SentiAspectExtractor Introduction

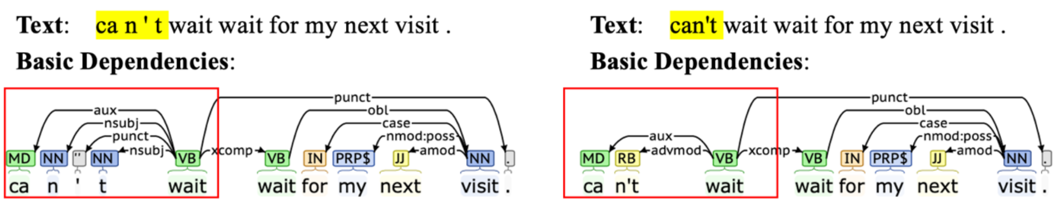
SentiAspectExtractor is an independent project that can extract aspects from given text and a list of text opinions using several syntax-based rules. In this introduction, we will provide detailed descriptions of its algorithmic workflow and how to use it.

• Algorithmic Workflow

The algorithm requires text and its corresponding list of opinions as input. The aspect extraction algorithm can be divided into five steps: 1) preprocessing text, 2) selecting representative nodes for input opinion expressions, 3) extracting aspect for representative opinion node based on a set of syntactic rules, 4) extending aspect, 5) trimming aspects, and 6) supplementing with frequent aspects.

1) Preprocessing Text

In the algorithm, we will utilize CoreNLP [1] for part-of-speech(POS) tagging, dependency parsing, and constituency parsing of the input text. Some incorrect text writing styles may affect the parsing results of CoreNLP. Therefore, in this step, we will correct these writing styles. Figure 2 provides an example illustrating how different writing styles of text can affect the analysis results of CoreNLP. On the right-hand side of the figure, there are common or correct writing styles, along with their corresponding dependency parse graphs. On the left-hand side, there are incorrect segmentation styles and their dependency parse graphs. The dependency parse graphs in this figure are extracted from the running examples on the official website of CoreNLP.



Text: ca n ' t wait wait for my next visit .
Basic Dependencies:
aux
nsubj
punct
nsubj
xcomp
punct
obl
case
nmod:poss
amod
MD NN '' NN VB VB IN PRP$ JJ NN .
ca n ' t wait wait for my next visit .
Text: can't wait wait for my next visit .
Basic Dependencies:
aux
advmod
xcomp
punct
obl
case
nmod:poss
amod
MD RB VB VB IN PRP$ JJ NN .
ca n't wait wait for my next visit .

Figure 1 An example illustrating how different writing styles of text can affect the analysis results of CoreNLP.

2) Selecting Representative Nodes For Input Opinion Expressions

Considering that there may be multi-word expressions for opinions, in this step, we will select a representative node for multi-word expressions, and proceed to the next step for aspect extraction. If the last word in a multi-word opinion expression has a POS tag of adjective, verb, or noun, we directly select this last word as the representative node. If the above condition is not met, we use the importance of a node in the dependency graph to select the representative node. Specifically, we calculate the sum of the in-degree and out-degree of each node in the multi-word expression in the dependency graph. The larger the sum, the more connected the node is to other nodes in the graph, and the more important it is. We select the node with the highest sum of in-degree and out-degree as the representative node.

3) Extracting aspects based on a set of syntactic rules

In this step, we will extract aspect for representative opinion node based on a set of syntactic rules. These rules are constructed based on the dependency graph of CoreNLP. A pair of dependency relationships in the graph can be described as "$Head \xrightarrow{reln} Dep$". The starting node of the arrow ($Head$) is a core node (also known as dominant words) in a relationship. The ending node of the arrow (Dep) is a dependent node. " $reln$ " is used to describe the dependency relationship between the two. For example, in the phrase “my next visit”, there is a relationship "$visit \xrightarrow{amod} next$", which can be described as "next" being a adjectival modifier ($amod$) for "visit".

The extraction rules can be divided into two parts: general extraction rules, which are applicable to all representative opinion nodes; and customized extraction rules, which will extract specifically based on the different POSs of the representative opinion nodes. Table 1 summarizes all the rules proposed in this section. Rules are mutually parallel and do not affect each other.

Table 1 Extraction Rule Summary

General Extraction Rule:		
	Subject Rules: $opinion - \boldsymbol{FindSubject} \rightarrow subjOfOP$	
	R-1	If $subjOfOP\ [LegalTag]$; Extract $subjOfOP$
	R-2	If, $subjOfOP[Adj] \xrightarrow{nmod} modOfSubj[LegalTag]$; Extract $modOfSubj$
	R-3	If, $subjOfOP[Verb] \xrightarrow{obj/obl} objOfSubj[LegalTag]$; Extract $objOfSubj$
	R-4	If $subjOfOP[Verb] \xrightarrow{obj/obl} NULL$; Extract $subjOfOP$
	*	If $subjOfOP\ \{\vert person\vert\}$; Skip out
	Predicative Rules:	
	R-5	If $BE \xleftarrow{cop} govOfBE \xrightarrow{subj} subj$, $opinion \dashleftarrow subj$, $govOfBE\ [Noun]$; Extract $govOfBE$
	R-6	If $BE \xrightarrow{ccomp} ccomp \xrightarrow{subj} subjOfcomp$, $BE \xrightarrow{subj} subj$, $opinion \dashleftarrow subj$, $subjOfcomp\ [LegalTag]$; Extract $subjOfcomp$
	Subordinate Structure Rules:	
	R-7	If $modedByParens[LegalTag]\ (\cdots opinion \cdots)$; Extract $modedByParens$
	R-8	If $opinion \dashleftarrow acl, acl \xleftarrow{acl} modedByACL[LegalTag]$; Extract $modedByACL$
	R-9	If $opinion \dashleftarrow acl$, $acl \xleftarrow{acl} modedByACL$, $modedByACL \xrightarrow{subj} subj[LegalTag]$, $modedByACL \xrightarrow{cop} BE$;

