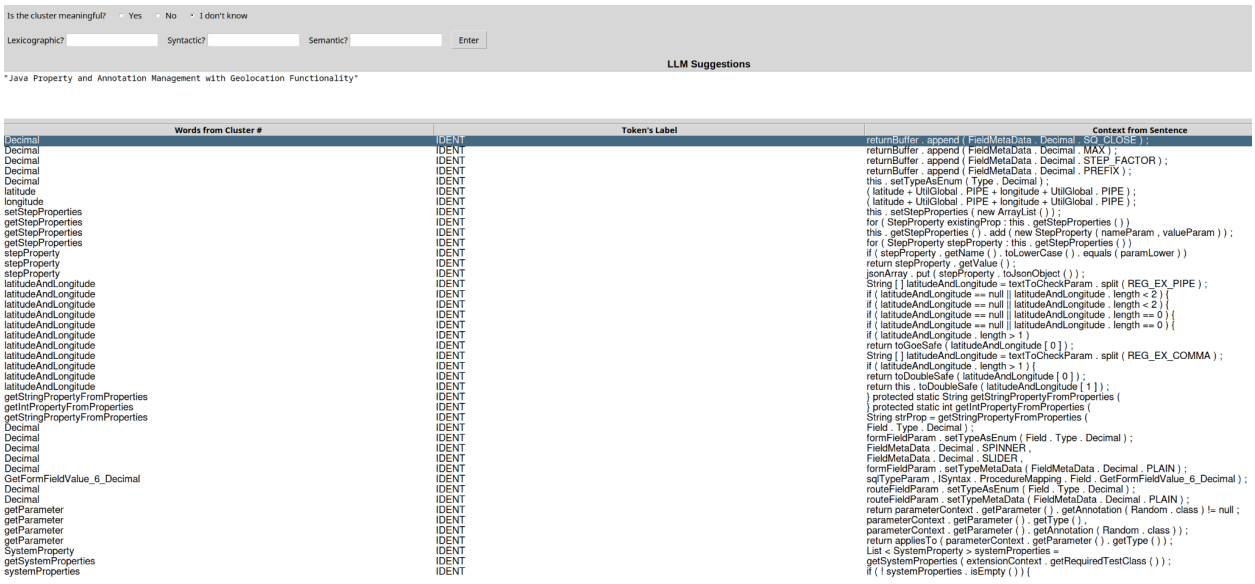
Annotation Instructions Tutorial

General Instructions:

A word cluster is represented in a form of a word cloud, where the frequency of the word in the data represents a relative size of the word in the word cloud. To understand the context of each word in the cluster it is important to hover over the a word, which will show the associated sentences for that word below.



1.

The annotation task consists of first looking at a group of code tokens (i.e., representing as a group or cluster) and answering the following questions.

1. Is the cluster or word group meaningful?
    a. **Yes:** if it represents a meaningful cluster.
    b. **No:** if it does not represent any meaningful cluster.
    c. **Don't know or can't judge:** if it does not have enough information to make a judgment. It is recommended to

categorize the word groups using this label when the word group is not understandable at all.

2. **Lexical labels**: patterns related to naming of tokens in cluster. Ex: all tokens end with “obj” or all tokens have common substring
3. **Syntactic labels**: tokens that are a part of the abstract syntax tree (also provided using .label file.)
4. **Semantic labels:** patterns based on context in which tokens are provided. The context sentences have been provided within the annotation tool for reference.

Multiple Concepts within cluster:

1. If the words can belong to multiple concepts, assign both concepts (for example, identifiers, numbers).
2. Label them in order of frequency. If order of frequency is unclear, put them down in a random order.
3. If there are more than three, end it with ‘etc.’ If there are too many, name it as miscellaneous if there is a theme, for example miscellaneous identifiers, or leave the field blank.

