundaram and D. Khemani, Natural Language Processing for Solving Simple Word Problems, in: *Proceedings of the 12th International Conference on Natural Language Processing*, NLP Association of India, Trivandrum, India, 2015, pp. 394–402. https://www.aclweb.org/anthology/W15-5955.

[58] S.S. Sundaram, S. Gurajada, M. Fisichella, S.S. Abraham et al., Why are NLP Models Fumbling at Elementary Math? A Survey of Deep Learning based Word Problem Solvers, *arXiv preprint arXiv:2205.15683* (2022).

[59] Y. Tay, M. Dehghani, V.Q. Tran, X. Garcia, J. Wei, X. Wang, H.W. Chung, S. Shakeri, D. Bahri, T. Schuster, H.S. Zheng, D. Zhou, N. Houlsby and D. Metzler, UL2: Unifying Language Learning Paradigms, 2023.

[60] R. Thoppilan, D.D. Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, Y. Li, H. Lee, H.S. Zheng, A. Ghafouri, M. Menegali, Y. Huang, M. Krikun, D. Lepikhin, J. Qin, D. Chen, Y. Xu, Z. Chen, A. Roberts, M. Bosma, Y. Zhou, C. Chang, I. Krivokon, W. Rusch, M. Pickett, K.S. Meier-Hellstern, M.R. Morris, T. Doshi, R.D. Santos, T. Duke, J. Soraker, B. Zevenbergen, V. Prabhakaran, M. Diaz, B. Hutchinson, K. Olson, A. Molina, E. Hoffman-John, J. Lee, L. Aroyo, R. Rajakumar, A. Butryna, M. Lamm, V. Kuzmina, J. Fenton, A. Cohen, R. Bernstein, R. Kurzweil, B.A. y Arcas, C. Cui, M. Croak, E.H. Chi and Q. Le, LaMDA: Language Models for Dialog Applications, *CoRR* **abs/2201.08239** (2022). https://arxiv.org/abs/2201.08239.

[61] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C.C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P.S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E.M. Smith, R. Subramanian, X.E. Tan, B. Tang, R. Taylor, A. Williams, J.X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov and T. Scialom, Llama 2: Open Foundation and Fine-Tuned Chat Models, 2023.

[62] V.E. V and P.S. Kumar, Automated generation of assessment tests from domain ontologies, *Semantic Web* **8**(6) (2017), 1023–1047. doi:10.3233/SW-170252.

[63] V.E. V and S.K. Puligundla, A novel approach to generate MCQs from domain ontology: Considering DL semantics and open-world assumption, *J. Web Semant.* **34** (2015), 40–54. doi:10.1016/j.websem.2015.05.005.

[64] L. Wang, D. Zhang, L. Gao, J. Song, L. Guo and H.T. Shen, MathDQN: Solving Arithmetic Word Problems via Deep Reinforcement Learning, in: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, S.A. McIlraith and K.Q. Weinberger, eds, AAAI Press, 2018, pp. 5545–5552. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16749.

[65] L. Wang, Y. Wang, D. Cai, D. Zhang and X. Liu, Translating Math Word Problem to Expression Tree, in: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, E. Riloff, D. Chiang, J. Hockenmaier and J. Tsujii, eds, Association for Computational Linguistics, 2018, pp. 1064–1069. doi:10.18653/v1/d18-1132.

[66] L. Wang, D. Zhang, J. Zhang, X. Xu, L. Gao, B.T. Dai and H.T. Shen, Template-Based Math Word Problem Solvers with Recursive Neural Networks, in: *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19, AAAI Press, 2019. ISBN 978-1-57735-809-1. doi:10.1609/aaai.v33i01.33017144.

[67] Y. Wang, X. Liu and S. Shi, Deep Neural Solver for Math Word Problems, in: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Copenhagen, Denmark, 2017, pp. 845–854. doi:10.18653/v1/D17-1088. https://www.aclweb.org/anthology/D17-1088.

[68] Z. Wang, A. Lan and R. Baraniuk, Math Word Problem Generation with Mathematical Consistency and Problem Context Constraints, in: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, M.-F. Moens, X. Huang, L. Specia and S.W.-t. Yih, eds, Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 2021, pp. 5986–5999. doi:10.18653/v1/2021.emnlp-main.484. https://aclanthology.org/2021.emnlp-main.484.

