A Stacked Multi-Layered Perceptron - LLM Model for Extracting the Relations in Textual Descriptions

***Abstract*—The automated extraction of the relations among the concepts in textual descriptions is important for problems that require creating implementations of those descriptions, e.g., synthesizing engineering designs and computer code. However, relation extraction remains challenging despite significant recent progress in Natural Language Processing. This paper proposes a novel, two-layered method to create structural representations of the relations in a text. Inspired by work in neuroscience, an upper layer implemented as a multi-layer perceptron models the behavior of closed class words (words with a fixed meaning), like prepositions and conjunctions. The lower layer prompts a Large Language Model to extract the nouns and verbs, which are then introduced into the structural representation produced by the upper layer. Experiments showed an average improvement of 13.6% as compared to using only LLMs.**

***Index Terms*—relations, dialog, structure, LLM**

I. Introduction

Relation extraction has been a main topic in semantic understanding of data in domains, like visual processing, speech understanding, text analysis, and music and art interpretation [1], [2]. Relations connect basic concepts, e.g., objects or words, into more and more complex structural constructs, while still their meaning can cost-effectively be understood at a reasonable level of precision and correctness. For example, grammars have been the traditional instrument in understanding natural language semantics by using explicit syntax rules [3]. However, recent work shows that modern Machine Learning (ML) models, like transformers and Large Language Models (LLMs), offer an intriguing promise in tackling relation extraction without explicit rules [2], [4].

Still, relation extraction using ML models, like LLMs, faces a number of challenges that originate in the "mismatches" between the mathematical processing inside the models [5] and the way in which humans process semantics, as explained by neuroscience [6], [7]. For example, the noun positions in an enumeration does not impact their meaning for humans, but affects the mathematical computation of the self-attention and normalization layer matrices of the models. Also, neglecting stop words or processing hierarchical constructs through linear mechanisms, i.e. sequences of tokens, might introduce errors in relation extraction. The analysis is further complicated for dialog text, which is often colloquial, ambiguous, and grammatically incorrect. In general, a more precise understanding is needed about the degree to which the mathematical computations of ML models performed on massive amounts of data can result in more relation errors.

This paper proposes a two-layered model for relation extraction from dialog text, including equivalences, alternatives, and hierarchical constructs. The top layer, implemented using a Feedforward Multi-Layered Perceptron (MLP) tackles closed class words, e.g., conjunctions, prepositions, and determiners, which are often important for relation (and structure) definition, but ignored as stop words by LLMs. The bottom-layer uses an LLM (GPT-4) to extract the relations of type subject - verb - object (SVO), as LLMs can reliably extract SVOs. SVOs are then combined with the MLP outputs to produce a structural, graph representation of textual inputs. Experiments discussed the model performance. An average improvement of 13.6% was observed for a sample of sentences when using the two-layered model as compared to LLMs, only.

The paper structure is as follows. Section II describes the problem followed by related work. Section IV presents an experimental and theoretical assessment of LLM performance for relation extraction. Section V details the proposed architecture. Experiments are in Section VI. Conclusions end the paper.

II. Problem Description

The objective of this work was to automatically construct for textual descriptions the structural representations that describe the relations among the description's nouns, verbs, pronouns, and adjectives. Once generated, the structural representations can be automatically translated into an implementation by mapping the nodes and edges in the representation to connected building blocks, and then optimizing the mapping [8]. In particular, we considered textual descriptions of the verbal dialog among the members of a team that jointly devised an implementation, i.e. a computer program or circuit design.

Description of structural representations. There are multiple, similar notations that can express structural representations, but for its simplicity, we chose the one that has nodes for the concepts (nouns and pronouns) of a textual description, edges connecting the nodes to describe the relations introduced by verbs, and structuring points to express the certain semantic roles of words, like equivalencies and alternatives. Structuring points are part of the representations. Adjectives can qualify nouns and adverbs the verbs.