References:

https://www.baeldung.com/cs/lexicon-vs-syntax-vs-semantics

Resolving ambiguities

<Add examples of ambiguous cases here>

1. Add syntactic labels using the suggestions (from the .label file) provided. Add the syntactic label first followed by additional observable concepts.
2. If words are all identifiers with no common theme : miscellaneous identifiers.
3. Multiple concepts in a cluster: we label them in order of frequency, etc.
    a. Example: Meaningful: Yes. Theme: miscellaneous identifiers. Cluster Concepts: 90% tokens exhibit a particular, identifiers with “Object” in name, etc.
    b. DateTime bool
        i. Meaningful: <yes, No, IDK> eg. Yes.
        ii. Theme: < description> < syntactic label> eg. miscellaneous identifiers .
        iii. Cluster Concepts:<brief contextual description> DateTime, bool
    c. PUT, 404, ERROR_CODE_OTHER, getRef
        i. Meaningful: yes. Theme: miscellaneous Identifiers, number. HTTP requests, Error codes.
4. Themes:

    Syntactic labels:

```
dictionary = {"::": "DOUBLECOLON","--":"DOUBLEMINUS","++":"DOUBLEPLUS","false":"BOOL","true":"BOOL","Modifier":"MODIFIER",
          'BasicType":"TYPE", "null":"IDENT","Keyword": "KEYWORD", "Identifier": "IDENT","DecimalInteger":"NUMBER","DecimalFloatingPoint":"NUMBER",
            "String":"STRING", "(": "LPAR", ")": "RPAR","[":"LSQB", "]":"RSQB",",":"COMMA", "?":"CONDITIONOP",
                ";":"SEMI","+":"PLUS","-":"MINUS","*":"STAR","/":"SLASH", ".": "DOT",  "=": "EQUAL",":": "COLON",
                "|":"VBAR","&":"AMPER", "<":"LESS",">":"GREATER","%":"PERCENT","{":"LBRACE","}":"RBRACE",
                 "==":"EQEQUAL","!=":"NOTEQUAL","<=":"LESSEQUAL",">=":"GREATEREQUAL", "~":"TILDE","^":"CIRCUMFLEX",
                 "<<":"LEFTSHIFT",">>":"RIGHTSHIFT", "**":"DOUBLESTAR","+=":"PLUSEUQAL","-=":"MINEQUAL","*=":"STAREQUAL",
                 "/=":"SLASHEQUAL","%=":"PERCENTEQUAL","&=":"AMPEREQUAL","|=":"VBAREQUAL","^=":"CIRCUMFLEXEQUAL",
                 "<<=":"LEFTSHIFTEQUAL",">>=":"RIGHTSHIFTEQUAL","**=":"DOUBLESTAREQUAL","//":"DOUBLESLASH","//=":"DOUBLESLASHEQUAL",
                 "@":"AT","@=":"ATEQUAL","->":"RARROW","...":"ELLIPSIS",":=":"COLONEQUAL","&&":"AND","!":"NOT","||":"OR"}
```

    If mixture of the above, use “miscellaneous syntactic characters” as theme

a. Object: If many tokens are related to objects (their creation or use)
    i. I.e. “new” keyword, object names, object types, etc.
b. Function: If many tokens are related to functions (function calls, return values, etc.)
    i. I.e function return statements, function titles, function calls, etc.
c. ENUM: If identifiers are all CAPS
    i. FIELD_VALUE
d. Syntactic: many tokens contain similar punctuation hcharacter
e. Lexicographic: similarities in the actual words
    i. Ex: all tokens end with “obj” or all tokens have common substring.
    ii. Pascal Case naming {or any other type of naming}
f. Semantic: similarities in the context of the token
    i. NOTE: this is different from cluster context because cluster context deals with other patterns in the sentence the token is used in not directly associated with the token
    ii. Ex. a semantic description could be all tokens are function calls, but the cluster context can include patterns outside of the token like if all of the functions are being called on a certain object
g.

5. Descriptions:
   Casting, Conditionals, Instantiation, Ternary Operator, function call, Mapping, Stringbuilder, Empty Method, Constructor, Argument, Instance Reference, parameters, Data, assignment, Accessor, Network, Error Handling,

Exception Handling, Lambda Function, Loops, Generics, Form Management, null assignment, Concatenation, setter, getter, opening parenthesis, closing parenthesis, return values, Encryption, user Management. Function Definition, Authentication

6. Classifications:
    a. LEXOGRAPHIC
        i. Any common substrings found in the tokens
    b. SYNTACTIC (words to use):
        i. Object
        ii. Variable
        iii. dataType Keyword [things like “double” “int” “Date” …]
        iv. Method name
    c. SEMANTIC (words to use):
        i. Casting, Conditionals, Instantiation, Ternary Operator, function call, Mapping, Stringbuilder, Empty Method, Constructor, Argument, Instance Reference, parameters, Data, assignment, Accessor, Network (things like HTTP), Error Handling, Exception Handling, Lambda Function, Loops, Generics, Form Management, null assignment, Concatenation, setter, getter, opening parenthesis, closing parenthesis, return values, Encryption, user Management. Function Definition, Authentication, configuration (things like “bean” and “proxy”)

Examples:

Meaningful: Yes. Theme: Lexicographic: Obj. Cluster Concepts: Instantiation.