		Extract $subj$
Customized Extraction Rules:		
	Adjective Extraction Rules:	
		(The modified object of the opinions)
	R-10	If $adjOP \xleftarrow{*mod} modedByAdj[Noun]$; Extract $modedByAdj$
	R-11	If $adjOP \xleftarrow{*mod} modedByAdj[Noun] \xleftarrow{nmod:at/with/in} indirectlyModed[Noun]$; Extract $indirectlyModed$
	R-12	If $adjOP \xleftarrow{*mod} modedByAdj[Adj] \xleftarrow{*mod} indirectlyModed[Noun]$; Extract $indirectlyModed$
		(The object of the opinions)
	R-13	If $adjOP \xleftarrow{advcl} verb\{find, make\} \xrightarrow{obj/obl} objOfAdj[LegalTag]$; Extract $objOfAdj$
	R-14	If $opinion - \textbf{FindSubject} \rightarrow subjOfOP\{NULL, \lvert person \rvert\}$ $adjOP \xrightarrow{obj/obl} objOfAdj[LegalTag]$; Extract $objOfAdj$
		(The complements of the opinions)
	R-15	If $adjOP \xrightarrow{xcomp} compNode[Verb]$; Extract $compNode$
	R-16	If $adjOP - \textbf{FindSubject} \rightarrow subjOfOP\{NULL, \lvert person \rvert\}$ $adjOP \xrightarrow{advcl/ccomp} clNode \xrightarrow{subj} subjOfCL[LegalTag]$; Extract $subjOfCL$
	R-17	If $adjOP - \textbf{FindSubject} \rightarrow subjOfOP\{NULL, \lvert person \rvert\}$ $adjOP \xrightarrow{xcomp} compNode[Verb] \xrightarrow{obj/obl} objOfComp[LegalTag]$; Extract $objOfComp$
	R-18	If $adjOP - \textbf{FindSubject} \rightarrow subjOfOP\{NULL, \lvert person \rvert\}$ $adjOP \xrightarrow{xcomp} compNode[Verb] \xrightarrow{ccomp} clNode \xrightarrow{subj} subjOfCL[LegalTag]$; Extract $subjOfCL$
	Verb Extraction Rules:	
		(The modified object of the opinions)
	R-19	If $verbOP \xleftarrow{*mod} modedByVerb[Noun]$; Extract $modedByVerb$
		(The object of the opinions)
	R-20	If $verbOP - FindSubject \rightarrow subjOfOP\{NULL, \lvert person \rvert\}$, $verbOP \xrightarrow{obj/obl} objOfVerb[LegalTag]$; Extract $objOfVerb$
		(The complements of the opinions)
	R-21	If $verbOP - FindSubject \rightarrow subjOfOP\{NULL, \lvert person \rvert\}$, $verbOP \xrightarrow{ccomp} clNode \xrightarrow{subj} subjOfCL[LegalTag]$; Extract $subjOfCL$
	Noun Extraction Rules:	
		(The modified object of the opinions)
	R-22	If $nounOP \xleftarrow{compound} cpNode[Noun]$; Extract $cpNode$
	R-23	If $nounOP \xrightarrow{compound} cpNode[Noun]$; Extract $cpNode$
	R-24	If $nounOP \xrightarrow{nmod} modedByNoun[LegalTag]$; Extract $modedByNoun$

	(The object of the opinions)	
	R-25	If $nounOP \xleftarrow{obj/obl} verb\{find, make\} \xrightarrow{obj/obl} objOfNoun[LegalTag]$; Extract $objOfNoun$
	R-26	If $nounOP \xrightarrow{obj/obl} objOfNoun[LegalTag]$]; Extract $objOfNoun$
	Adverb Extraction Rules:	
	(The modified object of the opinions)	
	R-27	If $advOP \xleftarrow{*mod/dep} modedByAdv[Noun]$; Extract $modedByAdv$
	R-28	If $advOP \xleftarrow{*mod} modedByAdv[Verb]$; Extract $modedByAdv$
	R-29	If $advOP \xleftarrow{*mod} modedByAdj[AdjAdv/Verb] \xleftarrow{*mod} indirectlyModed[Noun]$; Extract $indirectlyModed$
	(The object of the opinions)	
	R-30	If $advOP \xleftarrow{*mod} verb\{find, make\} \xrightarrow{obj/obl} objOfVerb[LegalTag]$; Extract $objOfVerb$
	R-31	If $advOP - FindSubject \to subjOfOP\{NULL, \lvert person \rvert\}$, $advOP \xleftarrow{*mod} modedByAdv \xrightarrow{obj/obl} objOfAdv[LegalTag]$; Extract $objOfVerb$
	Other Extraction Rules:	
	(The modified object of the opinions)	
	R-32	If $otherOP \overset{*}{\leftarrow} govOfOP[LegalTag]$; Extract $govOfOP$
Tips: FindSubject is an algorithm mentioned below; "[]" is a requirement for part of speech; "{}" is a requirement for the word itself;		

General Extraction Rule: it consists of three sub-rules: subject rules, predicative rules, and subordinate structure rules.

