Your Theory Is Wrong: Using Linguistic Frameworks for LLM Probing

Victoria Firsanova*
Department of Mathematical Linguistics
St Petersburg State University
St Petersburg, Russia 199034
st085687@student.spbu.ru

Abstract

The paper introduces a novel probing tool that uses linguistic frameworks, such as Lexical-Functional Grammar (LFG), Categorial Grammar (CG), and Head-Driven Phrase Structure Grammar (HPSG), to explore the correspondence between large language model (LLM) embeddings and formal language descriptions. The method uses LLM embeddings to construct a graph and compare it to representations generated from a linguistic framework description based on the Backus-Naur Form (BNF). By identifying intersections between the graphs, the method allows for assessing the similarity between LLM internal structure and formal linguistic theories. The findings suggest that while LLM representations do not fully correspond to any linguistic framework, they offer insights into the language structures that exceed traditional theories because LLMs' knowledge is derived from the real-world language data they are trained on. This idea questions existing linguistic theories providing new methods for verifying and refining linguistic hypotheses. The codebase for this paper is available at https://github.com/vifirsanova/llm-dmt/.

1 Introduction

Existing research on large language model (LLM) probing has largely focused on surface-level structures derived from syntax or word similarity [1, 2, 3]. Still, it remains unclear whether the internal representations of large language models (LLMs) align with linguistic categories described by established language theories, such as Lexical-Functional Grammar (LFG) [4], Categorial Grammar (CG) [5], and Head-Driven Phrase Structure Grammar (HPSG) [6]. Understanding the relationship between LLM internal structures and linguistic categories and rules declared by formal language theories could provide insights into language modeling and foundational linguistics.

The paper aims to introduce a novel tool for LLM probing based on formal linguistic frameworks. The proposed method compares the LLM embeddings with the structures described by linguistic theories, such as LFG, CG, and HPSG. To address this, the study proposes constructing two graphs: the LLM graph from the LLM embeddings and a framework-based graph from a Backus-Naur Form (BNF) description of a linguistic framework. Calculating the intersections between these graphs allows for assessing the extent to which the model's representations align with the formal structure of the theory. The approach implementation steps are as follows:

- Extract LLM embeddings
- Construct the LLM graph from LLM embeddings
- Obtain BNF description of a linguistic framework

*Alternative email address: vifirsanova@gmail.com

Interpretable AI: Past, Present and Future Workshop at NeurIPS 2024

- Convert BNF to graph representation
- Search for intersections between two graphs
- Calculate metric scores to measure similarity between two graphs

The research addresses the following questions:

- What are the criteria for choosing formal descriptions for linguistic theory modeling?
- Which formal descriptions other than BNF can be used to represent linguistic theories?
- Is it possible to describe all linguistic frameworks using formal rules?
- How to define and label edges and nodes of the graph representing LLM internal structure?

This study is a work in progress and would not answer some research questions. Further work is planned to obtain more explanations for the stated questions.

The paper describes a new method to evaluate the extent to which LLMs encode linguistic categories and structures as defined by formal language theories LFG, CG, and HPSG. The method is tested using a novel synthetic dataset introduced in this paper. This dataset is designed for LLM linguistic competence assessment, and its structure is inspired by the grammaticality judgment tests.

The study findings suggest that while LLM embeddings do not fully align with linguistic frameworks, they can provide additional insights into natural language structures. The reason for this finding is that most foundational linguistic frameworks are based on limited observational data because the humans' ability to observe and analyze data is limited and their judgments are subjective. The LLM observations are drawn from the real-world data on which they are trained. The study suggests integrating LLMs into foundational linguistic research and questions the adequacy of existing theories. In perspective, the proposed tools can be adapted to develop a novel approach to verifying and refining linguistic hypotheses through LLMs.