Please see the detailed instructions for each example below.

Detailed Instructions:

A word group is meaningful if it contains semantically, syntactically}, or lexically similar words. <Add example>. The labels for this question can be one of the following:

Labels:

1. **Yes:** if it represents a meaningful cluster.
2. **No:** if it does not represent any meaningful cluster.
3. **Don't know or can't judge:** if it does not have enough information to make a judgment. It is recommended to categorize the word groups using this label when the word group is not understandable at all.

Syntactic: Semantic: Function/Usage: Common theme(naming, Syntactic Character)

Example: Code4ML dataset https://zenodo.org/records/6607065

Description:

Meaningful
Lexical: Naming Object in name
Syntactic: identifier
Semantic: context

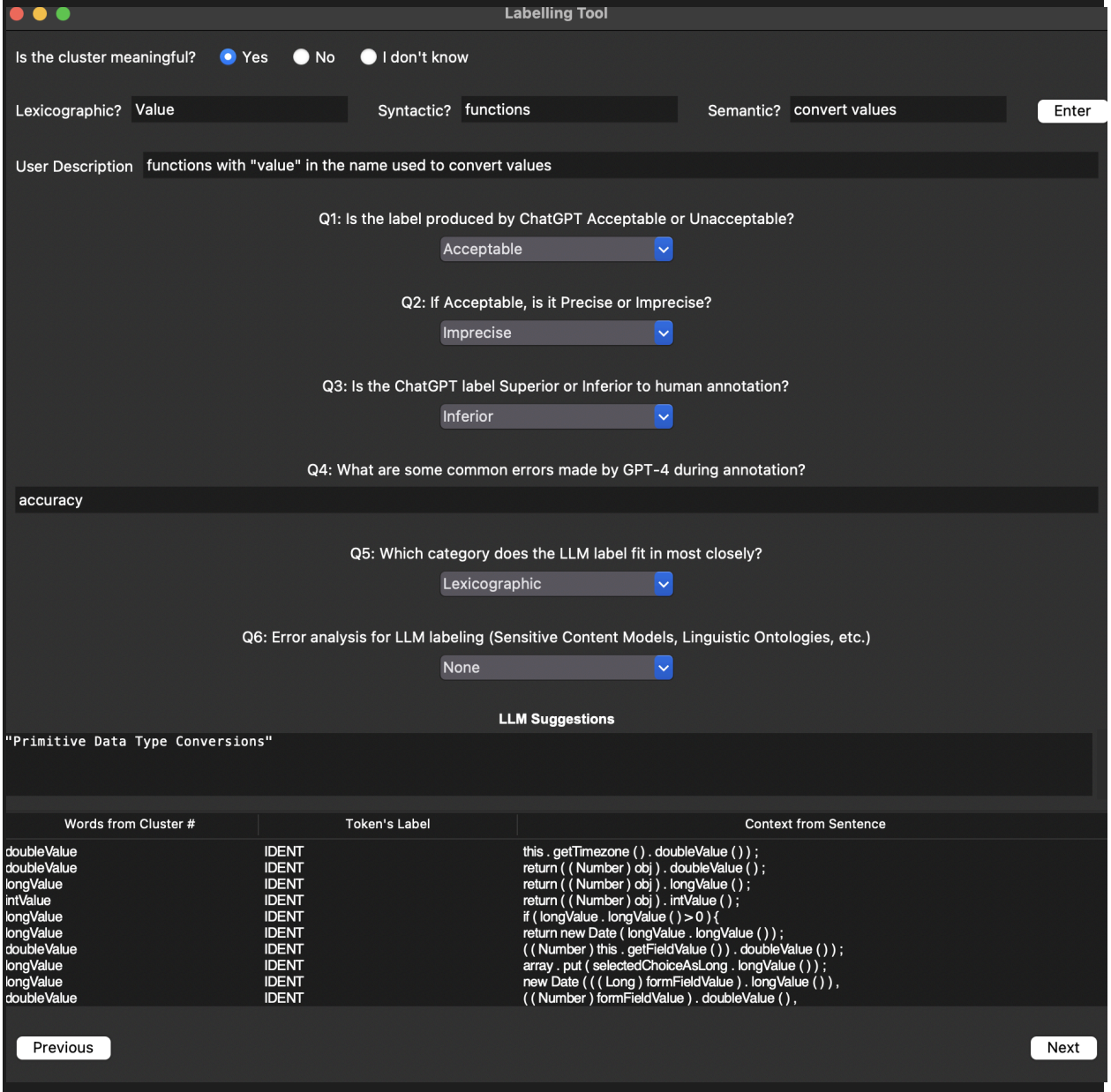
Theme: <lexical and/or syntactic> similarity in naming, syntactic labels eg identifiers(syntactic) with ‘Object’ in name(lexical).
Cluster Concept: <semantic> any patterns related to context
Reference: https://www.baeldung.com/cs/lexicon-vs-syntax-vs-semantics

