Progress Report: Efficient, Interpretable Parsing of Unstructured Text into Structured Data Models via Semantic Tree Decomposition, Attribute Embeddings, Quadratic Programming, and Code Generation

Johannes Losert, Narutatsu Ri

November 10, 2024

Abstract

Extracting structured information from unstructured text is a fundamental challenge in natural language processing. Existing solutions, such as OpenAI's structured outputs, often rely on large-scale language models that are both computationally intensive and costly to deploy. This paper presents a novel, cost-effective methodology that replaces such approaches with a generalizable framework leveraging advanced parsing techniques, semantic tree decomposition, and optimization. Our method maps textual descriptions into structured data models, combining attribute-based embeddings with optimization techniques. We intend to benchmark our framework against OpenAI's structured outputs endpoint, with early results indicating significant cost savings and interpretability gains.

1 Introduction

The transformation of unstructured textual data into structured formats is pivotal in natural language processing (NLP), with applications across information retrieval, data mining, and knowledge base construction. Traditional solutions often rely on large-scale language models to generate structured outputs directly from text (**?**). However, these models are resource-intensive and can be costly to deploy at scale.

To explore cost-effective alternatives, we collected structured data from Credit Karma using Beautiful Soup (BS4) to create a dataset in a serialized pickle file format. This dataset, containing card information, serves as a benchmark for evaluating the effectiveness of our framework, which leverages parsing and optimization. Our technique combines semantic tree decomposition, attribute-based embeddings, and Mixed Integer Linear Programming (MILP) to map textual descriptions into predefined data models efficiently.

2 Methodology

2.1 Data Collection

To obtain a real-world dataset of credit card descriptions, we used Beautiful Soup (BS4) to scrape structured information from Credit Karma. This scraping process involved collecting detailed text on various credit cards, including terms, rewards, and fees, and storing it in a serialized pickle format. This dataset provides a basis for testing our framework's accuracy and comparing it to OpenAI's structured output capabilities.

2.2 Semantic Tree Decomposition

Given a set of sentences T describing an object, we parse the text to construct a semantic tree $\mathcal{T} = (V, E)$, where nodes V represent phrases or tokens, and edges E capture syntactic and semantic relationships. This tree captures the hierarchical structure of the text, enabling subtree extraction aligned with the information's organizational structure. We use spaCy (**?**) for dependency parsing and constituency parsing.

2.3 Embedding Generation

For each node $v_i \in V$, we compute a contextual embedding $\mathbf{e}_{v_i} \in \mathbb{R}^d$ using a transformer-based encoder, such as Sentence Transformers (**?**). Each predefined attribute A_j is represented by aggregating embeddings from example sentences, yielding an attribute embedding $\mathbf{e}_{A_j}$ that captures linguistic variability associated with A_j.

2.4 Binary Quadratic Programming Formulation

The attribute association problem was initially formulated as a Binary Quadratic Program (BQP) to capture both phrase-to-attribute similarity and intra-attribute consistency by maximizing pairwise similarity among nodes assigned to the same attribute. Let $x_{i,j} \in \{0, 1\}$ be binary variables indicating whether node v_i is assigned to attribute A_j. Our intended objective function would maximize total similarity between nodes and attributes while also clustering similar nodes within each attribute:

$$\max_{\mathbf{x}} \left(\sum_{i=1}^{N} \sum_{j=1}^{M} s_{i,j} x_{i,j} + \sum_{j=1}^{M} \sum_{i=1}^{N} \sum_{k=i+1}^{N} s^*_{i,k} x_{i,j} x_{k,j} \right), \tag{1}$$

where $s_{i,j} = \cos(\mathbf{e}_{v_i}, \mathbf{e}_{A_j})$ represents the similarity between the embedding of node v_i and the embedding of attribute A_j, and $s^*_{i,k} = \cos(\mathbf{e}_{v_i}, \mathbf{e}_{v_k})$ represents the similarity between nodes v_i and v_k when assigned to the same attribute.