[69] J. Wei, X. Wang, D. Schuurmans, M. Bosma, b. ichter, F. Xia, E. Chi, Q.V. Le and D. Zhou, Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, in: *Advances in Neural Information Processing Systems*, Vol. 35, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho and A. Oh, eds, Curran Associates, Inc., 2022, pp. 24824–24837. https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

[70] S. Williams, Generating mathematical word problems, in: *2011 AAAI Fall symposium series*, 2011.

[71] M. Wooldridge, Artificial Intelligence Requires More Than Deep Learning – but What, Exactly?, *Artificial Intelligence* **289** (2020), 103386. doi:10.1016/j.artint.2020.103386.

[72] X. Yue, X. Qu, G. Zhang, Y. Fu, W. Huang, H. Sun, Y. Su and W. Chen, MAmmoTH: Building Math Generalist Models through Hybrid Instruction Tuning, 2023.

[73] D. Zhang, L. Wang, N. Xu, B.T. Dai and H.T. Shen, The Gap of Semantic Parsing: A Survey on Automatic Math Word Problem Solvers, *CoRR* **abs/1808.07290** (2018). http://arxiv.org/abs/1808.07290.

[74] L. Zhou, S. Dai and L. Chen, Learn to Solve Algebra Word Problems Using Quadratic Programming, in: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 817–822. doi:10.18653/v1/D15-1096. https://www.aclweb.org/anthology/D15-1096.

[75] Q. Zhou and D. Huang, Towards Generating Math Word Problems from Equations and Topics, in: *Proceedings of the 12th International Conference on Natural Language Generation*, Association for Computational Linguistics, Tokyo, Japan, 2019, pp. 494–503. doi:10.18653/v1/W19-8661. https://aclanthology.org/W19-8661.

[76] M. Zong and B. Krishnamachari, Solving Math Word Problems concerning Systems of Equations with GPT-3, *Proceedings of the AAAI Conference on Artificial Intelligence* **37**(13) (2023), 15972–15979. doi:10.1609/aaai.v37i13.26896. https://ojs.aaai.org/index.php/AAAI/article/view/26896.

[77] Y. Zou and W. Lu, Text2Math: End-to-end Parsing Text into Math Expressions, in: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Association for Computational Linguistics, Hong Kong, China, 2019, pp. 5327–5337. doi:10.18653/v1/D19-1536. https://www.aclweb.org/anthology/D19-1536.

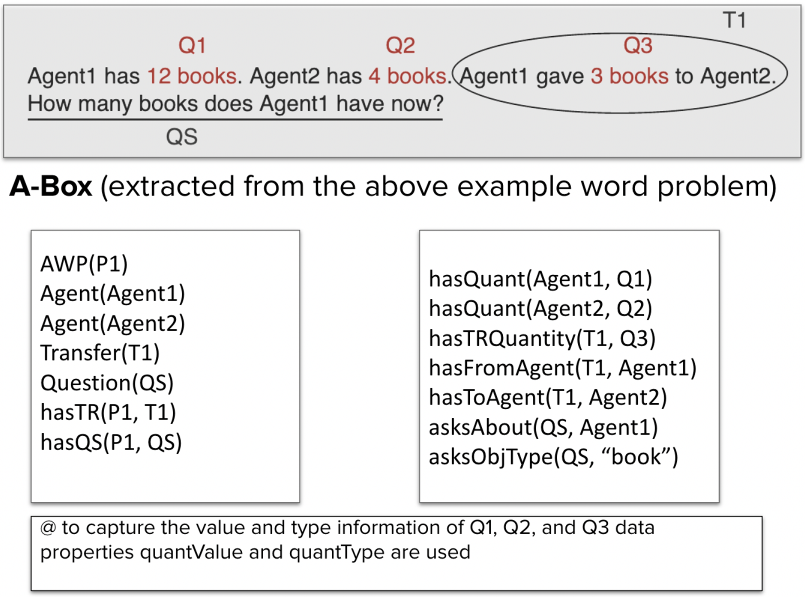


Fig. 9. ABox extracted from an example word problem

Appendix A. TC-Ontology : details and its use in KLAUS-Tr and OLGA systems

An ontology is a structured representation of knowledge that defines the concepts within a specific domain and the relationships between them. Ontologies typically consist of classes (representing concepts or categories), properties (depicting relationships between classes), and instances (individual members of classes). By organizing information in a hierarchical and interconnected manner, ontologies facilitate better knowledge management, interoperability, and reasoning about the entities and their interactions within a given domain.