Considerations:
Is LLM label suitable?
Does this cluster contain a composition of concepts (i.e multiple concepts)?
Should we make separate categories syntactic labels(we can probably just get these from the label file) and other categories:
Confidence scale - Meaningful cluster question.

Things to possibly update again:
>”function” and “method” were used interchangeably (standardize)
>add “operator” after the dots in syntactic

```
# Q1: Acceptable or Unacceptable
q1_label = tk.Label(research_frame, text="Q1: Is the label produced by ChatGPT
Acceptable or Unacceptable?")
q1_label.pack(side=tk.TOP, padx=10, pady=(10, 2))

q1_answer = tk.StringVar(value="Unanswered")  # Default value
q1_entry = ttk.Combobox(research_frame, textvariable=q1_answer, values=["Acceptable",
"Unacceptable"])
q1_entry.pack(side=tk.TOP, padx=10, pady=(0, 10))

# Q2: Precise or Imprecise
q2_label = tk.Label(research_frame, text="Q2: If Acceptable, is it Precise or
Imprecise?")
q2_label.pack(side=tk.TOP, padx=10, pady=(10, 2))

q2_answer = tk.StringVar(value="Unanswered")  # Default value
q2_entry = ttk.Combobox(research_frame, textvariable=q2_answer, values=["Precise",
"Imprecise"])
q2_entry.pack(side=tk.TOP, padx=10, pady=(0, 10))

# Q3: Superior or Inferior
```

```
q3_label = tk.Label(research_frame, text="Q3: Is the ChatGPT label Superior or
Inferior to human annotation?")
q3_label.pack(side=tk.TOP, padx=10, pady=(10, 2))

q3_answer = tk.StringVar(value="Unanswered")  # Default value
q3_entry = ttk.Combobox(research_frame, textvariable=q3_answer, values=["Superior",
"Inferior"])
q3_entry.pack(side=tk.TOP, padx=10, pady=(0, 10))
```




```
# Q4: Common Errors by GPT-4
q4_label = tk.Label(research_frame, text="Q4: What are some common errors made by
GPT-4 during annotation?")
```

```
q4_label.pack(side=tk.TOP, padx=10, pady=(10, 2))


q4_entry = tk.Entry(research_frame)
q4_entry.pack(fill=tk.X, padx=10, pady=(0, 10))
```

5) Which category(lexicographic, syntactic, semantic or descriptive) does the LLM label fit in most closely?

