

Detecting Parts of Speech and Grammatical Mistakes by Using Parsing Rules

Amirmahdi Ansaripour (Student ID: 810198358)

1 Introduction

English grammar follows precise structures which can be written as production (parsing) rules. Computers can distinguish between sentence structures if they are written as parsing rules. In this article, we propose **GrammarEditor**, a parser which detects **parts of speech of the words** in each sentence of an input text. GrammarEditor also finds **grammatical mistakes** through establishing **semantic actions** between nodes of the output parsing tree.

2 The first task: Finding part of speech

2.1 Grammar files

For this task, some .g4 (grammar) files are developed:

- **EnglishGrammar.g4**: In this file, different kinds of sentences which may appear in a paragraph are coded. Some structures developed are: the **basic sentence structure**, **compound sentence**, **relativeClause**, and **infinitivePhrase**.

```
43 sentence [int line, int index] returns [Sentence ret, int indexRet]
44 :
45 {
46     $ret = new Sentence($line);
47     $ret.setIndex($index);
48 }
49 [endpoint {$ret.capitalize();} | COMMA SPACE conjunction[$ret]]
50 (NEWLINE {$indexRet = $line + 1;} | SPACE {$indexRet = $line;})+
51 {$ret.changeLine($indexRet);}
52 sentenceStructure[$ret]
53 ;
54
```

Figure 1: the **sentence** rule in **EnglishGrammar.g4**

- **Present.g4**: In this file, the detailed components of each structure are handled. For instance, the rule **sentenceStructure** is made up of **subject + adverb + arbitrary parts** (objects and adverbs). Another example is **relativeClauseStructure**, which starts with a **relativePronoun** and is followed by a **verb** or a **subject**.

```

53 : sentenceStructure [Sentence s]
54 :
55 :   (subject[$s, $s.isCapital()] SPACE verb[$s])
56 :   arbitraryParts[$s]
57 : ;
58
59 : arbitraryParts [Sentence s]
60 :
61 :   {String object = "";}
62 :   {
63 :     (SPACE object[$s, false, null] SPACE preposition SPACE {object += ($preposition.ret + " ")};
64 :     (preposition {object += ($preposition.ret + " ")} SPACE)? object[$s, false, object])
65 :     | // I'll send a letter to him
66 :     (SPACE object[$s, false, null] SPACE object[$s, false, null])
67 :     | // I'll send him a letter
68 :     (SPACE object[$s, false, null]) // I'll send a letter
69 :   )?
70 :   (SPACE adverb[$s, false])*
71 : ;

```

Figure 2: the **sentenceStructure** rule in **Present.g4**

- **PartOfSpeech.g4**: Labels that a **verb** gets (**modal**, **present perfect**, **to be**), different types of **adverbs** (time, place, state), and **nounPhrases** are implemented here.

```

27 : object [Sentence s, Boolean cap, String objecct] returns [Object obj]
28 :
29 :   nounPhrase[s]
30 :   {
31 :     if(objecct != null) objecct += $nounPhrase.ret;
32 :     else objecct = $nounPhrase.ret;
33 :     $obj = new Object(objecct, cap, $s.getLine());
34 :     $s.setObject($obj);
35 :   }
36 :   (relativeClauseStructure[s])?
37 : ;

```

Figure 3: the **object** rule in **PartOfSpeech.g4**

- **Lexer.g4**: The most basic file containing definitions. For instance, **MODAL**: (will | should | must | would | ...), or **SEASON**: (spring | summer | fall | winter).

```

PREPOSITION: 'in' | 'In' | 'at' | 'At' | 'on' | 'On' | 'next to' | 'Next to' |
'to' | 'To' | 'into' | 'into' | 'by' | 'By' | 'with' | 'With' | 'of' | 'for' |
'since' | 'Since';

IDENTIFIER: 'a' | 'the' | 'A' | 'an' | 'An' | 'The' | 'some' | 'Some' | 'Most' | 'most' | 'Few' | 'few' | 'A few' | 'a few' |
'little' | 'Little' | 'A little' | 'a little' | 'a lot of' | 'More' | 'more' | 'That' | 'that' | 'those' | 'Those' | 'this' |
'This' | 'these' | 'These' | 'my' | 'My' | 'your' | 'Your' | 'his' | 'His' | 'her' | 'her' | 'our' | 'Our' | 'other' |
'Their' | 'their' | 'mine' | 'Mine' | 'Yours' | 'yours' | 'ours' | 'Ours' | 'theirs' | 'Theirs' | 'one' | 'two';

```

Figure 4: identifiers and prepositions in **Lexer.g4**

2.2 Sentence structures

In this section, sentence structures and how they are parsed by GrammarEditor are discussed.