Two major components of an ontology are the Terminological Box (TBox) and the Assertional Box (ABox). The TBox defines the vocabulary, concepts, and their relationships within a specific domain, establishing a structured framework for understanding and representing knowledge. The ABox, on the other hand, captures instances and specific data related to those instances, providing a means to describe individual objects and their properties within the defined concepts. These components together form a comprehensive ontology, facilitating effective knowledge representation and sharing.

Axioms are logical statements of TBox that say what is true in an application domain. In the following, we mention the important axioms devised for TC-AWP domain. Here, A.01 to A.04 are concept inclusion axioms, whereas A.05 and A.06 are concept equivalence axioms.

A.01: $\exists\ hasQuant.\ TC\text{-}Quantity \sqsubseteq Agent$

(Anyone who owns a TC quantity is an agent)

A.02: $TC\text{-}Quantity \sqsubseteq PositiveQuantity$

(Every TC quantity is a positive quantity)

A.03: $MinuendQuantity \sqsubseteq TC\text{-}Quantity$

A.04: $SubtrahendQuantity \sqsubseteq TC\text{-}Quantity$

A.05: $MinuendQuantity \equiv (TC\text{-}Quantity\ \sqcap$
$\exists\ isOwnedBy.\ Agent)$

(A.05 expresses "Minuend quantity is a TC quantity that is owned by an agent")

A.06: $SubtrahendQuantity \equiv (TC\text{-}Quantity\ \sqcap$
$\exists\ isGainedBy.\ Agent\ \sqcap \exists\ isLostBy.\ Agent)$

(A.06 expresses "Subtrahend quantity is a TC quantity that is gained by an agent and lost by an agent")

Note that the properties isOwnedBy, isGainedBy and isLostBy are inverse-properties of the object-properties hasQuant, hasGained and hasLost, respectively. Axioms A.05 and A.06 infer the minuend and subtrahend quantities of a subtraction operation, respectively.
In Section 2, we presented a summary of the vocabulary of the TC-Ontology and some important axioms are mentioned above. The detailed information of the TC-Ontology is available in KLAUST-Tr [24]. Both KLAUS-Tr and OLGA systems process word problems sentence-wise. Based on the type of a sentence, these systems extract important information and populate the Ontology ABox. An example ABox extracted from a word problem is given in Figure 9. In contrast, P1 represents an example word problem. Q1, Q2, and Q3 represent quantities and T1 represents the *transfer*. The data property assertions for quantities Q1, Q2, and Q3 (using the quantValue and quantType data properties) are straightforward and thus not shown.

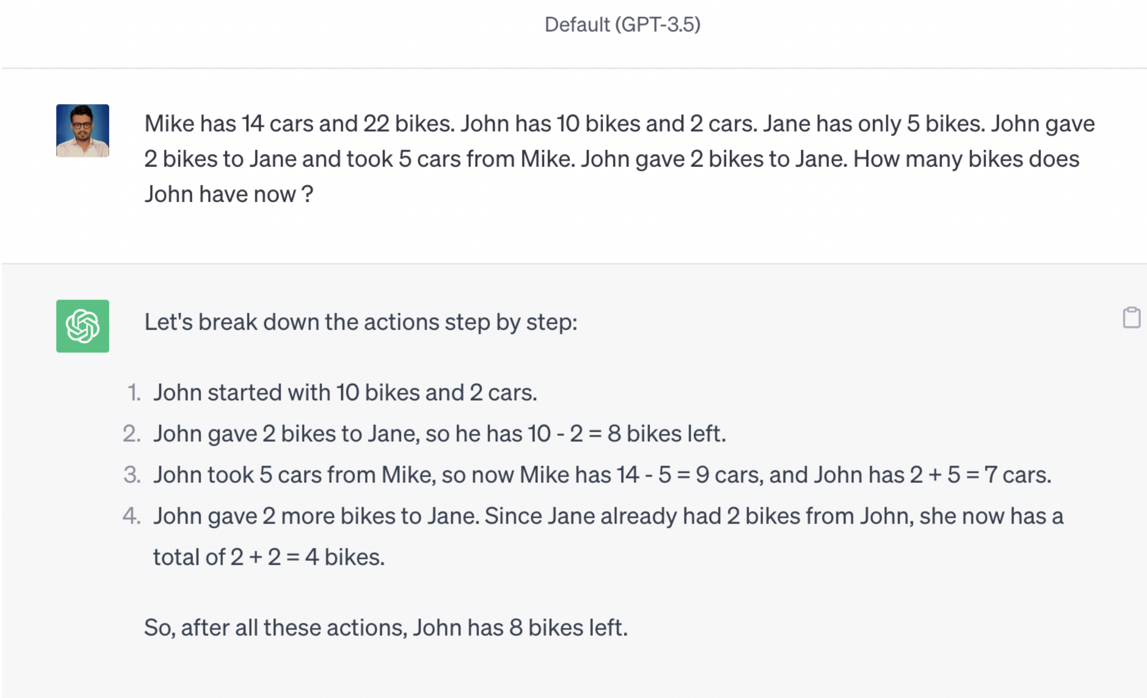
Appendix B. Solving complex TC-AWPs with ChatGPT and Gemini