The edge of a verb together with its associated nodes form a *structural frame*, as shown in Fig. 2. The right part of the figure depicts the structural representation for the dialog sentence "`distance close to sum and product and then set it to high`". Frames can be linked together to express relations, like sequencing (introduced by "`and then`"), alternatives (i.e. due to the conjunction "`or`"), and equivalencies (e.g., through the conjunction "`and`"). *Structuring points* indicate words or frames with the same role in

a sentence, like the words "sum" and "product" linked to their equivalence point in the figure. Also, representations can be *hierarchical*, e.g., when a frame or a set of frames play the same role like a single concept, for example when a noun is defined by detailing its features. Finally, the representation should help tackle unknowns and ambiguities, like finding the noun to which the pronoun "it" refers in the above sentence.

Note that the created structural representations are symbolic, which is similar to the symbolic nature of the human mental representations built during problem solving [9].

III. Related Work

Used often in scene understanding, recent relation identification methods use transformer models to predict relations among the objects in a scene [4], [10]. Methods are of two kinds: two-stage and one-stage methods. Two-stage methods [1], [10], [11] first employ a separate step, like a Convolutional Neural Network (CNN), to detect the objects of a scene. Then, the pairwise relations among all node pairs are learned. For example, Relation Transformer Networks (RTNs) use a transformer encoder to capture all positional node to node interactions, followed by decoding to find the object to relation interactions [10]. The learned interactions are used by a classifier to predict the likely relations among the scene objects by using the rich positional encodings of objects and object pairs. One-stage methods [4], [12], like ReITR [4], create entity encodings in their visual contexts followed by multiple decodings, like to capture the dependencies between subjects and objects, create independent subject and object representations, and connect entity and triplet detection. The decoder outputs are used in classification to find relations. Still, these methods require having ground truths objects or fixed position encodings, which can be a limitation. It can be argued that two-stage methods first "decompose" a scene into its component objects followed by "assembling" them back to learn the relations among any pair. Instead, one-stage methods learn an object's relations while decoupling it from its scene.

Relation identification is also related to the area of Spoken Language Identification (SLU), in which the intent of sentences (usually describing commands for an equipment) is found together with the filling of the command slots (parameters) with words in the sentence [15]. Even though SLU assumes a more rigid slot structure, slot filling is similar to relation finding. Single models separately address intention finding and slot filling using conditional random fields, Recurrent Neural Networks, or LSTM [13], [14]. Joint models simultaneously address intention finding and slot filling to capture their dependencies, like using transformer models [15].

Compared to the existing relation identification methods, the proposed model pursues an in-between approach, in which basic relation triplets are found (without going down to the individual entity level, like in one-stage methods), and then stitching them together (like in two-stage methods). It offers more flexibility in the structure of the sentences it processes, including hierarchical structures, enumerations, alternatives, and unknowns.

IV. Using LLMs for identifying relations

We conducted an experimental study to assess the performance of existing LLMs in automatically extracting the relations in textual descriptions of the dialog among the team members jointly solving a problem. The following LLMs were studied: GPT-4 [18], Meta [19], Gemini [20], Copilot [21], Claude [22], and Mistral [23]. Few experiments used OpenAI's new LLM, GPT-4o [18]. Table I summarizes the results.

The LLM performance was defined with respect to the following capabilities:

1. Word significance identification: How well do LLMs identify the words significant for relation representation?

2. Relation mapping: How capable are LLMs to correctly represent the relations between the significant words?

3. Contextual understanding: Do LLMs consistently handle contextual nuances, such as negations, acknowledgments, and idiomatic expressions?

4. Simplification: Can LLMs simplify more verbose expressions as simpler equivalents to enhance the clarity of the structural representations?

The experimental performance evaluation summarized in this section used a set of 15 sentences of varying complexity and linguistic features. The sentences, in textual format, are part of the dialog recorded during programming code development by teams of three members. The selected sentences included elements, like negations, acknowledgments, enumerations, alternatives, unknowns, hierarchical structures, and implications. Here are some of the used samples: "`Yeah, I don't think there's any other. I think you have to sort it`"; "`I know it gives a weird, it gives a broken output. Okay. I think it's because of this, the size`"; "`Probably do a double for loop and update it every time the smallest one gets closest`", and "`So you're manipulating the numbers to get as close as you can to the target?`". Note that the sentences were colloquial, not always grammatically correct, and used coding jargon.