Subject Rules : We will extract the subject ($subjOfOP$) of the opinion as the aspect. Different treatments will be applied based on the different cases of the subject. If the `POS` of $subjOfOP$ belongs to `LegalTag`, the $subjOfOP$ will be extracted as an aspect.(Figure 3(R-1)) If $subjOfOP$ is an adjective (`JJ*`), its modified object $modOfSubj$ ($subjOfOP \xrightarrow{nmod} modOfSubj$) will be extracted as an aspect.(Figure 3 (R-2)). If $subjOfOP$ is a verb (`VB*`) and has an object $objOfSubj$ ($subjOfOP \xrightarrow{obj/obl} objOfSubj$) whose `POS` belongs to `LegalTag`, the $objOfSubj$ will be extracted as an aspect.(Figure 3(R-3)). If $subjOfOP$ is a verb and doesn't have objects ($subjOfOP \xrightarrow{obj/obl} NULL$), the $subjOfOP$ will be extracted as an aspect.(Figure 3(R-4)). Noted that if we find $subjOfOP$ is a person during the above processes, we will skip out. Because we cannot distinguish between "He is good" and "He is very satisfied". The subject extraction algorithm is shown in Figure 4

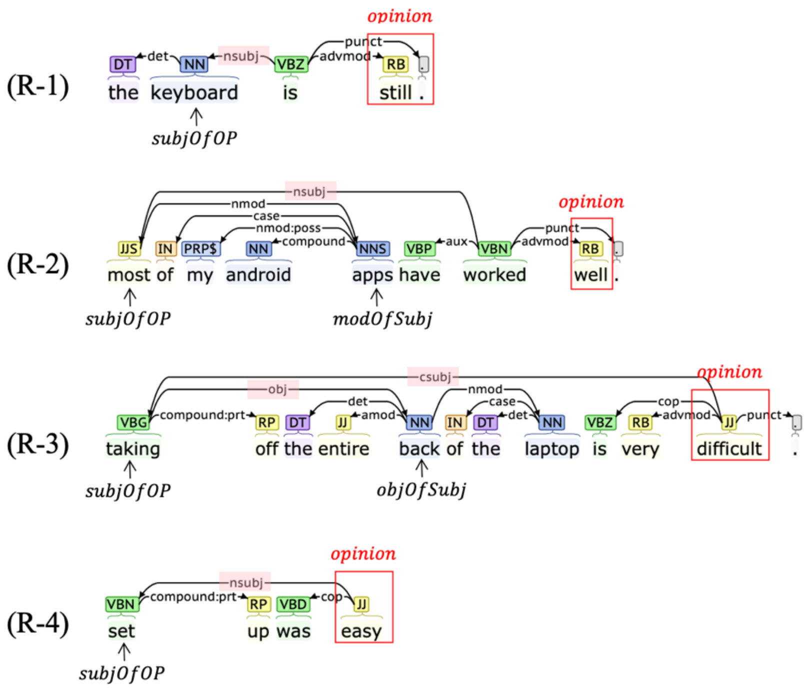
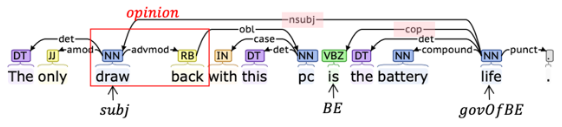


Figure 3 Examples of the subject rules.

```
Algorithm 1 Find Subject
Input:  Node in the dependency graph node, the dependency graph graph
Output: The subject of node
1:  nodeListToRoot = graph.getPathToRoot(node); // get the path from the current node to the root
2:  for i from 0 to |nodeListToRoot| do:
3:     nodeCurr = nodeListToRoot[i] ; // the current node
4:     nodeGov = nodeListToRoot[i+1] ; // the head/parent node of current node
5:     reln = graph.getRelation(nodeGov, nodeCurr) ;  // get the relationship between the two
6:     // If the current node is part of the subject (connected to nodeGov with a subject relationship), then exit the check directly
7:     if  isSubj(reln) then :
8:        return Null;
9:     end if
10:    nodeCurr_ChildSet = graph.getChildren(nodeCurr) ;
11:    for child in nodeCurr_ChildSet do:
12:       reln = graph.getRelation(nodeCurr, child) ;
13:       if  isSubj(reln) then :
14:          return child;
15:       end if
16:    end for
17: end for
```

Figure 4 The subject extraction algorithm

Predicative Rules : We will use the BE-verb as a clue to traverse the subject-link verb-predicative structure in the sentence. If the opinion is part of the subject, we take the predicative as the aspect. There are two cases for the predicative: if it is a noun phrase, we take the noun phrase as the aspect; if it is a clause, we take the subject of the clause as the aspect. The dependency relationship between these two cases is also different. When the predicate is a phrase, it is $BE \xleftarrow{cop} govOfBE \xrightarrow{subj} subj$, if $opinion \dashleftarrow subj$ and $govOfBE$ is a noun, we will select $govOfBE$ as an aspect. Here, we use "$Dep \dashleftarrow Head$" to describe that Dep is directly or indirectly dependent on $Head$ (manifested as a one-way path from $Head$ to the Dep in the graph), or Dep is $Head$ itself. The example can be seen in Figure 5(R-5). When the predicate is a clause, it is $BE \xrightarrow{subj} subj, BE \xrightarrow{ccomp} ccomp \xrightarrow{subj} subjOfcomp$, and if $opinion \dashleftarrow subj$ and the POS of $subjOfcomp$ belongs to noun(NN*), pronoun(PRP*), number(CD), or determiner (DT) (which are hereinafter referred to as LegalTag), we will select $subjOfcomp$ as an aspect. The example can be seen in Figure 5(R-6).