The paper contributions are the following:

- A novel synthetic dataset for the LLM linguistic competence assessment[2]
- A new LLM probing approach[3]
- A new tool for linguistic hypothesis verification

The study limitation is the limited scope of language phenomena; the proposed method focuses on syntactic structures and might not capture semantics, pragmatics, or discourse-level features. Another limitation is the method scalability since constructing and comparing large graphs using LLMs can be computationally expensive. Also, the provided approach does not answer why certain aspects of language are or are not captured by LLMs, focusing on the similarities between graph representations instead of their cause. The limitations highlight the prospects for future work to address these challenges.

2 Background

The study utilizes syntactic frameworks for linguistic research widely used in computational linguistics. The frameworks are Lexical-Functional Grammar (LFG), Categorial Grammar (CG), and Head-Driven Phrase Structure Grammar (HPSG). This section explains how LFG, CG, and HPSG can provide structural insights into how LLMs process language by constructing graphs for interpretability studies.

LFG represents language through two primary structures: Constituent Structure (c-structure) and Functional Structure (f-structure) [4]. To construct a graph, the c-structure can be represented as a parse tree. The tree nodes should correspond to syntactic categories, such as a noun phrase (NP) or a verb phrase (VP). The edges should denote the hierarchical relationships between them. The f-structure can be represented as a feature graph, where nodes represent grammatical functions (e.g.,

[2]The dataset is open and freely available at `https://huggingface.co/datasets/missvector/multi-wiki-grammar/`

[3]The repository with the source code is avaliable at `https://github.com/vifirsanova/llm-dmt/`

Table 1: Dataset sample

Language	Title	Grammatical Sentence	Non-Grammatical Sentence
English	School	Most countries have systems of formal education.	The countries {most}{Word Order Error} have systems of formal education.

subject, object) and their values, while edges indicate dependencies or relationships between these functions. By comparing the LFG graph with the LLM-based graph, it might be possible to indicate errors in how the model handles grammatical functions (e.g., misassigning the subject or object).

CG represents both syntax and semantics through category assignments. Each lexical item is assigned a category which determines how it can combine with other categories to form valid sentences. Each category can be a function that takes other categories as arguments [5]. In graph form, nodes can represent lexical items with their categories (e.g., N for nouns, S/N for sentence-forming elements), and edges can represent how one category applies to another to form a valid expression. Comparing the CG graph with the LLM-based graph can reveal errors in how the model handles grammatical functions (e.g., misassigning the subject or object).

HPSG represents syntactic structures through feature structures emphasizing the head of the phrase with its arguments and adjuncts [6]. The feature structures can be represented as directed acyclic graphs, where the nodes correspond to syntactic categories with associated features (e.g., number, tense, case), and edges denote the relationships between features and categories. Comparing the LFG-graph to the HPSG-graph allows for the detection and correction of grammar violations, where the LLM violates constraints, such as incorrect feature unification (e.g., mismatched subject-verb agreement).

One of the study research questions is whether it is possible to describe various linguistic frameworks using formal rules since rules allow for straightforward graph construction. For example, early transformational grammar used explicit phrase structure rules for generating deep language structures and transformational rules to map the underlying language structures onto surface structures. However, starting with the Principles and Parameters theory there were less explicit rules, which continued into the current Chomskyan minimalist framework [7, 8, 9].

While some frameworks can be represented with rules, others might use systems of constraints to describe language. Since LFG, CG, and HPSG can be described by a set of explicit rules, such approaches often find applications in computational linguistics. That is why this study focuses on these approaches and defines them in terms of graph theory for the implementation of the proposed method.

3 Data

The paper presents a novel synthetically annotated dataset. Table 1 shows the dataset sample. The dataset is available on Hugging Face at `https://huggingface.co/datasets/missvector/multi-wiki-grammar`. The dataset is designed to evaluate LLMs' sensitivity to grammatical errors across different languages. The dataset provides a resource for testing linguistic hypotheses and can be used for educational purposes. The dataset is multilingual and provides information in English, Chinese, and Russian. The dataset contains the following components:

- Language: The represented languages.
- Title: Titles of the source articles.
- Grammatical Sentence: Cleaned and truncated sentence-level segments of the source articles.
- Non-Grammatical Sentence: One or more non-grammatical variations of each grammatical sentence. The variations are generated synthetically through LLMs. The variations target specific grammatical errors. Each error is annotated with its type and position in the sentence.