Two types of prompts were used in the experimental study:

- **Pattern-based prompts**: These prompts provided clear and specific instructions to guide the LLM on how to process the sentence. By defining patterns, like subject-verb-object (SVO), an LLM was instructed to follow a particular structure when analyzing sentences. For example, a used prompt was "`Extract the subject-verb-object triplets as tuples with (word1, word2, relation)`."
- **Rule-based prompts**: These prompts were more open-ended and relied on the LLM's inherent understanding of natural language to process a sentence by leveraging its grasp of grammar and meaning. For example, a used prompt was "`Represent the given sentence as a simple graph with words as nodes and the relation between words as edges. Use significant words only`".

TABLE I
LLM PERFORMANCE ON SENTENCE REPRESENTATION TASKS

Model	Performance Metrics					
	SS	ISW	RR	SC	CU	SA
GPT-4	8.1	8.1	7.8	8.1	7.9	8.1
Meta	6.6	6.1	6.9	6.9	6.6	7.4
Gemini	7.7	7.4	7.6	7.4	7.4	7.6
Copilot	6.6	6.8	5.8	7.0	6.2	7.0
Claude	7.4	6.3	7.3	7.0	6	7.5
Mistral	7.2	7.2	6.7	7.0	6.3	6.2

Pattern-based prompts attempt a more local search, while rule-based prompts give more freedom for generalizations.

All LLM parameters were set at their default values.

LLM performance evaluation. Similar to previous work [24], experiments highlighted the critical role of prompt engineering in optimizing LLM performance. Providing explicit instructions or contextual guidance enhanced the ability to accurately extract relations, like for Gemini. This underscores the need to design task-specific prompts to guide LLMs.

Table I presents the LLM performance using the following metrics (the table columns): SS is the similarity score between human-created and LLM-generated representations, ISW describes the inclusion of significant words in the outputs, RR presents the correctness of the relation representations, SC expresses the structural clarity, CU represents the contextual understanding, and SA quantifies the simplification ability. Scores were given by comparing the structural graphs generated by LLM with the graphs manually created by human raters. The average scores were tabulated.

The results indicate that GPT-4 outperformed other LLMs across all metrics. Notably, all models exhibit their lowest performance in either relation representation (RR) or contextual understanding (CU), while achieving their highest scores in simplification ability (SA). Claude, in particular, shows strong proficiency in SA, though it faces challenges in CU.

Discussion. The results highlighted a key limitation of LLMs, particularly their difficulty to correctly extract all relations of a sentence, even though LLMs could successfully extract basic subject-verb-object (SVO) relations. The edges that link the nodes, are often ambiguous and less reliable. Depending on the context, edges between nodes can signify a range of relations, e.g., actions, identity descriptions, changes, role specifications, and property descriptions [3]. The following main limitations were encountered:

1. Neglecting stop words: Stop words, like "`and`", "`but`", "`in`", "`on`", "`at`", "`with`", "`it`" and so on, were often ignored by LLMs, as they were assumed to offer little meaningful information. However, stop words can influence the overall meaning of a sentence, like conjunctions, such as "`and`", "`but`", "`in`" etc. The exclusion of stop words can lead to loss of meaning in the generated structural representations. This points to a general limitation in ignoring words that are usually treated as fillers but may carry important meaning.

2. Inconsistent handling of acknowledgment-negation words: LLMs were unable to consider acknowledgment words, like "`yeah`", or negation words, i.e. "`no`", when the context demanded it. However, such words convey agreement, negation, or emphasis, and their ignoring produced an incomplete understanding of the text's intention and tone.

3. Difficulty in identifying nuances of word significance: LLMs sometimes struggled to identify the context-dependent importance of certain words in a sentence. For example, in sentences involving hierarchical structures or hypothetical scenarios, LLMs occasionally failed to capture the importance of particular words (e.g., the words defining the structural hierarchy, like "`that`", "`whichever`", etc.) or phrases crucial to the sentence's meaning. For instance, in the sentence "`But what if we have an extremely large number in there?`", the phrase "`in there`" was not recognized by some LLMs, even though it indicates the data source. LLMs, such as GPT-4 and Meta, demonstrated superior context awareness, while others, like Gemini, performed less reliably when processing more complex sentences.

Theoretical explanation of the LLM limitations. The mathematical model of LLMs by Frieder et al. [5] was used to theoretically explain the experimentally-observed limitations. The analysis suggested that the limitations are intrinsic to LLMs, and thus cannot be addressed only within the LLM context, like using better prompts or parameter tuning.