(R-5)

opinion
DT JJ NN RB IN DT NN VBZ DT NN NN
The only draw back with this pc is the battery life .
det amod advmod obl case det nsubj cop det compound punct
subj
BE
govOfBE

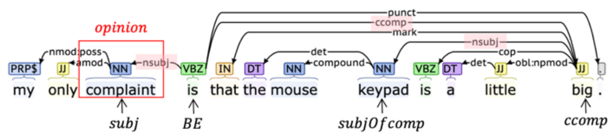
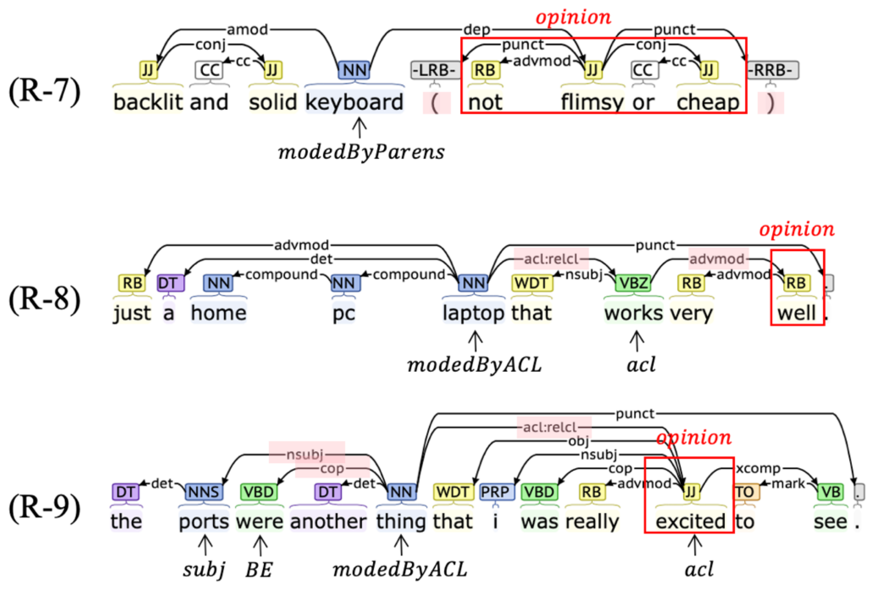
(R-6)

opinion
PRP$ JJ NN VBZ IN DT NN NN VBZ DT JJ JJ
my only complaint is that the mouse keypad is a little big .
nmod:poss amod nsubj punct ccomp mark det compound nsubj cop det obl:npmod
subj
BE
subjOfcomp
ccomp

Figure 5 Examples of the predicative rules.

Subordinate Structure Rules : If the opinion is in a subordinate structure, the object modified by the subordinate structure will be considered as the aspect. Here, we consider two types of subordinate structures, namely parentheses clauses and adnominal clauses. For parentheses clauses, if we detect a structure like "$modedByParens\,(\cdots opinion \cdots)$", and the `POS` of $modedByParens$ belongs to `LegalTag`, we extract $modedByParens$ as the aspect (Figure 6(R-7)) . For adnominal clauses (acl), if the opinion is part of an adnominal clause ($opinion \leftarrow\!\cdots acl$), and there is a `LegalTag` $modedByACL$ modified by acl ($modedByACL \xrightarrow{acl} acl$) , we extract $modedByACL$ as the aspect (Figure 6(R-8)). Combining with the subject rules, if the node being modified by the acl is also a predicate in a subject-copula-predicate relationship ($modedByACL \xrightarrow{subj} subj$, $modedByACL \xrightarrow{cop} cop$), and the `POS` of $subj$ belongs to `LegalTag`, we will extract $subj$ as the aspect (Figure 6(R-9)).



(R-7)
opinion
backlit and solid keyboard (not flimsy or cheap)
modedByParens
(R-8)
opinion
just a home pc laptop that works very well .
modedByACL
acl
(R-9)
opinion
the ports were another thing that i was really excited to see .
subj
BE
modedByACL
acl

Figure 2 Examples of the subordinate structure rules.

Customized Extraction Rules: Nodes with different `POS` in the dependency graph are positioned in different dependency relationships. Therefore, we have custom rules for four important `POS`s (i.e. adjectives, verbs, nouns, adverbs) to search aspects. These rules are roughly divided into the extraction rules for the modified object of the opinions, the extraction rules for the object of the opinions, and the extraction rules for the complements of the opinions.

Adjective Extraction Rules: In order to extract the **modified objects of adjective opinions**, we have set the following rules: **1)** $adjOP \xleftarrow{*mod} modedByAdj[Noun]$, if the modified object

($modedByAdj$) of the adjective opinion is a noun, then it will be extracted as an aspect (Figure 7(R-10)); **2)** $adjOP \xleftarrow{*mod} modedByAdj[Noun] \xleftarrow{nmod:at/with/in} indirectlyModed[Noun]$, If the POS of $modedByAdj$ is noun, it has a noun dependent ($indirectlyModed[Noun]$) , the relationship between them is a compound noun modifier ($nmod$), and the preposition between the two is "at", "with" , or "in", then $indirectlyModed$ will be extracted as an aspect (Figure 7(R-11)); **3)** $adjOP \xleftarrow{*mod} modedByAdj[Adj] \xleftarrow{*mod} indirectlyModed[Noun]$, If the POS of $modedByAdj$ is adjective, it modifies a noun ($modedByAdj[Adj] \xleftarrow{*mod} indirectlyModed[Noun]$), then $indirectlyModed$ will be extracted as an aspect (Figure 7(R-12)).