The results on an example TC-AWP involving three object transfers are given below:

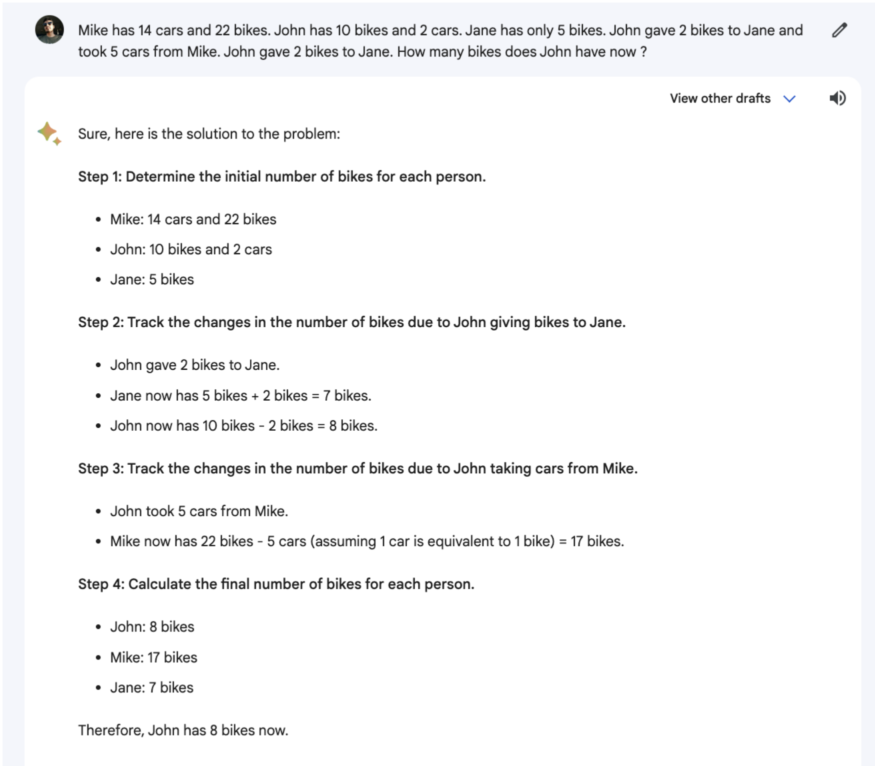
Q1. Mike has 14 cars and 22 bikes. John has 10 bikes and 2 cars. Jane has only 5 bikes. John gave 2 bikes to Jane and took 5 cars from Mike. John gave 2 bikes to Jane. How many bikes does John have now ?

Correct Answer: 6
(Both ChatGPT and Gemini gave answer as 8)

Response by chatGPT:



Response by Gemini:



Mike has 14 cars and 22 bikes. John has 10 bikes and 2 cars. Jane has only 5 bikes. John gave 2 bikes to Jane and took 5 cars from Mike. John gave 2 bikes to Jane. How many bikes does John have now ?

View other drafts

Sure, here is the solution to the problem:

Step 1: Determine the initial number of bikes for each person.

- Mike: 14 cars and 22 bikes
- John: 10 bikes and 2 cars
- Jane: 5 bikes

Step 2: Track the changes in the number of bikes due to John giving bikes to Jane.