Frieder et al. propose that a transformer τ with L levels is described by the following mathematical relation [5]:

$$\tau = (Id + M_L^* \circ N_{M,L}^*) \circ (Id + A_L^* \circ N_{A,L}^*) \circ \ldots \\ (Id + M_1^* \circ N_{M,1}^*) \circ (Id + A_1^* \circ N_{A,1}^*) \quad (1)$$

where variable Id is the identity mapping (skip), parameters A_i^* are the self-attention maps, parameters M_i^* describe feedforward multi-layer perceptrons, and parameters $N_{M,i}^*$ and $N_{A,i}^*$ are normalization layers.

Equation (1) can be rewritten to explicitly show how the different terms representing sequences of multiplications of matrices form a transformer's output:

$$\tau = Id + \\ \sum_{i=1}^{L} M_i^* \circ N_{M,i}^* + \sum_{i=1}^{L} A_i^* \circ N_{A,i}^* + \sum_{i,j=1}^{L} M_i^* \circ N_{M,i}^* \circ M_j^* \circ N_{M,j}^* \\ + \sum_{i,j=1}^{L} M_i^* \circ N_{M,i}^* \circ A_j^* \circ N_{A,j}^* + \sum_{i,j=1}^{L} A_i^* \circ A_{A,i}^* \circ A_j^* \circ N_{A,j}^* \\ + \sum_{i,j,k=1}^{L} M_i^* \circ N_{M,i}^* \circ M_j^* \circ N_{M,j}^* \circ M_k^* \circ N_{M,k}^* + \ldots \quad (2)$$

There are two important consequences of equation (2) for the task of computationally finding the relations in a text:

1) It is hard to model cases when the word positions do not influence the resulting meaning, e.g., enumerations and alternatives, as matrix multiplication is noncommutative.
2) Normalization layers can incorrectly modify the significance of a word, i.e. by amplifying the impact of a word followed by words that are less associated with the rest of a sentence.

The two consequences are mathematical outcomes and do not necessarily reflect the meaning of a text.

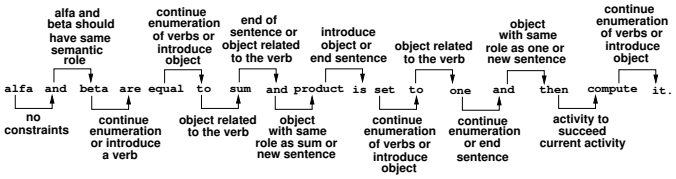

Fig. 1. The sequence of constraints on the words of a sentence.

Using the next example, we discussed the implications for text processing of equation (2), including the above limitations.

Example: Let's consider the following statement that is part of the recorded discussions on code development: "`alfa and beta are equal to sum and product is set to one and compute it`." Fig. 1 illustrates the example.

The figure shows the constraints introduced by the preceding words on the next word. For example, the word "`alfa`" does not introduce any constraints on the next word. However, the following word ("`and`") indicates that "`alfa`" and the following word should have the same role in the sentence meaning that is constructed. Similarly, the word "`beta`" and its preceding words impose the constraints that the following word should either continue the enumeration of words with the same role as "`alfa`" and "`beta`" or introduce a verb. The constraints in Fig. 1 should be accurately captured by matrices M_i^* and A_j^* and their normalization layers in equation (2).

Using the prompt "`generate a graph with nouns or pronouns as nodes and verbs as edges`", GPT-4 generated three nodes for "`alfa`", "`beta`", and "`sum`" and linked each pair using edges labeled as "`equal to`". A separate structure was created for "`product is set to 1`". While this representation was correct, adding one more variable, "`gamma`", to the enumeration of "`alfa`" and "`beta`" did not create an edge between "`alfa`" and "`gamma`", which suggests that edge (hence, relation) identification was inconsistent. To the prompt to find the triplets `subject-verb-object`, GPT-4o produced the following list: "`alfa, are equal, beta`"; "`alfa, are equal, beta`"; "`beta, are equal, gamma`"; "`product, is set to; one`; and "`sum, is set to, one`. Note that the last triplet is incorrect.