The non-grammatical variations of sentences targets the following grammar violations:

- Agreement Errors: Errors in subject-verb or noun-adjective agreement.

- Word Order Errors: Incorrect word sequencing.
- Missing Articles/Particles: Omissions of articles or particles.
- Incorrect Case Usage: Mistakes in case marking.
- Improper Verb Tense: Errors in verb tense usage.

The dataset contains multilingual text data sourced from Wikipedia and academic papers shared under the Creative Commons Attribution license. The dataset statistics are the following:

- The dataset size: 5.3 GB.
- The number of source articles: around 350,000.
- The average number of grammatical sentences in each article: 100.
- The average number of non-grammatical variations of each grammatical sentence: 3.
- The sentence length range: from 50 to 100 characters.

The dataset collection process started with scraping Wikipedia articles and academic papers from online services for aggregating academic sources distributed through Creative Commons license. Each article was segmented on a sentence level. The sentences were filtered by length. The minimum sequence length was set to 50 characters, and the maximum sequence length was set to 100 characters. This simple filtering allowed for cleaning the data from noise, as well as simplify the syntax annotation task so that the model would not have to parse complex sentence with several clauses. Each sentence was fed to an LLM with the instructions to generate non-grammatical versions of the provided sentences according to a set of pre-defined error types. The annotation results were transformed to JSON objects and appended to the data frame into a specified column.

The raw texts of source articles from Wikipedia and academic sources were normalized using Unicode Normalization Form KD (NFKD). Wikipedia-specific artifacts, such as soft hyphens, accents, and bracketed content, were removed. Each processed text from the source articles was segmented at the sentence level. Sentences shorter than 50 characters and longer than 100 characters were removed to ensure manageable sentence lengths. The processed sentences were stored in the Grammatical Sentence dataset column.

For each grammatical sentence, a set of non-grammatical variations was generated using Llama 3 [10] with 8 billion parameters quantized to a 4-bit format through GGUF for efficiency. This type of the dataset annotation was inspired by grammaticality judgment tests, a type of linguistic experiments. The procedure involved generating non-grammatical versions of sentences by applying specific grammatical errors. The prompt template used in this annotation process is the following:

```
Generate Non-Grammatical sentence for the following example: <sentence>.
Consider the following set of errors: Agreement Errors, Word Order Errors,
Missing Articles/Particles, Incorrect Case Usage, Improper Verb Tense
```

Given the sentence "Most countries have systems of formal education, which is sometimes compulsory", the LLM produced its non-grammatical variation "The countries most have systems of formal education, which sometimes is compulsory for them." and the following judgments:

```
- Agreement Errors: "The countries" is a plural noun, but "have" and "is"
  should be in the plural form ("have" -> "have" and "is" -> "are").
- Word Order Errors: "Most" is incorrectly placed after the subject "the
  countries."
- Missing Articles/Particles: No error in the sentence.
- Incorrect Case Usage: The subject "the countries" is in the nominative
  case, but "them" is the objective form (used as the indirect object).
- Improper Verb Tense: The sentence is in the correct tense.
```

The generated non-grammatical sentences were added to the Non-Grammatical Sentence column of the dataset. The LLM judgments were used to annotate the synthetic sentences with generated error types and their positions in the text, for example: "The countries {most}{Word Order Error} have systems of formal education...". This annotation highlights the word "most" to indicate the error position, and marks "Word Order Error" as the error type.

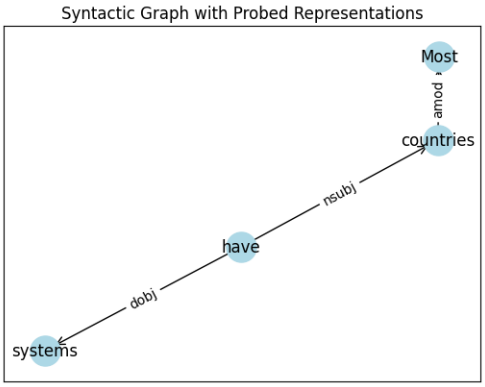

Figure 1: Sample graph structure.