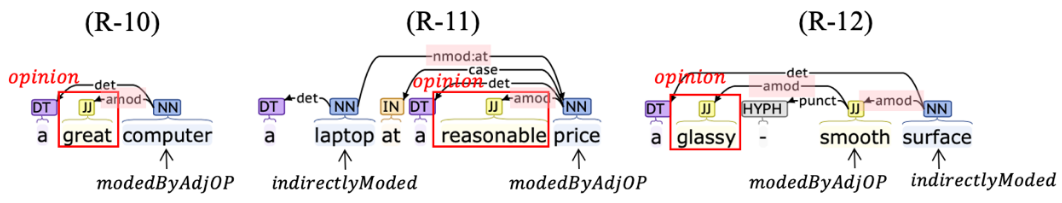


Figure 3 Examples of extracting the modified objects of adjective opinions

In order to extract the **objects of adjective opinions**, we have set the following rules: **1)** $adjOP \xleftarrow{advcl} verb\{find, make\} \xrightarrow{obj/obl} objOfAdj[LegalTag]$, in the structure like 'make/find sth. adj.', we will extract 'sth.' as the aspect (Figure 8(R-13)) ; **2)** $adjOP - FindSubject \rightarrow subjOfOP\{NULL, |person|\}, adjOP \xrightarrow{obj/obl} objOfAdj[LegalTag]$, if the opinion does not have a subject or the subject is a person, we will extract its object whose POS belongs to LegalTag as the aspect (Figure 8(R-14)).

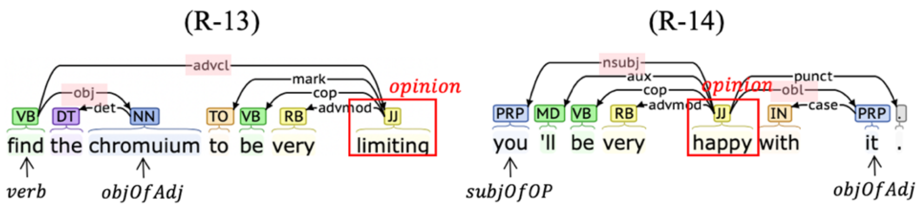

(R-13)
(R-14)
opinion
find the chromuium to be very limiting
verb
objOfAdj
you 'll be very happy with it .
subjOfOP
objOfAdj

Figure 4 Examples of extracting the objects of adjective opinions

In order to extract the **complements of adjective opinions**, we have set the following rules: **1)** $adjOP \xrightarrow{xcomp} compNode[Verb]$, if the open clausal completion node ($xcompNode$) of $adjOP$ is a verb, we will extract $xcompNode$ as the aspect (Figure 9(R-15)) ; **2)** $adjOP - FindSubject \rightarrow subjOfOP\{NULL, |person|\}, adjOP \xrightarrow{advcl/ccomp} clNode \xrightarrow{subj} subjOfCL[LegalTag]$, if $adjOP$ has no subject or has a person subject, we will extract the subject of its complement clause as aspect (Figure 9(R-16)) ; **3)** $adjOP - FindSubject \rightarrow subjOfOP\{NULL, |person|\},$

$adjOP \xrightarrow{xcomp} compNode[Verb] \xrightarrow{obj/obl} objOfComp[LegalTag]$, if $adjOP$ has no subject or has a person subject, we will extract the object of its complement verb as aspect(Figure 9(R-17)) ; **4)** $adjOP - FindSubject \rightarrow subjOfOP\{NULL, |person|\}, adjOP \xrightarrow{xcomp} compNode[Verb] \xrightarrow{ccomp} clNode \xrightarrow{subj} subjOfCL[LegalTag]$, if $adjOP$ has no subject or has a person subject, we will extract the subject of its complement verb clause as aspect (Figure 9(R-18)) .