The two consequences of equation (2) created the following outcomes for the successive processing of the sentence words:

- Words are influenced by preceding words even if they pertain to different sub-sentences, hence should be separated. For example, the meaning associated to noun "`product`" is influenced by the matrices of the preceding words, even though they are in another sentence, e.g., "`beta`". Incorrect relations, like the triplet "`sum, is set to, one`, are found. Neglecting stop words created this issue.
- Correctly addressing negation words, like "`no`" and "`not`", might create structures that are dissimilar to the representations for the text without them. This dissimilarity is not captured by equation (2), and created incorrect handling of negation words.
- Swapping the words "`alfa`" and "`beta`" or the first two sub-sentences should not impact the results, however, this is against the noncommutativity of matrix multiplication.

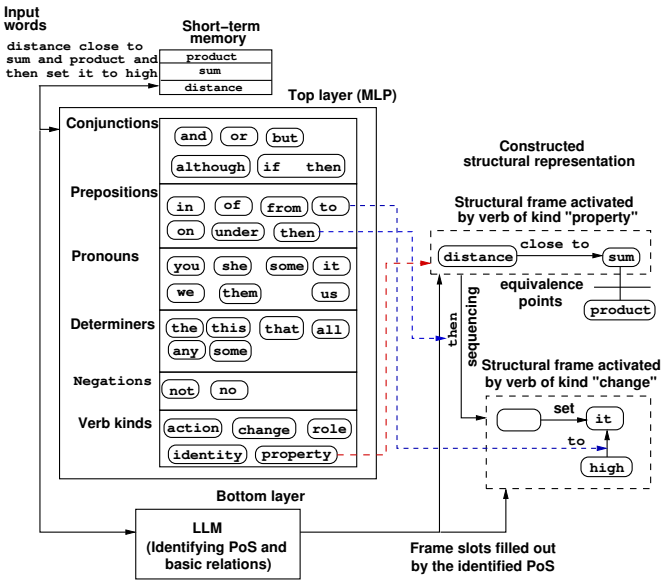

Fig. 2. Two-layered model for creating structural representations.

Mindful about the implications of equation (2), the next section discusses the proposed two-layered architecture for relation identification in text. Section VI presents its implementation.

V. Two-layered Architecture for Relation Extraction using MLP and LLM

The proposed architecture uses insight on speech processing from neuroscience, which suggests that closed class words are processed in other brain regions than nouns and verbs [6]. Closed class words have a limited and mostly invariant meaning, as they serve as grammatical or function words, like prepositions (e.g., in, on, of, at, to, from, etc.), conjunctions (i.e. and, or, but, although, if, etc.), negations (such as not, no), pronouns (i.e. you, me, she, them, it, etc.), and determiners (e.g., the, a, this, that, some, any, all). These words are usually treated as stop words by LLMs, and often ignored. Also, category-specific semantic processing is performed by the frontal, temporal, and parietal cortices to activate neural areas depending on the kind of the verbs, like verbs expressing an activity, i.e. the verb "`run`", activate the motor cortex while verbs such as "`see`" activate the visual areas [7], [16]. This insight is used as an analogy by the architecture to generate specific representation structures depending on the verb kinds.

The proposed architecture for relation extraction is a stacked, two-layered model, as shown in Fig. 2. The top layer processes the closed class words and verb kinds to generate the structural frames and structuring points of a representation while the second layer extracts the related parts of speech (PoS), i.e. nouns, verbs, adjectives, and adverbs, to populate the structural frames associated to the structuring points. Based on the experimental study in Section IV, the bottom layer uses GPT-4 to extract PoS, as LLMs were shown to be robust for this activity. The top layer is realized as a Feedforward Multilayer Perceptron (MLP) trained to generate the structuring points specific to closed class words and different verb kinds.