The dataset limitation is that it relies on synthetically generated grammaticality judgments. For example, in the given judgment, the system indicates the Agreement Error between "The countries" and "is", which, according to the generated judgment, should be in plural. However, in the original sentence "is" is related to "which" and thus should be singular. To address this limitations, a manual review was conducted on a subset of the generated non-grammatical sentences. The review was conducted by 15 experts in linguistics and native speakers of the languages represented in the dataset.

4 Method

The proposed method compares the internal representations of an LLM with a linguistic framework described using BNF. The study uses BNF because it is a widespread solution for setting a formalism in LLM-based solutions. For example, BNF is widely used as a tool for describing a JSON structure [11]. The goal is to evaluate how well the LLM's representations align with the formal structure of the linguistic framework. The pseudocode is given in the Appendix A.

The LLM for probing extracts embeddings for a given grammatical or non-grammatical sentence from the proposed dataset. In this study, the embeddings from LLMs were extracted through loading a pre-trained AnglE model [13], encoding the input text into embeddings with a prompt "Represent this sentence for searching relevant passages: text'", and visualizing the embeddings using t-SNE [14]. The t-SNE mapping was used to construct a directed syntactic graph. This procedure follows a structural probing procedure described in [1], however, the approach described in [1] is proposed for encoder language models, while the proposed method is adapted to decoder models through applying AnglE to obtain embeddings from LLMs. Figure 1 shows a simplified graph visualization.

A linguistic framework, for example, LFG, CG, or HPSG, is described using BNF. Consider describing the LFG framework with BNF. A sentence (S) in LFG typically consists of a subject and a predicate:

```
<S> ::= <NP> <VP>
```

A verb phrase can consist of a verb followed by optional noun phrases or prepositional phrases:

```
<VP> ::= <V> | <V> <NP> | <V> <NP> <PP>
```

These rules capture the c-structure (constituency structure) in LFG. To include the f-structure (functional structure), the relationships between syntactic categories and their grammatical functions (e.g., subject, object, etc.) should be added. The sentence in c-structure consists of a subject (NP) and a predicate (VP). In f-structure, the NP corresponds to the subject (SUBJ), and the VP corresponds to the predicate (PRED):

```
<S> ::= <NP> <VP> { (↑ SUBJ) = ↓, (↑ PRED) = ↓ }
```

Here, ↑ refers to the root in the graph, and ↓ refers to the current node. The f-structure annotations (↑ SUBJ) = ↓ mean that the subject function (SUBJ) of the sentence is filled by the noun phrase (NP), and (↑ PRED) = ↓ means that the verb phrase (VP) corresponds to the predicate function.

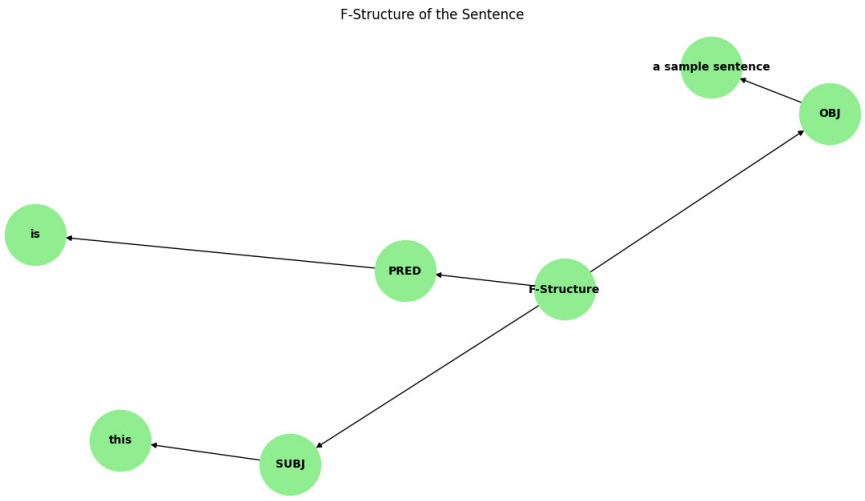

Figure 2: Sample graph structure.