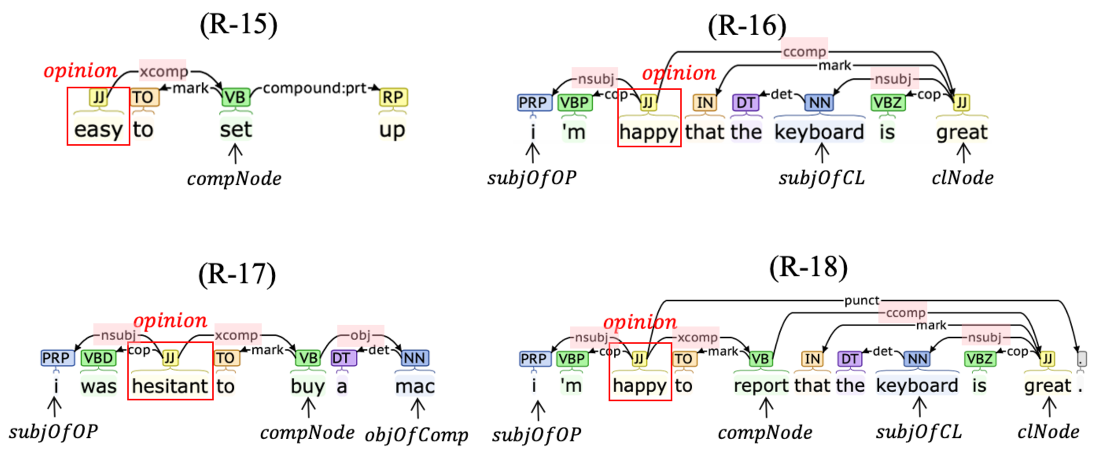


Figure 5 Examples of extracting the complements of adjective opinions

Verb Extraction Rules: In order to extract the **modified objects of verb opinions**, we have: $verbOP \xleftarrow{*mod} modedByVerb[Noun]$ (Figure 10(R-19)) . In order to extract the **objects of verb opinions**, we have: $verbOP - FindSubject \rightarrow subjOfOP\{NULL, |person|\}$, $verbOP \xrightarrow{obj/obl} objOfVerb[LegalTag]$ (Figure 10(R-20)) . In order to extract the **complements of verb opinions**, we have: $verbOP - FindSubject \rightarrow subjOfOP\{NULL, |person|\}$, $verbOP \xrightarrow{ccomp} clNode \xrightarrow{subj} subjOfCL[LegalTag]$ (Figure 10(R-21)) .

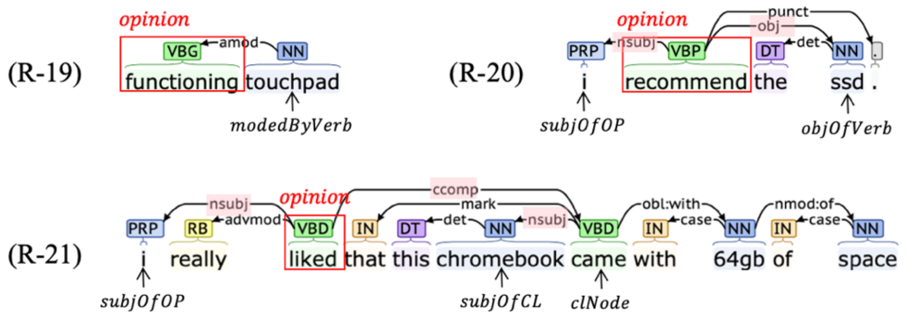


Figure 6 Examples of verb extraction rules

Noun Extraction Rules: In order to extract the **modified objects of noun opinions**, we have: **1)** $nounOP \xleftarrow{compound} cpNode[Noun]$ (Figure 11(R-22)); **2)** $nounOP \xrightarrow{compound} cpNode[Noun]$ (Figure 11(R-23)); **3)** $nounOP \xrightarrow{nmod} modedByNoun[LegalTag]$ (Figure 11(R-24)). In order to extract the

objects of noun opinions, we have: **1)** $nounOP \xleftarrow{obj/obl} verb\{find, make\} \xrightarrow{obj/obl} objOfNoun[LegalTag]$ (Figure 11(R-25)); **2)** $nounOP \xrightarrow{obj/obl} objOfNoun[LegalTag]$ (Figure 11(R-26)).

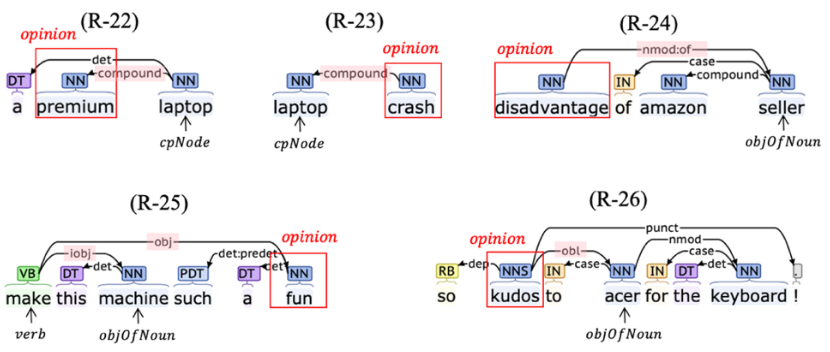


Figure 7 Examples of noun extraction rules

Adverb Extraction Rules: In order to extract the **modified objects of noun opinions**, we have: **1)** $advOP \xleftarrow{*mod/dep} modedByAdv[Noun]$ (Figure 12(R-27)); **2)** $advOP \xleftarrow{*mod} modedByAdv[Verb]$ (Figure 12(R-28)) ; **3)** $advOP \xleftarrow{*mod} modedByAdj[AdjAdv/Verb] \xleftarrow{*mod} indirectlyModed[Noun]$ (Figure 12(R-29)). In order to extract the **objects of noun opinions**, we have: **1)** $advOP \xleftarrow{*mod} verb\{find, make\} \xrightarrow{obj/obl} objOfVerb[LegalTag]$ (Figure 12(R-30)); **2)** $advOP - FindSubject \rightarrow subjOfOP\{NULL, |person|\}$, $advOP \xleftarrow{*mod} modedByAdv \xrightarrow{obj/obl} objOfAdv[LegalTag]$ (Figure 12(R-31)).