2.2.1 Simple sentences

Simple sentences follow this structure: (**subject + verb + object + adverb**). A sentence may have zero, one, or two objects (**direct and indirect object**). Also, a sentence may

take **arbitrary number of adverbs**.

I have studied a lot to become successful.

I was running in the park.

The apple has been eaten.

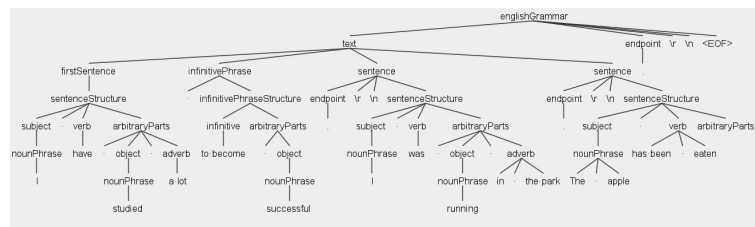


Figure 5: the parsing tree of the example

Another example:

I gave him a notebook, some pens, an eraser, and a sharpener in the classroom. He thanked me a lot.

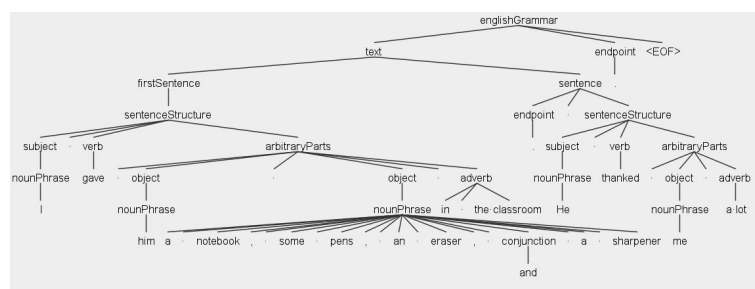


Figure 6: the parsing tree of a simple sentence having more than one object

2.2.2 Infinitive phrases

There is a subtle point we should consider when using '**to**'. It can be placed as a **preposition** before adverbs or objects. In addition, it can be used when using an infinitive after a verb. The important caution is that the verb following '**to**' in the second case should always be in **simple form**.

I told him to drive the car to the ceremony. I told him not to drive fast.

2.2.3 Compound sentences

In this type of sentence, two (or more) **independent sentences** are connected by **conjunctions** (like **and**, **but**, **so**, etc). Since we have already implemented complete sentences, it is not difficult to implement compound sentences.

I could not run anymore, but my competitor won the marathon, and he became the champion.

I did not study hard, so I will not achieve a good mark.

