Finding Parts of Speech and Grammatical Mistakes by Using Production Rules

Amirmahdi Ansaripour (Student ID: 810198358)

1 Introduction

The grammar of English Language follows precise structures which can be written as production (parsing) rules. Since computers can understand production rules, they can detect grammatical mistakes and parts of speech of the words in sentences. In this report, we describe how a model named GrammarEditor can take advantage of production rules and semantic actions in order to detect parts of speech and grammatical errors .

2 The first task: Finding part of speech

2.1 Grammar files

For this task, some .g4 files are created which have the following usages:

- **EnglishGrammar.g4**: In this file, different kinds of sentences in a paragraph are coded. The basic type (rules **firstSentence** and **sentences**), compound sentences joint by **conjunctions**, **relativeClauses** which come after a noun, and **infinitivePhrases**.

```
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: Rule **sentence** in **EnglishGrammar.g4**

- **Present.g4**: In this file, the components of each type of phrases, sentences, and clauses discussed in the previous part are managed. 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**.

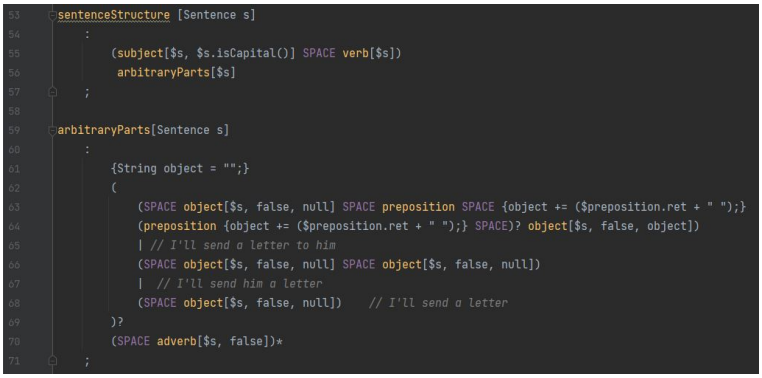

Figure 2: Rule **sentenceStructure** in **Present.g4**

- **PartOfSpeech.g4**: Different rules that a word may get in a sentence are included in this file. Forms that a **verb** can get (modals, present perfect, to be), different types of **adverbs** (following prepositions of time, place, etc), and **nounPhrases**, which are used for **subject** and **object**, are some rules contained in this file.

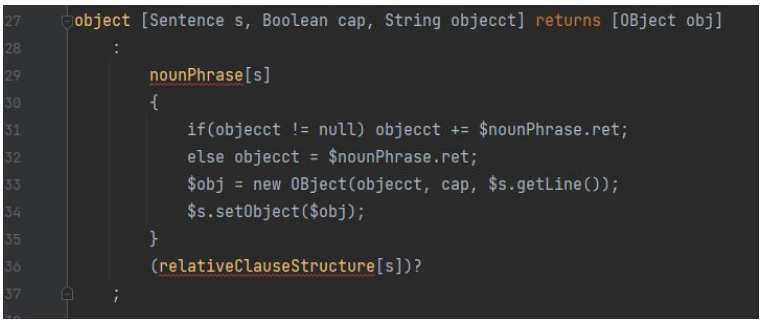

Figure 3: Rule **object** in **PartOfSpeech.g4**

- **Lexer.g4**: The most basic file which contains 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, some basic sentence structures and how they are parsed 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 arbitrary number of adverbs. Consider the following text:

I have studied a lot to become successful.

I was running in the park.

The apple has been eaten.

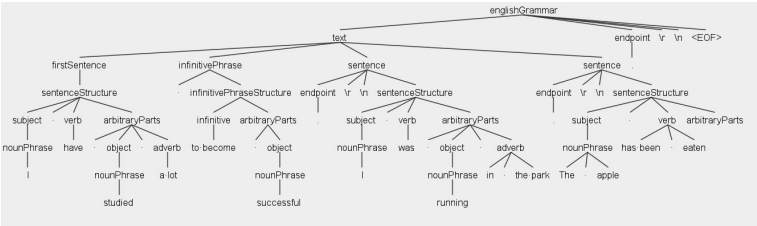

Figure 5: The parsing tree of the example

Another example:

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

2.2.2 Infinitive phrases

There is a subtle difference when using the word 'to'. It can be placed as a preposition before adverbs and objects. Also, it can be used when using a infinitive after a verb. The important point is that the verb following 'to' in the second case should always have simple form.

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

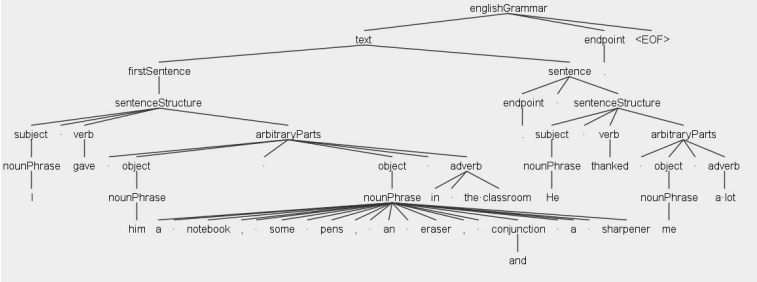

Figure 6: The parsing tree of a simple sentence having more than one object

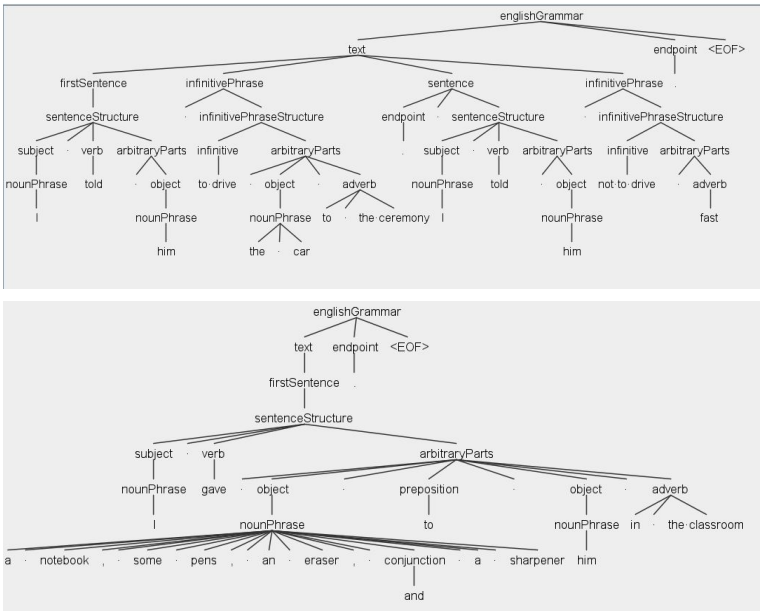

Figure 7: The parsing tree of a sentence containing infinitive

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.

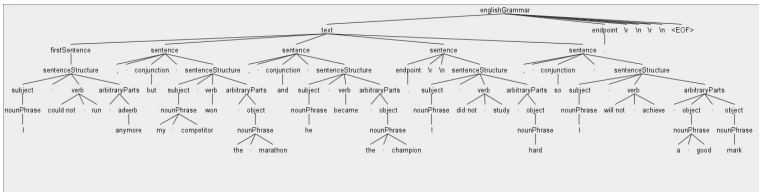

Figure 8: The parsing tree of compound sentences

2.2.4 Relative clauses

Relative clauses have the responsibility of describing and 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 containing a noun and the relative clauses (adjective clauses) coming after the noun. 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 tree whose nodes represent different parts of speech is obtained. This tree is called **AST tree**. In order to do more analysis on the nodes of AST tree, we should define some **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 us detect grammatical or dictation errors.

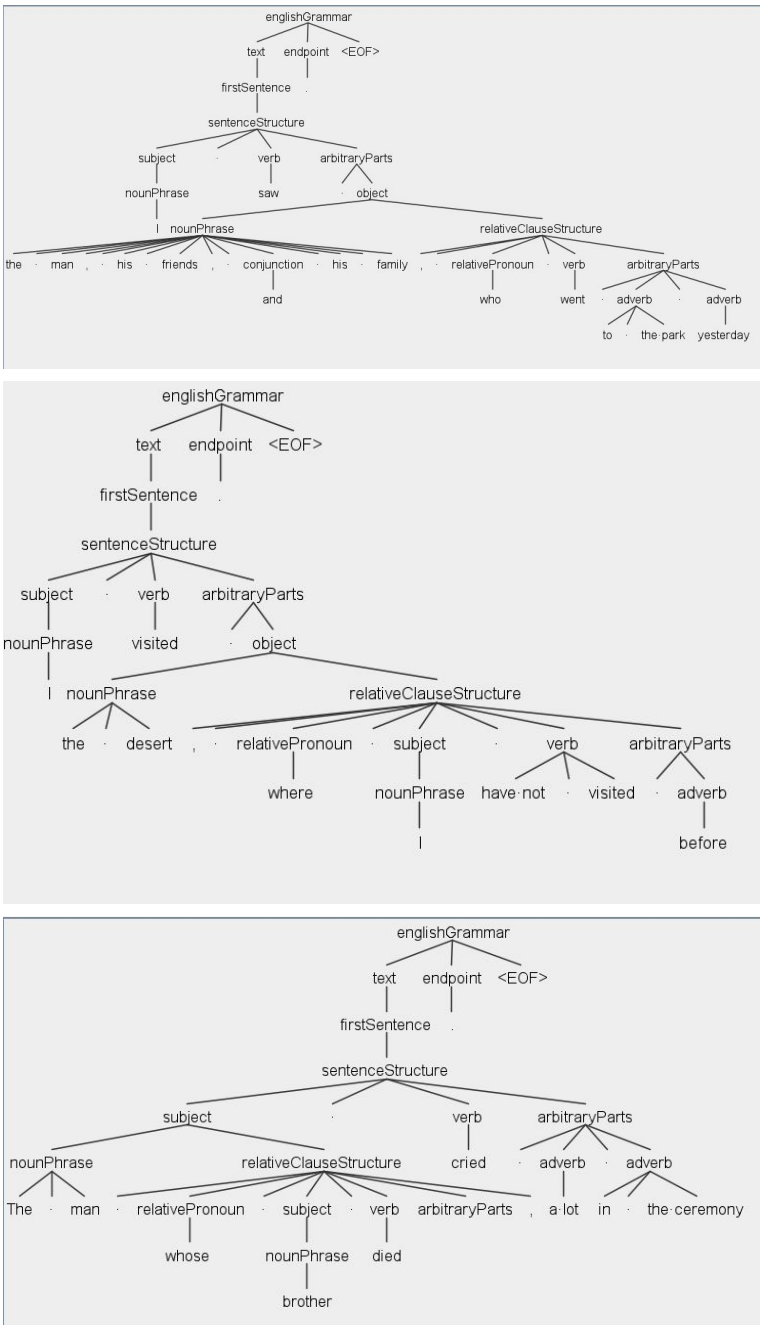

Figure 9: The parsing tree of sentences having relative clauses

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

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

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

For example, we can check the subject-verb agreement in each sentence, or we can check the tense of the verb and adverbs of a sentence (Figure 11).