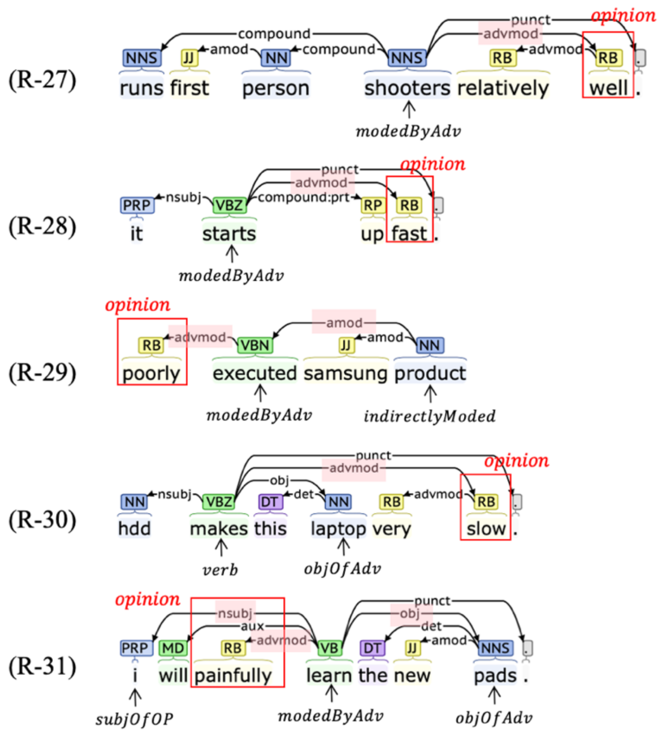


Figure 8 Examples of adverb extraction rules

Other Extraction Rules: For opinions that do not belong to the above POSs, we will check $otherOP \overset{*}{\leftarrow} govOfOP[LegalTag]$ (Figure 13(R-32)).

(R-32)

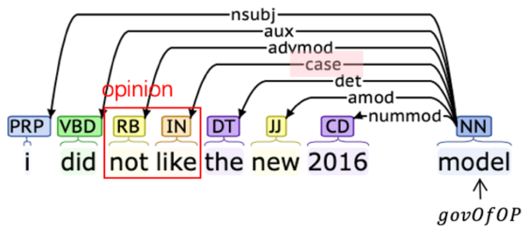


Figure 9 Examples of other extraction rules

Above is the rules delineated according to the actual implementation logic. For the sake of clarity in the paper, we have reorganized them. Their mapping relationships are as follows:

Subject Structure Based Aspect Extraction Pattern: R-1、R-2、R-3、R-4、R-5、R-6、R-9;

Modifying Relationship Based Aspect Extraction Pattern: R-7、R-8、R-10、R-11、R-12、R-13、R-15、R-19、R-22、R-23、R-24、R-25、R-27、R-28、R-29;

Opinion Object Based Aspect Extraction Pattern: R-14、R-16、R-17、R-18、R-20、R-21、R-26、R-31、R-32

<u>4) Extending aspect from the core node</u>

According to step three, we can only find one node, but in real-world data, there are often multi-word aspects, such as 'backlit keyboard'. To address this issue, we have set up several aspect extension rules (AER). **AER-1**: If the core word is a verb, we will expand based on the "compound:prt" relation (Figure 14(AER-1)) (As of the version up to 2024.10.12, we have decided to stop considering verb aspects). **AER-2**: If the core word is a pronoun, pronoun possessive, or any determiner in 'this',' that ',' that ', and' these ', we will use CoreNLP for coreference resolution (Figure 14(AER-2)). For the vast majority of aspects with noun `POS`, we have set three granularity extension rules. **AER-3.1**: Only use 'compound' relationship to extend (Figure 14(AER-3.1)). **AER-3.2**: On the basis of AER-3.1, further extend through the " *nmod* " relationship (Figure 14(AER-3.2)). **AER-3.3**: Extension using constituency parsing. Starting from the leaf node of the core node in the constituency tree, the range of the aspect is determined based on the last noun constituents (Figure 14(AER-3.3)). The granularity of AER-3.1 to AER-3.3 extension is gradually relaxed. AER-3.1 used by default in the algorithm.

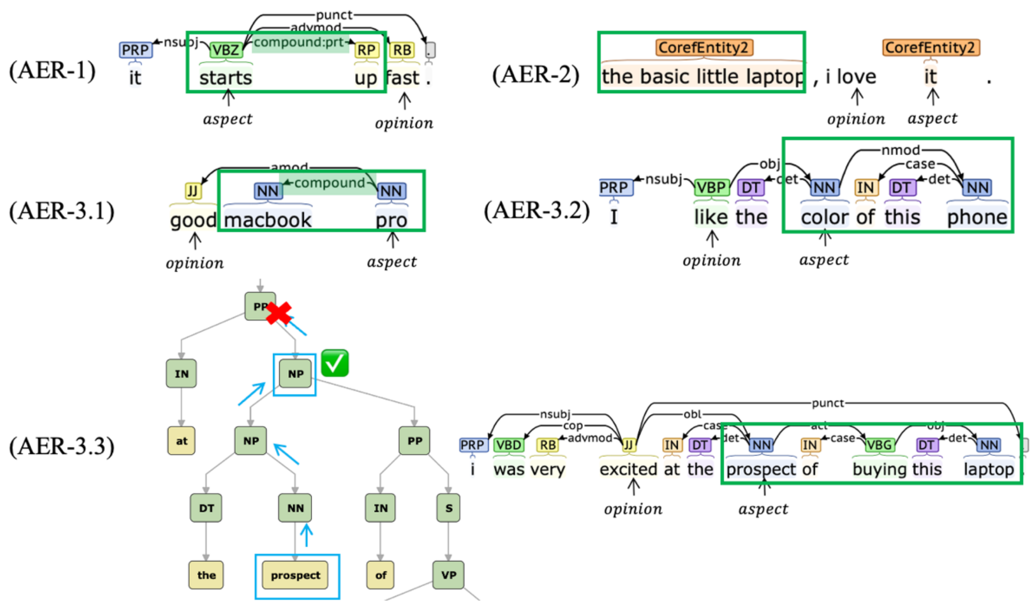

(AER-1)
it starts up fast .
PRP VBZ RP RB
nsubj advmod compound:prt punct
aspect
opinion
(AER-2)
CorefEntity2
the basic little laptop , i love it .
opinion
aspect
(AER-3.1)
good macbook pro
JJ NN NN
amod compound
opinion
aspect
(AER-3.2)
I like the color of this phone
PRP VBP DT NN IN DT NN
nsubj obj det nmod case det
opinion
aspect
(AER-3.3)
PP NP IN NP PP at DT NN IN S the prospect of VP
i was very excited at the prospect of buying this laptop
PRP VBD RB JJ IN DT NN IN VBG DT NN
nsubj cop advmod obl case det punct acl case obj det
opinion
aspect

Figure 10 Examples of aspect extension rules. The "aspect" represents the aspect that was initially found through the rules in step three. The green box indicates the extended aspect.