- John gave 2 bikes to Jane.
- Jane now has 5 bikes + 2 bikes = 7 bikes.
- John now has 10 bikes - 2 bikes = 8 bikes.

Step 3: Track the changes in the number of bikes due to John taking cars from Mike.

- John took 5 cars from Mike.
- Mike now has 22 bikes - 5 cars (assuming 1 car is equivalent to 1 bike) = 17 bikes.

Step 4: Calculate the final number of bikes for each person.

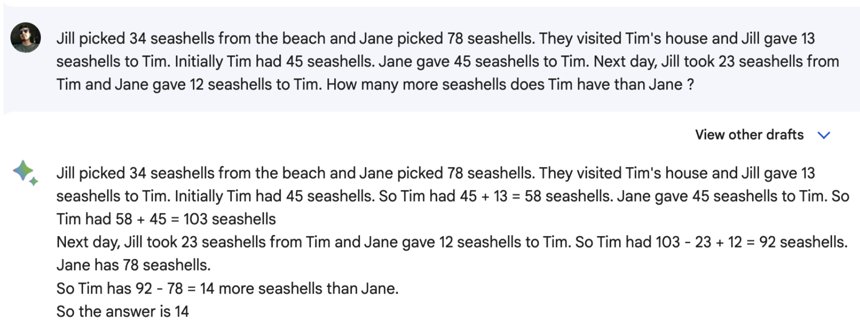
- John: 8 bikes
- Mike: 17 bikes
- Jane: 7 bikes

Therefore, John has 8 bikes now.

The results on an example TC-AWP involving four object transfers are given below:

Q2. Jill picked 34 seashells from the beach and Jane picked 78 seashells. They visited Tim's house and Jill gave 13 seashells to Tim. Initially Tim had 45 seashells. Jane gave 45 seashells to Tim. Next day, Jill took 23 seashells from Tim and Jane gave 12 seashells to Tim. How many more seashells does Tim have than Jane ?

Correct Answer: 71
(Gemini and ChatGPT gave answers 14 and 21, respectively)
Response by Gemini:



Jill picked 34 seashells from the beach and Jane picked 78 seashells. They visited Tim's house and Jill gave 13 seashells to Tim. Initially Tim had 45 seashells. Jane gave 45 seashells to Tim. Next day, Jill took 23 seashells from Tim and Jane gave 12 seashells to Tim. How many more seashells does Tim have than Jane ?

View other drafts

Jill picked 34 seashells from the beach and Jane picked 78 seashells. They visited Tim's house and Jill gave 13 seashells to Tim. Initially Tim had 45 seashells. So Tim had 45 + 13 = 58 seashells. Jane gave 45 seashells to Tim. So Tim had 58 + 45 = 103 seashells
Next day, Jill took 23 seashells from Tim and Jane gave 12 seashells to Tim. So Tim had 103 - 23 + 12 = 92 seashells.
Jane has 78 seashells.
So Tim has 92 - 78 = 14 more seashells than Jane.
So the answer is 14

Response by chatGPT:

Appendix C. Reproducing the results

Note that implementation of the proposed system is different from a typical ML/DL model. There are three key components of the proposed system: **A)** Sentence Classification: we use zero-shot text classification with OpenAI gpt-3.5-turbo model, **B)** Extract important information from the word problem sentences - we use BERT-based language models, and **C)** Ontology editor - we use Protégé tool and Owlready2 python library.
The overall process-flow of the proposed solver is: The sentences of the word problem at hand are labeled using the sentence-classification-module. We deploy BERT-based LMs (we use the architecture proposed by the OLGA system) to extract important information from the sentences (based on the labels). This information is populated into Ontology using the Owlready2 python library (populating the Assertional-Box (ABox) of Ontology). A domain ontology has two components: Terminological-Box (TBox) and Assertional-Box (ABox). Using the Protégé tool, we encode the domain knowledge about transfer-type word problems (TBox of the Ontology). To solve a given word problem, we utilize the encoded domain knowledge, ABox information, and semantic web rule language (SWRL) rules (we develop these rules and make them available inside the ontology, under SWRL Tab) that update the state of the ontology (i.e. computes the effects of the object transfer). Therefore, building two splits (train and test) are relevant to the components/modules (A) and (B) only, which are similar to the modules used in the existing systems KLAUS-Tr [24] and OLGA [25], respectively. Therefore, we skip these details.