The BNF description is used to construct LLM guardrails. In this study, by guardrails we mean a set of rules controling the LLM behaviour. The guardrails control the LLM output structure so that it strictly follows the BNF rules, for example, the LFG rules. In the LFG example, the guardrail should describe rules defining f-structure and c-structure according to BNF desription, validate the syntactic structure and populate the syntax graph given lexical items from the dataset. The lexical items are represented by word-level tokens of grammatical and non-grammatical sentences from the proposed dataset.

To automate the process of creating BNF descriptions, a formal description of each of the framework chosen for this study was generated through Qwen 2.5 with 72 billion parameters using prompt-engineering techniques. The prompting resulted in a list of rules generated with the model. The rules was converted into BNF using another prompt, and subsequently written to a reproducible file.

The guardrail XML schema for the LFG framework is given in the Appendix B. This guardrail outlines the rules and constraints for f-structure and c-structure in LFG including definitions and validations for syntactic rules, lexical entries, and functional mappings. Using the guardrails, the LLM populates the syntax structures defined by the framework with lexical entries from sentences from the dataset, for example:

```
<Det> ::= "the" { (↑ DET) = "the" }
<N> ::= "dog" { (↑ OBJ) = "school" }
```

The BNF structure was loaded to an LLM with a system prompt to construct a framework-based graph. For example, to visualize the LFG-based graph, the LLM was tasked to populate a graph with nodes and edges denoting the elements of the c-structure and f-structure of a given sentence. The graph was used to generate graphical representations of syntactic and functional components of a sentence for interpretability. Figure 2 shows an example of the generated visualization.

The algorithm checks for matching nodes between the LLM graph and the BNF graph. If a match is found (i.e., the nodes from both graphs represent similar concepts or structures), the intersection is recorded. The node ratio is calculated by comparing the number of intersections with the total number of nodes in both graphs. Similarly, the edge ratio is calculated by comparing the number of intersections with the total number of edges in both graphs. The overall similarity score is the average of the node and edge ratios, representing how closely the LLM's representations align with the BNF framework.

Table 2: The results obtained on grammatical and non-grammatical sentences using Llama 3 after quantization

Framework	Intersection score (grammatical)	Intersection score (non-grammatical)
LFG	0.78	0.44
CG	0.71	0.39
HPSG	0.67	0.38

5 Results and Discussion

The method was tested with Llama 3 quantized to 4-bit format through GGUF for efficiency, since the experiments were conducted in a low-resource computational setting. Both grammatical and non-grammatical sentences from the proposed dataset were used to build and compare graphs using the proposed method. Table 2 shows the evaluation results based on the average graph intersection score, where 1 indicates perfect alignment (full intersection) and 0 indicates no alignment (no intersection). These scores are based on the expected alignment between the LLM-graphs and the framework-based graphs as described in the previous section. Appendix C shows the BNF rules constructed for each framework to perform the evaluation.

The comparison of alignment between grammatical and non-grammatical sentences is necessary to assess the study significance. As expected, grammatical sentences align more closely with the structure of linguistic frameworks, as they adhere to the rules defined by these theories, while non-grammatical sentences deviating from standard linguistic rules result in a lower intersection score between graphs. LFG captures syntactic structures that LLMs are likely to represent reasonably well, leading to a high intersection. CG's emphasis on compositional semantics aligns moderately well with LLMs, so the score is slightly lower than LFG. Due to HPSG's complexity the LLM captures some aspects of it, the overall alignment is lower compared to LFG and CG.

The method evaluates how well an LLM's internal representations captured in embeddings match the formal structure of a linguistic theory. The limitation here is that different frameworks capture different types of relationships, while LLM embeddings only encode relationships based on vector similarity and those dependency classes that are pre-defined by the developer, for example, Universal Dependencies [12] annotation captured in Figure 1.