The trained MLP must perform the following activities: (1) segment a sentence into its composing frames, i.e. find the two frames of the input sentence in Fig. 2, (2) create the frame corresponding to the kind of the verb in the frame, (3) identify the conditions specific to the different kinds of structuring points (e.g., equivalence and alternative), and (4) associate the corresponding words and frames to the found structuring points. Note that these steps are complicated by the fact that the meaning of the encountered closed class words also depends on the following words, like the conjunction "`and`" can be part of an enumeration of nouns (which then pertain to the same equivalence point), or can separate two frames, if the conjunction is followed by a verb. Even though word disambiguation could be likely learned by MLP using an adequate training set, we decided to aid training by adding an additional label, called "`inhibition`", to denote situations, in which predicted synchronization points are inhibited because of subsequent words. For example, the initial association of a noun to an equivalence class because of "`and`" is then inhibited by the occurrence of a verb following the noun. Inhibition was inspired by the inhibition mechanism in Hebbian learning [16].

Note that Step (1) partitions equation (2), so that matrices M_i^* and A^*, j are less influenced by less related words, i.e. words in other frames. Step (3) tackles stop words, which are often ignored by LLMs.

Finally, unknowns introduced in the representation by pronouns should be addressed, like the pronoun "`it`" in Fig. 2. The pronoun can refer to any of the three preceding nouns in the sentence, "`distance`", "`sum`", or "`product`". The architecture assumes that those nouns are more likely to be associated to a pronoun, which does not produce any redundant actions. For example, associating "`it`" to "`distance`" would mean that the action of making the latter close to "`sum`" and "`product`" is immediately undone by setting its value to high. Hence, the first action becomes redundant. Then, "`it`" refers either to noun "`sum`" or "`product`".

VI. Implementation and Experimental Results

A typical MLP model was trained to classify sentences into four categories corresponding to the considered structuring points: *alternative*, *equivalence*, *inhibit*, and *none*. In addition, the MLP produced a second label, *hierarchical* or *no*, to indicate the presence or not of a hierarchical structure in a sentence. Each neuron of the input layer corresponded to the features derived from text embeddings of a sentence's words. The MLP had two hidden layers, with 500 neurons in the first hidden layer and 50 neurons in the second. The six neurons of the output layer corresponded to the six labels produced by the MLP, structuring points and presence of hierarchical structures. Softmax activation was used in the MLP model.

A. Multi-Layer Perceptron Training Data

We used a set of 3,714 recorded sentences representing the dialog of thirty teams of three members, each solving a programming exercise over a 20 minutes duration. Members were freshmen in our department enrolled in a programming fundamentals course. Speech recordings were automatically translated to text using an in-house system.

To address the high effort of sentence labeling for MLP training, the recorded sentences were labeled into four categories using GPT-4: *equivalent*, *alternative*, *inhibit*, and *none*, the types of structuring points used in our structural representations. For example, the label *equivalent* applies when actions are at the same level, indicated by words like "`and`" or "`together with`", while the label *alternative* captures choices between actions using terms, i.e. "`or`" or "`either`". The label *inhibit* describes the cases where one action overrides another, often signaled by verbs, like "`is`" or "`equals`". Otherwise, sentences were labeled as *none*. The model was also prompted to identify whether a hierarchical structure was present or not in a sentence using markers, like "`if...then`" or "`first...then`", to detect sequences or conditions.

This approach labeled 1,476 instances, about 39.8% of the total dataset. The results were manually verified. The lower labeling rate highlights the complexity of human dialogue.

There are several limitations in using LLMs for sentence labeling: The LLM's interpretation of the label meanings differed from our intended definitions, which led to mislabeling. For example, the mathematical operations, like "`subtract each of them by one`", and repeated words, like "`Okay, okay`" or "`wait, wait`" were often labeled as *equivalent* instead of *none*. Simple statements, like "`If plus a, plus b`". were labeled as *none*, despite fitting into other categories.

B. Performance Results

The MLP model's performance on relation extraction was evaluated using three different word embedding methods: TF-IDF, Word2Vec, and GloVe. Results were summarized in Table II. GloVe embeddings offered the best overall performance, achieving the highest accuracy of 0.75 and the best macro-average F1-score of 0.71. The performance improvement was notable for the class "*equivalent*", where GloVe achieved an F1-score of 0.81, higher than TF-IDF (0.75) and Word2Vec (0.77). Word2Vec embeddings showed intermediate performance, with an accuracy of 0.72 and a macro-average F1-score of 0.69. Interestingly, Word2Vec performed well on the class "*alternative*", achieving the highest F1-score (0.59) among all embedding methods. TF-IDF, despite being a simpler representation, still provided reasonable results with an accuracy of 0.69 and a macro-average F1-score of 0.62. However, it consistently underperformed, particularly for the class "*alternative*", where it had the lowest F1-score of 0.43.