```
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

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

```
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

3.2 Grammatical mistakes

In this section, some errors which can be detected 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.

Also pay attention to these correct sentences:

I saw some people who were running away.
I want some milk in my coffee.

3.2.2 Verb-Adverb agreement and capitalization

When using adverbs of time, it is required to pay attention to the tense of the verb. 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.

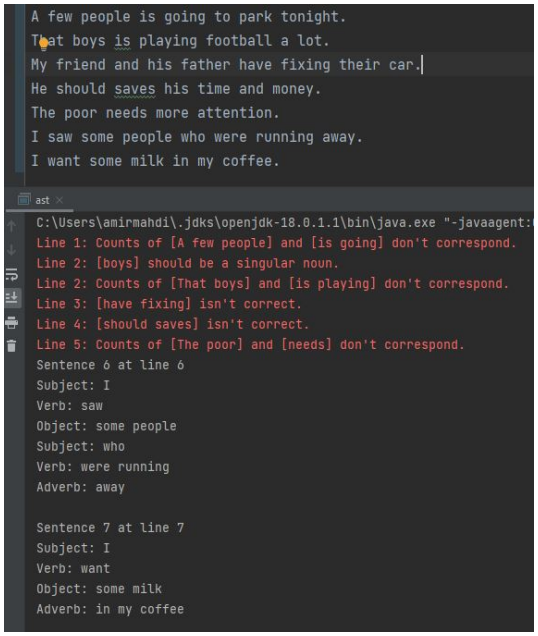

Figure 13: Wrong sentences whose mistakes are detected, and correct sentences which are parsed.

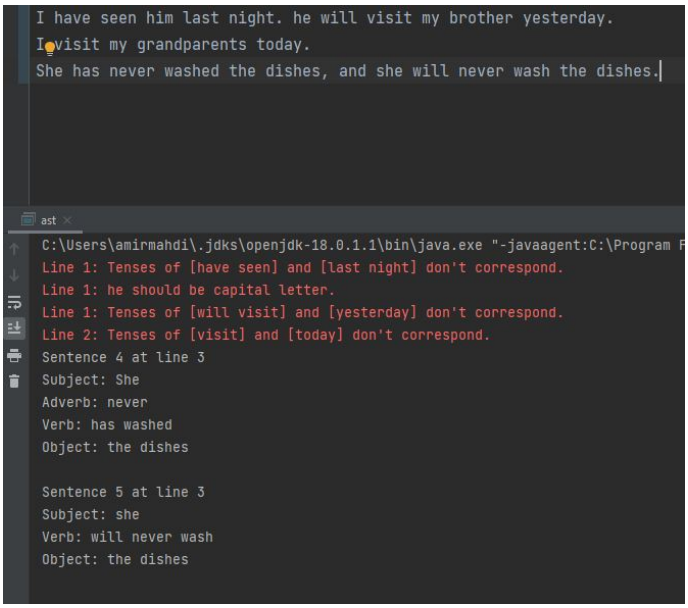

Figure 14: The detection of tense-agreement and capitalization errors.

4 Conclusion

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