2.5 Limitations and Workaround with Mixed Integer Linear Programming (MILP)

Due to limitations in the SCIP solver, which does not support quadratic constraints in its core optimization model, we restructured our BQP into a Mixed Integer Linear Programming (MILP) model. This allows us to retain phrase-to-attribute similarity terms while omitting the quadratic terms that enforced intra-attribute consistency. Consequently, our MILP objective is:

$$\max_{\mathbf{x}} \sum_{i=1}^{N} \sum_{j=1}^{M} s_{i,j} x_{i,j}, \tag{2}$$

which maximizes individual phrase-to-attribute assignments but lacks the clustering of similar nodes within each attribute as in the original BQP model. While this solution provides a feasible workaround, it sacrifices the expressive power of the quadratic objective.

2.6 Constraints

Single Assignment Constraint Each node v_i can be assigned to at most one attribute:

$$\sum_{j=1}^{M} x_{i,j} \leq 1, \quad \forall i.$$

Hierarchical Consistency Constraint To maintain a consistent hierarchical structure, if a child node v_i is assigned to attribute A_j, then its parent node $v_{p(i)}$ must also be assigned to A_j or a related parent attribute:

$$x_{i,j} \leq x_{p(i),j}, \quad \forall i, j.$$

Non-Overlapping Constraint for Sibling Attributes Nodes cannot be assigned to multiple sibling attributes. For sibling attributes A_j and A_k under the same parent attribute, we enforce:

$$x_{i,j} + x_{i,k} \leq 1, \quad \forall i.$$

Schema-Based Subtree Constraint Schema-based constraints control the number of subtrees per attribute. If the schema specifies a single instance of attribute A_j, then only one subtree in $\mathcal{T}$ may be assigned to A_j. For list-type attributes, we permit multiple non-overlapping subtrees:

$$\sum_{i \in V_j} x_{i,j} = 1, \quad \text{if single instance for } A_j,$$

where V_j is the set of nodes in all subtrees assigned to A_j. For list-type attributes, we allow:

$$\sum_{i \in V_j} x_{i,j} \geq 1.$$

3 Implementation Challenges and Current Output

3.1 Interpretability and Cost Efficiency

One of the major advantages of our approach is its interpretability. By leveraging parsed outputs with a structured hierarchy, users can more easily trace how phrases are assigned to each attribute. This approach contrasts with OpenAI's structured outputs, where model decisions are often opaque. Furthermore, because we are generating the structured data purely through Python and optimization-based parsing, this technique is much more cost-effective. There is no need for expensive API calls, as in the case with OpenAI's model-based approaches.

3.2 Next Steps: Code Generation with CodeLlama

The next step in our pipeline is to employ a model like CodeLlama to automatically generate Python code that can parse the structured JSON objects output by our framework. This process will enable end-to-end automation, where unstructured text is parsed, structured, and processed through custom code for further data analysis or integration. Code generation via CodeLlama provides additional cost savings, as the generated code can be reused and fine-tuned to accommodate different data processing needs without repeatedly invoking large models.

3.3 Output Quality and Duplication

The current implementation faces two primary issues:

- **Sparse Sub-attributes**: Due to the lack of intra-attribute similarity clustering, sub-attributes remain inconsistently populated, even when relevant phrases exist.
- **Duplication of Phrases**: Minor variations in phrasing result in repeated entries across attributes, reducing interpretability.

3.4 Future Improvements

To address these issues, we plan to:

- **Restore Quadratic Constraints**: Experiment with solvers or approximation techniques that support quadratic constraints, allowing us to reintroduce intra-attribute clustering for better alignment with the original BQP intent.
- **Enhanced Embedding Consistency**: Use clustering techniques to ensure phrases are more consistently assigned to relevant attributes.
- **Benchmarking Against OpenAI**: Using the scraped Credit Karma dataset, we plan to benchmark our structured data generation pipeline against OpenAI's structured outputs endpoint. This comparison will focus on both interpretability and cost-efficiency to assess the advantages of our approach.

4 Conclusion

This paper presents a linearized solution for parsing unstructured text into structured data models, a necessary adjustment to accommodate SCIP's limitations. This linear solution provides an efficient and feasible framework, while our planned reintroduction of quadratic constraints in the objective is anticipated to yield a higher-quality structured output that better captures complex attribute relationships. Additionally, by employing CodeLlama for code generation to handle structured JSON outputs, our approach promises to remain highly interpretable and cost-effective.