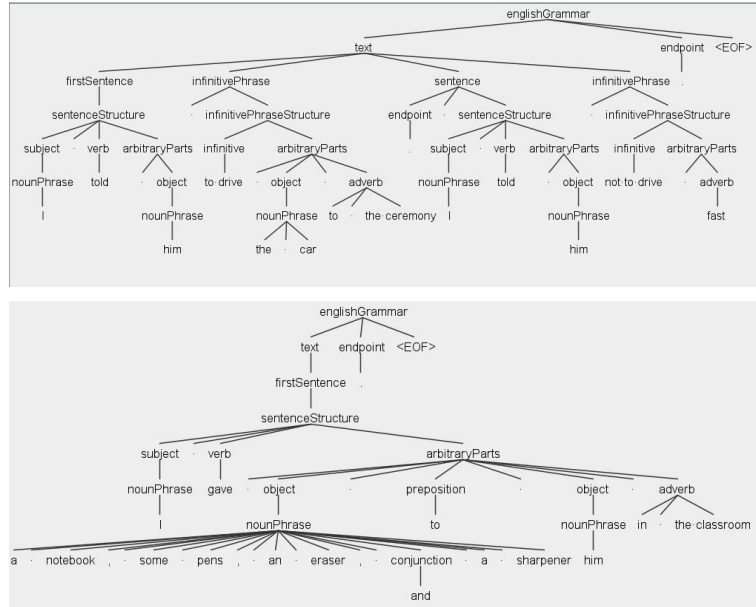


Figure 7: the parsing tree of a sentence containing infinitives

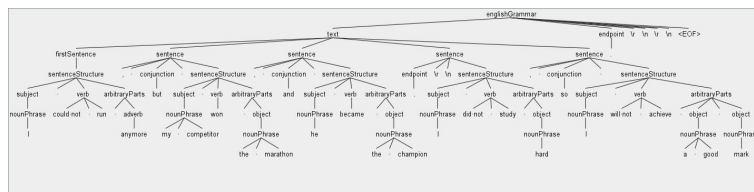


Figure 8: An example of the parsing tree of compound sentences

2.2.4 Relative clauses

Relative clauses have the responsibility of giving more information about the nouns after which they come. They act like adjectives, and one part of speech is assigned to the whole phrase (adjective clauses). Consider the following examples (Figure 9):

I saw the man, his friends, and his family, who went to the park yesterday
(object)

I visited the desert, where I have not visited before.
(object)

The man whose brother died cried a lot in the ceremony.
(subject)

3 The second task: Finding grammatical mistakes

3.1 Architecture

As we saw in the previous section, a **parsing tree** whose nodes represent different words' parts of speech has been obtained. In order to do more analysis on the nodes, we should define **semantic actions** just after defining each node. Consider the semantic actions of **subject** and **object** rules (Figure 10):

It is also worthwhile to take a look at the **Sentence** class. It has some attributes and methods which can help discover grammatical or dictation errors. For example, we can check the **subject-verb agreement** in each sentence, or we can check the **tense-agreement** between verb and adverbs (Figure 11).

When all words in sentences have gotten their parts of speech, the **Analyzer** class starts working. Its duty is to call the overridden method **visit** for each sentence and word (more generally, each node of the output parsing tree). As a result, a list of errors are collected. If a sentence does not have any mistakes, **ASTTree** prints the words' parts of speech. Otherwise, **ErrorPrinter** prints the errors.

3.2 Grammatical mistakes

In this section, some errors handled by GrammarEditor are discussed.

3.2.1 Subject-Verb agreement

Consider the following wrong sentences: (Figure 13)

A few people is going to park tonight.

That boys is playing football a lot.

My friend and his father have fixing their car.

He should saves his time and money.

The poor needs more attention.