The MLP performance loss was likely due to two issues: (1) Using an imbalanced training dataset (*alternative*: 139, *equivalent*: 609, *inhibit*: 430 instances) lead to lower F1-scores and poor recall for the minority class across all embeddings. (2) Overfitting occurred as shown by moderate precision but low recall for the *alternative* class across all embeddings. The superior performance on the majority *equivalent* class shows potential overemphasis of the frequent features. Having a smaller dataset (1178 instances) exacerbated the two issues.

TABLE II
MLP PERFORMANCE ON RELATION EXTRACTION CATEGORIES USING DIFFERENT EMBEDDINGS

Class	TF-IDF				Word2Vec				GloVe			
	Precision	**Recall**	**F1**	**Specificity**	**Precision**	**Recall**	**F1**	**Specificity**	**Precision**	**Recall**	**F1**	**Specificity**
Alternative	0.57	0.34	0.43	0.9654	0.55	0.63	0.59	0.9308	0.59	0.57	0.58	0.9352
Equivalent	0.73	0.76	0.75	0.7468	0.74	0.79	0.77	0.7468	0.76	0.86	0.81	0.266
Inhibit	0.66	0.71	0.68	0.9204	0.76	0.66	0.71	0.9115	0.79	0.68	0.73	0.8704
Accuracy	0.69 (n=295)				0.72 (n=295)				0.75 (n=295)			
Macro Avg.	0.65	0.60	0.62	0.8775	0.68	0.70	0.69	0.8777	0.71	0.70	0.71	0.8441
Weighted Avg.	0.68	0.69	0.68	0.8447	0.73	0.72	0.72	0.1538	0.75	0.75	0.75	0.8711
Hierarchy (No)	0.82	0.74	0.77	0.9093	0.88	0.91	0.90	0.8707	0.88	0.91	0.89	0.9035
Hierarchy (Yes)	0.84	0.90	0.87	0.7230	0.78	0.72	0.75	0.7760	0.77	0.71	0.74	0.7052
Accuracy	0.83 (n=295)				0.85 (n=742)				0.85 (n=742)			
Macro Avg.	0.83	0.82	0.82	0.8162	0.83	0.82	0.82	0.8237	0.83	0.81	0.82	0.8044
Weighted Avg.	0.83	0.83	0.83	0.8531	0.85	0.85	0.85	0.8423	0.85	0.85	0.85	0.8437

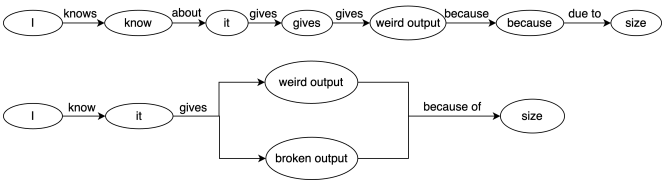

Fig. 3. Graphs generated by LLM before (top) and after (bottom)

TABLE III
LLM PERFORMANCE USING OUR APPROACH

Model	Performance Metrics					
	SS	**ISW**	**RR**	**SC**	**CU**	**SA**
GPT-4	9.2	9.4	9.1	8.8	8.4	9.1
% Improvement	13.6%	16.0%	16.7%	8.6%	6.3%	12.3%

Table III presents the results for the sentences initially analyzed sentences and summarized in Table I. The similarity to human-identified relations (SS) increased to 9.2 out of 10, for an average increase of 13.6%. The highest increase (18.2%) was obtained for the sentence "`I know it gives a weird, it gives a broken output. Okay. I think it's because of this, the size.`" All performance metrics were improved by the two-layered model. It generated more accurate and structurally sound representations, as shown in Fig. 3.

VII. CONCLUSIONS

Identifying the relations in a textual description is important for transforming the description into an engineering implementation while preserving its meaning. Mathematical and technical aspects of modern ML methods, like LLMs, e.g., mathematical properties of matrix computations, ignoring stop words, and sequential processing of tokens, produce differences between human and machine interpretation. To address this issue, this paper proposes a novel two-layered architecture for relation extraction from text corresponding to team dialog. The top layer is an MLP that processes closed class worlds, e.g., conjunctions and prepositions, and finds hierarchical structures in a sentence. The second layer, an LLM (GPT-4), identifies the basic subject - verb - object relations, which are then stitched together with the MLP outputs to create graph representations of the relations in text.