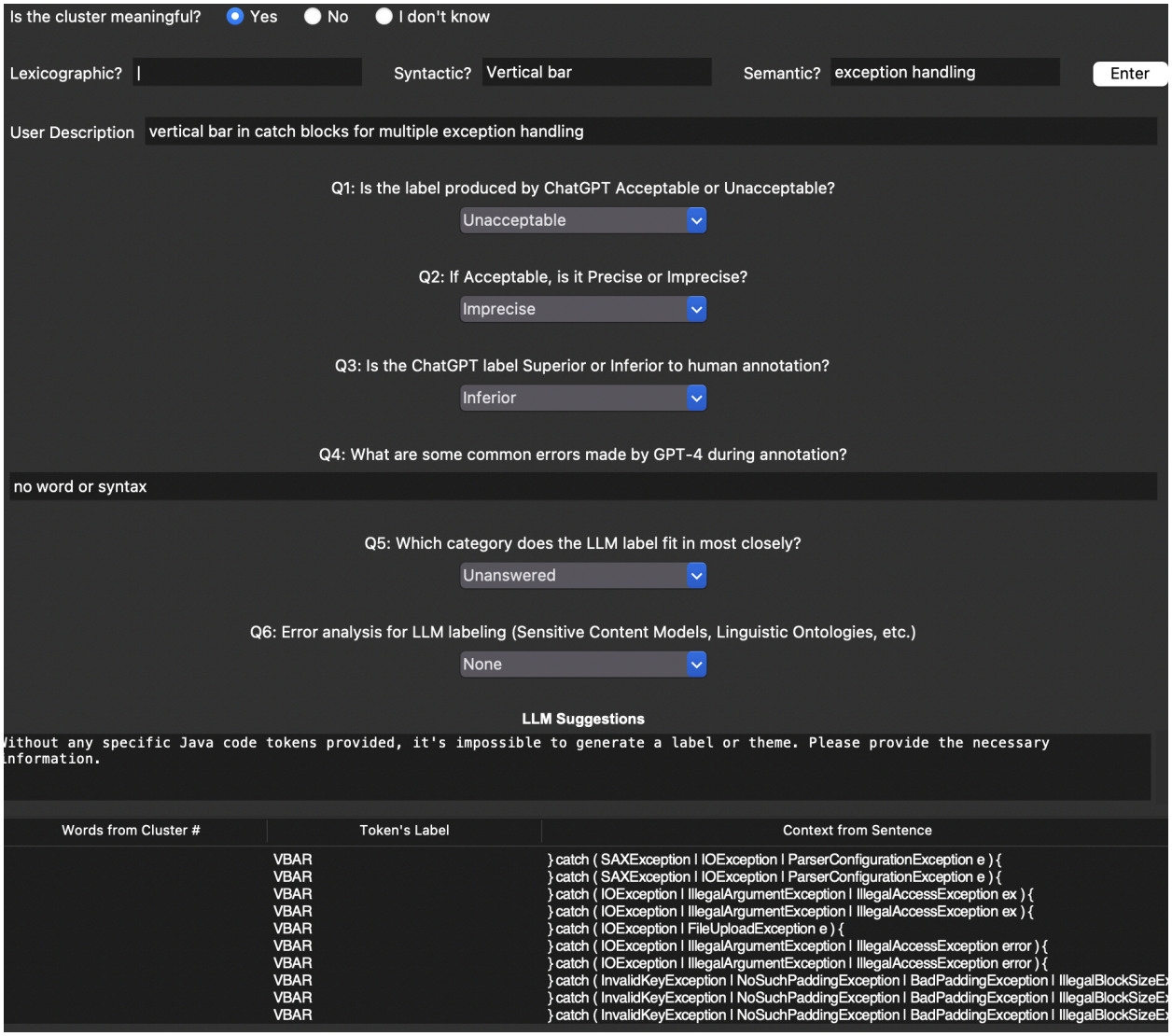
Acceptable : if description fits some aspect of the cluster correctly
Precise: if it's to the point and not overly vague, no extra information
Superior: based on precision and accuracy
Descriptive: it could be a description of the error that the LLM has while labeling.