Another limitation is that the population of lexical entities requires manual labeling or scaling with LLM as described in this paper. Manual labeling requires expert work, is difficult to scale, and requires specialists with the required level of linguistic competence. Labeling with LLM solves these challenges, but the framework-based graph may be biased by LLM embeddings and the experimental results will not be accurate. However, it will still be possible to compare LLM embeddings with framework-based graphs and conduct an in-depth study of LLM interpretability.

6 Conclusion

The method presented in this study extracts embeddings from LLMs to construct and visualize a syntax graph representing the internal linguistic structure of a language model using t-SNE. Next, the method constructs directed graphs representing various linguistic frameworks through generating rules and BNF descriptions allowing for building syntactic graphs automatically from their natural language descriptions. The approach aims to capture common nodes and edges between LLM-based and framework-based graphs.

The study findings indicate that while the internal representations of Large Language Models (LLMs) do not align with established linguistic frameworks, they offer insights into language structures that extend beyond the scope of traditional theories. LLMs capture patterns that linguistic theories may not fully address due to the limitations of the data that an expert can observe to formulate the linguistic theory. LLMs provide a novel perspective that challenges linguistic theories, offering new paths for verifying and refining linguistic hypotheses. It is important to note that the presented work is still in progress, and further research is required to deepen the understanding and address the limitations identified. Future studies will focus on refining the methodologies and exploring additional linguistic frameworks to better integrate LLM insights with theoretical models.

References

[1] John Hewitt and Christopher D. Manning. 2019. A Structural Probe for Finding Syntax in Word Representations. 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), pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

[2] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What Does BERT Look at? An Analysis of BERT's Attention. In Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pages 276–286, Florence, Italy. Association for Computational Linguistics.

[3] Ethan A. Chi, John Hewitt, and Christopher D. Manning. 2020. Finding Universal Grammatical Relations in Multilingual BERT. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 5564–5577, Online. Association for Computational Linguistics.

[4] Robert C. Berwick. 1981. Computational Complexity and Lexical Functional Grammar. In 19th Annual Meeting of the Association for Computational Linguistics, pages 7–12, Stanford, California, USA. Association for Computational Linguistics.

[5] Gosse Bouma and Gertjan van Noord. 1994. Constraint-Based Categorial Grammar. In 32nd Annual Meeting of the Association for Computational Linguistics, pages 147–154, Las Cruces, New Mexico, USA. Association for Computational Linguistics.

[6] Derek Proudian and Carl Pollard. 1985. Parsing Head-Driven Phrase Structure Grammar. In 23rd Annual Meeting of the Association for Computational Linguistics, pages 167–171, Chicago, Illinois, USA. Association for Computational Linguistics.

[7] Noam Chomsky. 1991. Some Notes on Economy of Derivation and Representation. In Principles and Parameters in Comparative Grammar, ed. Robert Freidin, pages 417-454, Cambridge, MA: MIT Press.

[8] Noam Chomsky and Howard Lasnik. 1993. Principles and Parameters Theory. In Syntax: An International Handbook of Contemporary Research, eds. Joachim Jacobs, Arnim v. Stechow, Wolfgang Sternefeld and Theo Vennemann, pages 506-569. Berlin: de Gruyter.

[9] Barbara H. Partee, Alice G. ter Meulen, and Robert Wall. 2012. Mathematical methods in linguistics. Vol. 30. Springer Science & Business Media.

[10] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, et al. 2024. The llama 3 herd of models. arXiv preprint arXiv:2407.21783.

[11] Jiaye Wang. 2024. Constraining Large Language Model for Generating Computer-Parsable Content. arXiv preprint arXiv:2404.05499.

[12] Marie-Catherine De Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. 2021. Universal dependencies. Computational linguistics 47, no. 2, pages 255-308.

[13] Xianming Li, and Li Jing. 2023. Angle-optimized text embeddings. arXiv preprint arXiv:2309.12871.

[14] Laurens Van der Maaten, & Geoffrey Hinton. 2008. Visualizing data using t-SNE. Journal of machine learning research, 9(11).