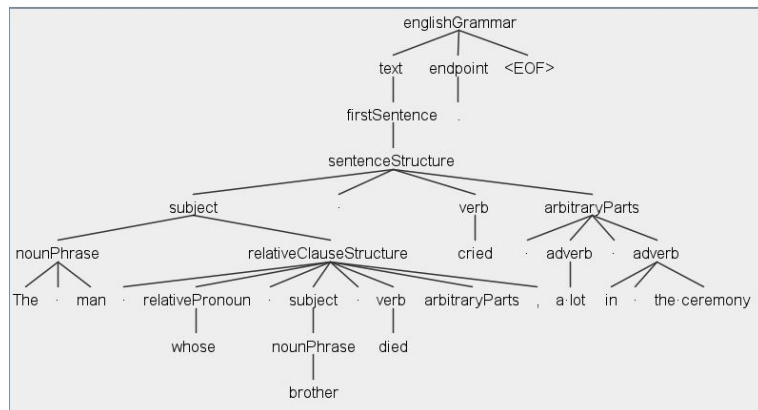
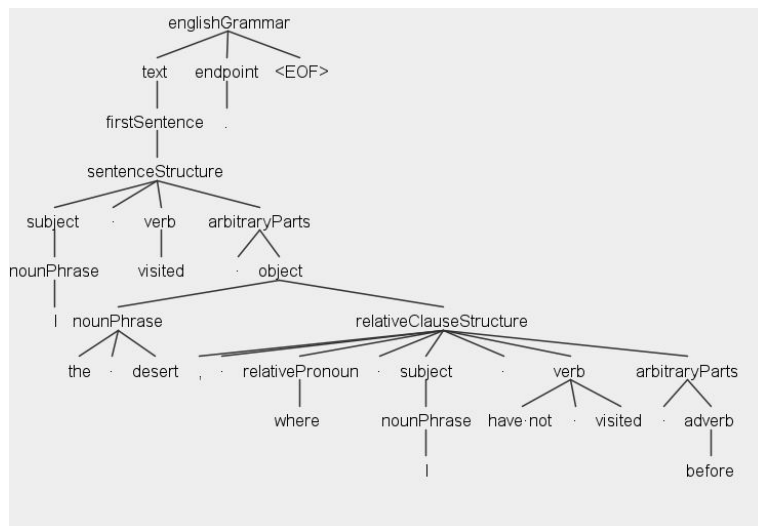
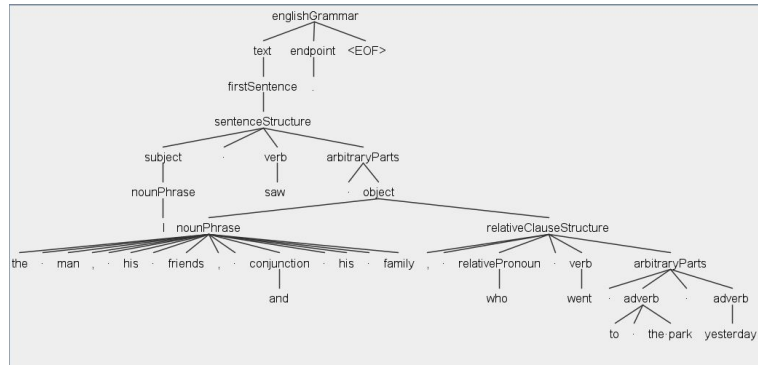


Figure 9: The parsing tree of sentences having relative clauses

```

subject [Sentence s, Boolean cap] returns [Subject sub]
:
    nounPhrase[s]
    {
        $sub = new Subject($nounPhrase.ret, cap, $.getline());
        $.setSubject($sub);
    }
    (relativeClauseStructure[s])?
;

object [Sentence s, Boolean cap, String objectt] returns [Object obj]
:
    nounPhrase[s]
    {
        if(objectt != null) objectt += $nounPhrase.ret;
        else objectt = $nounPhrase.ret;
        $obj = new Object(objectt, cap, $.getline());
        $.setObject($obj);
    }
    (relativeClauseStructure[s])?
;

```

Figure 10: the **subject** and **object** parsing rules and their respective semantic actions

```

public class Sentence extends astNode {
    private int index;
    private Subject subject;
    private Object object;
    private final ArrayList<Adverb> adverb;
    private Verb verb;
    private final ArrayList<ArrayList<String>> words;
    private Boolean capital;
    public Sentence(int line_){
        capital = false;
        words = new ArrayList<ArrayList<String>>();
        adverb = new ArrayList<Adverb>();
        line = line_;
    }
    public void setIndex(int index){this.index = index;}
    public int getIndex(){return this.index;}
    public void setSubject(Subject subject){