6) Error analysis for LLM labeling.(Sensitive Content Models,Linguistic Ontologies, Insufficient Context,Uninterpretable Concepts, None)

Error Analysis

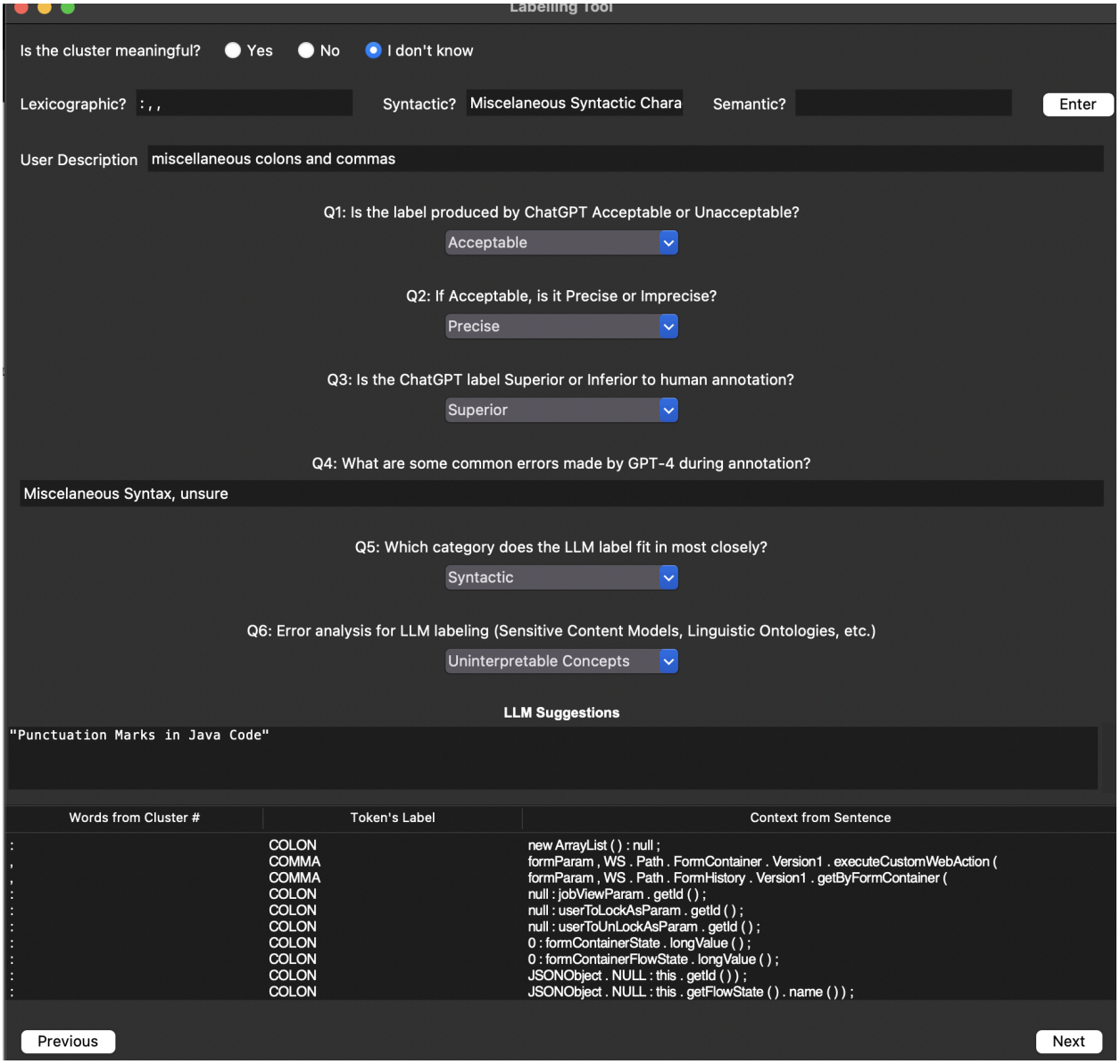
After completing the annotation task, we looked into the errors that occurred while matching the label provided by LLM and human annotation.

Insufficient context: The context sentence needs to be more comprehensive; it is difficult to determine the label correctly. In such cases, the GPT must be supplied with sufficient context or words to decide on a correct label, providing an incorrect token label.



In this case, the '|' token was not provided. As a result, the GPT was unable to provide the correct token.

Uninterpretable concepts: In instances where the cluster or concepts are difficult for us annotators to understand, the GPT sometimes generates accurate labels. For instance, even when the cluster was incomprehensible to human annotators and labeled it miscellaneous, the GPT managed to produce accurate labels such as commas, and colon.



Precision error:

In this example, it is evident that the methods intended for value conversion are being inaccurately categorized as identifiers by GPT. This mislabeling is causing confusion across a wide range of representations. The issue at hand pertains to the lack of precision in the token labels.