A Appendix / supplemental material

The high-level pseudocode to implement the proposed method:

```
# Step 1: Extract LLM embeddings and construct the graph
# Input: text - input sentence to be processed
# Output: llm_graph - graph representation of the embeddings

embeddings = llm_model.get_embeddings(text)
llm_graph = Graph(embeddings)

# Step 2: Obtain BNF description of a linguistic framework
# Input: linguistic_framework - object representing a linguistic framework
# Output: bnf_description - BNF description of the framework

bnf_description = linguistic_framework.get_bnf_description
```

```
# Step 3: Convert BNF to graph representation
# Input: bnf_description - BNF description of the linguistic framework
# Output: bnf_graph - graph representation of the BNF description

bnf_graph = Graph(bnf_description)

# Step 4: Search for intersections between graphs
# Input: llm_graph - graph from embeddings, bnf_graph - graph from BNF
# Output: intersections - list of matching nodes between the graphs

if match_nodes(llm_graph, bnf_graph):
    intersections.append(llm_node.id, bnf_node.id)

# Step 5: Calculate metrics to assess similarity
# Input: intersections - list of matching nodes, llm_graph, bnf_graph
# Output: similarity_score - final similarity score between the two graphs

node_ratio = intersections / (llm_graph.node_count, bnf_graph.node_count)
edge_ratio = intersections / (llm_graph.edge_count, bnf_graph.edge_count)

similarity_score = (node_ratio + edge_ratio) / 2
```

B Appendix / supplemental material

The proposed guardrails for the LFG-graph construction.

```
<GrammarGuardrails>

    <!-- C-Structure Guardrails -->
    <CStructure>
        <!-- Rule Enforcement -->
        <RuleEnforcement>
            <Description>Ensure all sentences follow the c-structure
            rules defined in BNF.</Description>
            <Validation>
                <Check>Validate that the c-structure tree adheres to
                the rule: S ::= NP VP</Check>
                <Check>Ensure each node is consistent with allowed
                phrase structures.</Check>
            </Validation>
        </RuleEnforcement>

        <!-- Lexicon Integrity -->
        <LexiconIntegrity>
            <Description>Ensure all terminal symbols are correctly
            mapped to syntactic categories.</Description>
            <Validation>
                <Check>Verify that every lexical entry in the tree
                corresponds to a valid syntactic category (e.g.,
                Det, N, V).</Check>
                <Check>Ensure no lexical entry is missing from the
                lexicon.</Check>
            </Validation>
        </LexiconIntegrity>

        <!-- Tree Validity -->
        <TreeValidity>
            <Description>Ensure the c-structure forms a valid
            syntactic tree.</Description>
            <Validation>
                <Check>Ensure the tree has a single root node.</Check>
                <Check>Check for disconnected nodes or cycles
                in the tree.</Check>
            </Validation>
```

```
        </TreeValidity>
    </CStructure>

    <!-- F-Structure Guardrails -->
    <FStructure>
        <!-- Functional Annotation Consistency -->
        <FunctionalAnnotationConsistency>
            <Description>Ensure that every c-structure node has a
            corresponding functional annotation.</Description>
            <Validation>
                <Check>Verify that every NP has the correct functional
                role (SUBJ, OBJ).</Check>
                <Check>Ensure that every VP is properly linked to the
                predicate (PRED).</Check>
            </Validation>
        </FunctionalAnnotationConsistency>

        <!-- Functional Equation Validation -->
        <FunctionalEquationValidation>
            <Description>Ensure functional equations in f-structure
            are consistent and free of contradictions.</Description>
            <Validation>
                <Check>Run checks to detect contradictions in
                functional assignments.</Check>
                <Check>Validate that all functional attributes are
                correctly assigned without conflicting roles.</Check>
            </Validation>
        </FunctionalEquationValidation>

        <!-- Mapping Completeness -->
        <MappingCompleteness>
            <Description>Ensure all relevant c-structure nodes are
            properly annotated in f-structure.</Description>
            <Validation>
                <Check>Ensure every critical c-structure node has
                a corresponding f-structure role assigned.</Check>
                <Check>Review a checklist of required functional
                roles to ensure no roles are omitted.</Check>
            </Validation>
        </MappingCompleteness>
    </FStructure>

</GrammarGuardrails>
```