<u>5) Trimming aspect</u>

In this step, we will trim the aspects generated during the above steps. We have several trimming rules:

TR-1: When extending, nodes with `POS` such as "PRP", "DT", and "IN" will not be included in the list representing multi-word aspects, even if they are included in the scope according to the extension rules. However, in general, the scope of the aspect is determined by the first and last nodes that will not be trimmed. Therefore, if the above `POS` appears between these two nodes, they may still exist in the final output aspect. Taking Figure 14 AER3.3 as an example, the initial range found is "the prospect of buying this laptop". Based on the `POS`, "the", "of", and "this" will be trimmed, but the first and last nodes that are not trimmed are "prospect" and "laptop". Therefore, the final aspect is "prospect of buying this laptop". (This trimming rules are mainly for AER-3.3 to prevent the scope it extends from being too large.)

TR-2 Words in English that have lower meanings such as "thing" and "feeling" are often referred to as "generic" or "vague" words. We believe that such words cannot be regarded as explicit aspects, so before expanding the core, we check whether the core node is a vague word. If it is a vague word, we won't extend it , and the output "[-1, -1]" indicates that we have found an implicit aspect. In the algorithm, we use `implicitAspectWordList.txt` to store these words.

TR-3: Several rules in Step 3 are designed to extract verb aspects, but we are not interested in all actions. We have set up lists of verb aspects, and only the verb aspects that exist in these lists will be kept in the final output result. (As of the version up to 2024.10.12, we have decided to stop considering verb aspects)

If no aspect is left after the above trimming, the output will also be [-1,-1].

6) Supplementing with frequent aspects (New added in the 2024.10.12 version)

After the first round of analysis, we will conduct a statistical analysis of the detected aspects. If the frequency of an aspect appearing in the text is greater than or equal to 1%, it will be considered a frequent aspect. For texts in which no explicit aspect is detected during the first round, we will use the frequent aspect(s) that appear in the text as their aspect.

- **Instruction For Use**

This algorithm is published in the form of a jar package. Below, we will introduce some parameter descriptions when using jar packages:

Parameter	Parameter Description
`-help`	Output parameter description.
`-inputFile`	Set the input file address.
`-outputFile`	Set the output file path. If not set, the analysis result will be output in the same directory as the input file.
`-dict`	Set the address of the dictionary to be used for analysis.
`-explain`	Set whether to output the reason for extracting the aspect when outputting the result.
`-coreExtendType`	Set the expansion method for noun aspects. 0 corresponds to AER-3.1, 1 corresponds to AER-3.2, 2 corresponds to AER-3.3. The default parameter is 0.
`-outputFormat`	Set the output format of the result 0 represents the text-based output format, such as '[4,5]:[0,1] , [5,6] ;' 1 represents the JSON output format, such as '{"opinion": [1,2], "aspect": [3,4]}' The default parameter is 1.
`-implicitOpinionDealType`	Set the handling method for illegal or implicit opinions. 0: Output 'cannot_deal' 1: Output [-1, -1] as an aspect 2: Do not output this opinion The default parameter is 1.
`-isTextPreprocessing`	Set whether to perform text preprocessing. The default parameter is true.
`-lang`	Set which language to handle. en: English ch: Chinese The default parameter is 'en'.
`-handleSPA`	Whether to handle cases with SHAP potential aspects(SPA). When set to handle SPA (SHAP potential aspect), it allows input opinion indices to be marked with the '(SPA)' to indicate that the opinion may potentially be an aspect. When SPA is inputted into SentiAspectExtractor, it directly starts from Step-4 Extending Aspects, instead of being processed like regular opinions. true: Handle SPA false: Not handle SPA

	The default parameter is 'true'.
`-analysisParseResult`	Analyze the rule distribution of all extracted aspects.
`-usesubjectpredicativerule`	Whether to use Subject Structure Based Aspect Extraction Pattern. (Default is 'true')
`-usemodifierrule`	Whether to use Modifying Relationship Based Aspect Extraction Pattern. (Default is 'true')
`-useobjectrule`	Whether to use Opinion Object Based Aspect Extraction Pattern. (Default is 'true')
`-setloadpretag`	**(New added in the 2024.10.12 version)** SentiAspectExtractor will read the part-of-speech (POS) analysis results for 4o from the path specified by this parameter and will base subsequent dependency analysis on these results during the analysis. For texts not present in the dataset, SentiAspectExtractor will use the original CoreNLP analysis.(This operation is currently only applicable to Chinese.)

The words in the `implicitAspectWordList.txt` will not be output as explicit aspects, but in some fields, there may be situations where the words in this file are used as explicit aspects. To evaluate this risk, the "`/dataProcess/ListRiskChecker.java`" in the jar package can be used to calculate the number of times the words in the file appear in the marked explicit aspects. (This requires providing a certain amount of labeled data.)

- **Reference**

[1] https://corenlp.com