The main limitations of the work, and thus the objectives of our future work, are MLP accuracy and the features it can recognize. We plan to further analyze the sentences that are hard to classify by MLP. Also, we intend to extend the set of features it can handle, especially by adding negation, and expanding the handling of unknowns introduced by prepositions.

REFERENCES

[1] R. Zellers, M. Yatskar, S. Thomson, and Y. Choi, "Neural motifs: Scene graph parsing with global context", Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2018, pp. 5831–5840.

[2] L. Qin, T. Xie, W. Che, and T. Liu, Ting, "A Survey on Spoken Language Understanding: Recent Advances and New Frontiers" 10.48550/arXiv.2103.03095, 2021.

[3] R. Jackendoff, "Semantic Interpretation in Generative Grammar", The MIT Press, 1972.

[4] Y. Cong, M. Yang, and B. Rosenhahn, "RelTR: Relation Transformer for Scene Graph Generation", IEEE Transactions on Pattern Analysis & Machine Intelligence, vol. 45, no. 09, pp. 11169–11183, 2023.

[5] S. Frieder, J. Berner, P. Petersen, and T. Lukasiewicz, "Large Language Models for Mathematicians", arXiv, 2312.04556, 2024, https://arxiv.org/abs/2312.04556.

[6] T. Munte, B. Wieringa, and et al., "Differences in brain potentials to open and closed class words: class and frequency effects", Neuropsychologia, vol. 39, Issue 1, pp. 91–102, 2001.

[7] F. Pulvermuller, "Words in the brain's language", Behavioral and Brain Sciences, vol. 22, no. 2, pp. 253–279, 1999.

[8] Removed for blind review.

[9] M. Beni, "On the origin of mental representations", Biosystems, vol. 184, 2019.

[10] R. Koner, S. Shit, and V. Tresp, "Relation Transformer Network", arXiv, 2004.06193, 2021, https://arxiv.org/abs/2004.06193.

[11] A. Zareian, and et al., "Bridging knowledge graphs to generate scene graphs", European Conf. Comp. Vision, 2020, pp. 606–623.

[12] P. Sun, Y. Jiang, E. Xie, W. Shao, Z. Yuan, C. Wang, and P. Luo, "What makes for end-to-end object detection?", Intern. Conf. Machine Learning. PMLR, 2021, pp. 9934–9944.

[13] S. Ravuri and A. Stolcke, "Recurrent neural network and LSTM models for lexical utterance classification", Proc.Interspeech, 2015, pp. 135–139.

[14] P. Xu and R. Sarikaya, "Convolutional neural network based triangular CRF for joint intent detection and slot filling", IEEE Workshop Automatic Speech Recognition and Understanding, 2013, pp.78–83.

[15] L. Qin, T. Liu and et al., "A Co-Interactive Transformer for Joint Slot Filling and Intent Detection", Proc. ICASSP, 2021, pp.8193–8197.

[16] R. Tomasello, and el., "Brain connections of words, perceptions and actions: A neurobiological model of spatio-temporal semantic activation in the human cortex", Neuropsychologia, vol. 98, pp. 111–129, 2017.

[17] J. Liu, Y. Yang, and H. He, "Multi-level semantic representation enhancement network for relationship extraction", Neurocomputing, vol. 403, pp. 282-293, 2020.

[18] OpenAI. *GPT-4*. OpenAI, 2023.

[19] Meta AI. *Meta's LLM*. Meta Platforms, 2023.

[20] Gemini AI. *Gemini Language Model*. Gemini AI, 2023.

[21] Microsoft. *GitHub Copilot*. Microsoft, 2023.

[22] Anthropic. *Claude Language Model*. Anthropic, 2023.

[23] Mistral AI. *Mistral LLM*. Mistral AI, 2023.

[24] P. Liu and et al., "A Systematic Survey of Prompt Engineering in Large Language Models:Techniques and Applications", arXiv, 2402.07927, 2023.