C Appendix / supplemental material

1.1. The examples of c-structure BNF rules for LFG

```
<S> ::= <NP> <VP>
<NP> ::= <Det> <N> | <N>
<VP> ::= <V> | <V> <NP> | <V> <NP> <PP>
<PP> ::= <P> <NP>
```

These rules define the sentence surface structure.

1.2. The examples of f-structure BNF rules for LFG

1.2.1. The sentence structure rule

```
<S> ::= <NP> <VP> { (↑ SUBJ) = ↓, (↑ PRED) = ↓ }
```

In this rule, the subject of the sentence (SUBJ) is associated with the NP (noun phrase), and the predicate (PRED) is associated with the VP (verb phrase). The arrow "↑" refers to the higher-level structure (in this case, the sentence), and "↓" refers to the current structure (the NP or VP).

1.2.2. The noun phrase rule

```
<NP> ::= <Det> <N> { (↑ SPEC) = ↓, (↑ PRED) = ↓ }
    | <N> { (↑ PRED) = ↓ }
```

In the first option, the determiner (Det) is linked to the SPEC function (specifier), and the noun (N) is linked to the PRED function. In the second option, the noun directly serves as the predicate of the noun phrase.

1.2.3. The verb phrase rule

```
<VP> ::= <V> { (↑ PRED) = ↓ }
    | <V> <NP> { (↑ PRED) = ↓, (↑ OBJ) = ↓ }
    | <V> <NP> <PP> { (↑ PRED) = ↓, (↑ OBJ) = ↓, (↑ OBL) = ↓ }
```

In the first option, the verb (V) is linked to the predicate function. In the second option, the verb is linked to the predicate, and the noun phrase is linked to the object (OBJ). In the third option, in addition to the verb and object, the prepositional phrase (PP) is linked to the oblique function (OBL).

1.2.4. The preposition phrase rule

```
<PP> ::= <P> <NP> { (↑ PRED) = ↓, (↑ OBJ) = ↓ }
```

The preposition (P) is linked to the predicate, and the noun phrase within the prepositional phrase is linked to the object of the preposition.

2. The examples of BNF rules for CG

2.1. The sentence structure rule

```
<S> ::= <NP> <VP>
```

A sentence (S) is composed of a noun phrase (NP) followed by a verb phrase (VP).

2.2. The verb phrase rule

```
<VP> ::= <V_trans> <NP>
       | <V_intrans>
```

A verb phrase can be a transitive verb (V_trans) followed by a noun phrase (NP), or an intransitive verb (V_intrans).

3. The examples of BNF rules for HPSG

3.1. The sentence structure rule

```
<S> ::= <NP> <VP> { (HEAD verb), (SUBJ <NP>) }
```

A sentence consists of a noun phrase (subject) followed by a verb phrase. The features indicate that the head of the sentence is a verb, and the subject is an NP.

3.2. The noun phrase rule

```
<NP> ::= <Det> <N> { (HEAD noun), (AGR <agreement features>) }
   | <ProperNoun> { (HEAD noun), (AGR <agreement features>) }
```

A noun phrase can be a determiner followed by a noun or a proper noun. The feature structure includes the head being a noun and agreement features like number and gender.

3.3. The verb phrase rule

```
<VP> ::= <V> <NP> { (HEAD verb), (SUBJ <NP>), (OBJ <NP>) }
   | <V> { (HEAD verb), (SUBJ <NP>) }
```

A verb phrase consists of a verb and its complements (subject and object). The head of the VP is a verb, and it may include subject and object features.

3.4. The determiner rule

```
<Det> ::= "the" | "a" { (HEAD det), (AGR <agreement features>) }
```

Determiners like "the" or "a" have agreement features.