```

Figure 11: the **Sentence** class and its corresponding attributes and methods

```

public class GrammarCompiler {
    public void compile(CharStream programName){
        EnglishGrammarLexer lexer = new EnglishGrammarLexer(programName);
        CommonTokenStream tokenStream = new CommonTokenStream(lexer);
        EnglishGrammarParser parser = new EnglishGrammarParser(tokenStream);

        Text text = parser.englishGrammar().whole;
        Analyzer analyzer = new Analyzer(text);
        text.accept(analyzer);
        ErrorPrinter errorPrinter = new ErrorPrinter();
        text.accept(errorPrinter);
        ASTTree astTreePrinter = new ASTTree();
        text.accept(astTreePrinter);
    }
}

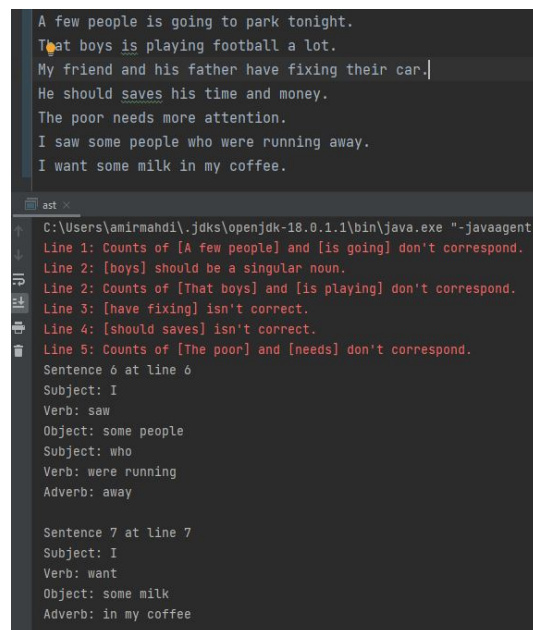
```

Figure 12: the general architecture of GrammarEditor

Also pay attention to these correct sentences:

I saw some people who were running away.

I want some milk in my coffee.



```
A few people is going to park tonight.
That boys is playing football a lot.
My friend and his father have fixing their car.
He should saves his time and money.
The poor needs more attention.
I saw some people who were running away.
I want some milk in my coffee.
```

```
ast
C:\Users\amirmahdi\.jdk\openjdk-18.0.1.1\bin\java.exe "-javaagent:
Line 1: Counts of [A few people] and [is going] don't correspond.
Line 2: [boys] should be a singular noun.
Line 2: Counts of [That boys] and [is playing] don't correspond.
Line 3: [have fixing] isn't correct.
Line 4: [should saves] isn't correct.
Line 5: Counts of [The poor] and [needs] don't correspond.
Sentence 6 at line 6
Subject: I
Verb: saw
Object: some people
Subject: who
Verb: were running
Adverb: away

Sentence 7 at line 7
Subject: I
Verb: want
Object: some milk
Adverb: in my coffee
```

Figure 13: Wrong sentences whose mistakes are recognized, and correct sentences which are parsed.

3.2.2 Verb-Adverb agreement and capitalization

When using adverbs of time, it is required to pay attention to the verb tense. Moreover, when a new sentence is going to be started (not dependent ones in compound or complex sentences), it should be capitalized (Figure 14).

I have seen him last night. he will visit my brother yesterday.

I visit my grandparents today.

She has never washed the dishes, and she will never wash the dishes.

4 Conclusion

We have seen how different sentence structures in English can be converted into mathematical parsing rules. This fact enables us to recognize parts of speech of different words in a sentence. In addition, by defining semantic actions, some grammatical mistakes are handled by the parser.

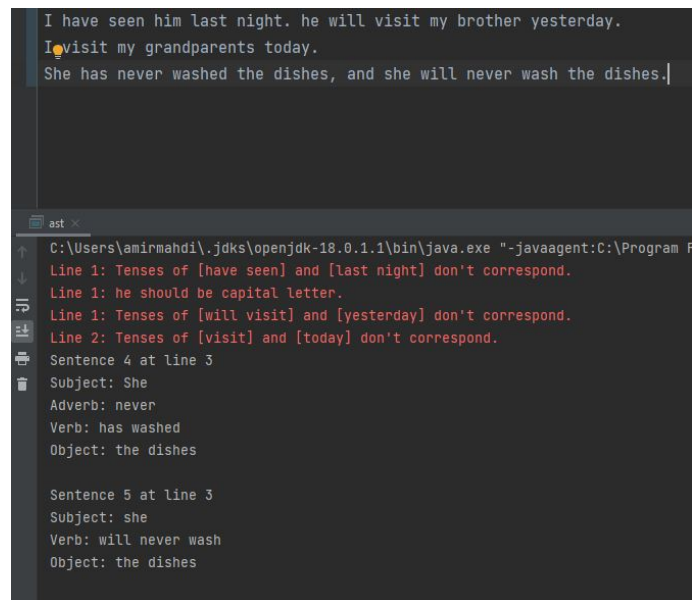


Figure 14: Tense-agreement and capitalization errors found by GrammarEditor