**Pseudo-code for the 20 features implemented:**

**int f1():**

counter = 1

for token in sentence:

if token == “it”:

return counter

else:

counter = counter + 1

return -1 /// in case no “it” is found in sentence.

**Note**: token is a unit that composes a sentence, e.g., a word or a punctuation

**int f2():**

counter = 0

for token in sentence:

counter = counter + 1

return counter

**int f3():**

counter = 0

for token in sentence:

if token is punctuation:

counter = counter + 1

return counter

**Note**: f1,f2,f3 can be calculated together within 1 loop thus it is also reasonable to implement them in one function. (Pseudo Code only provides the logics)

**int f4():**

counter = 0

find all noun phrases in sentence

for every noun phrase in sentence:

if it exists before “it”:

counter = counter + 1

return counter

**int f5():**

counter = 0

find all noun phrases in sentence

for every noun phrase in sentence:

if it exists after “it”:

counter = counter + 1

return counter

**Note**: ① find all noun phrases can be done using Constituent in Java ② f4, f5 can be implemented together ③ a noun phrases is considered to has at least 2 words

**boolean f6():**

find all prepositional phrases in sentence

for every prepositional phrase in sentence:

if it is immediately followed by “it”:

return true

return false

**String[] f7():**

String[] result = new String[] // Make an empty array

for four tokens preceding the “it” and four tokens succeeding the “it”:

if such token exists:

add that token into result sequentially

else: // Case that such token doesn’t exist

add ABS into result sequentially

**Note**: we can check whether tokens preceding “it” exists by looking if its index is greater than or equal to 0; and we can check whether tokens succeeding “it” exists by using “try and catch” in Java

**boolean f8():**

for each word after “it”:

if that word has a tag of “VBG”:

return true

return false

**boolean f9():**

for each word after “it”:

if that word has a tag of “IN”:

return true

return false

**int f10():**

counter = 0

for each word after “it”:

if it is an adjective:

counter = counter + 1

return counter

**boolean f11():**

for each word before “it”:

if it is a verb:

return true

return false

**boolean f12():**

for each word after “it”:

if it is a verb:

return true

return false

**boolean f13():**

for each word after “it”:

if it is an adjective:

return true

return false

**boolean f14():**

find all noun phrases in sentence

for every noun phrase in sentence:

if this phrase exists after “it”:

for every word in this phrase:

if that word is an adjective:

return true

return false

**Note**: we can find all noun phrases using constituent parsing tree provided by coreNLP

**int f15():**

for each word after “it”:

if that word is “to”:

if next word of “to” is a verb in base form:

return the quantity of words before that word

return 0 /// In case no infinitive verb is found

**int f16():**

for each word after “it”:

if that word has a tag of “IN”:

return the quantity of words between “it” and that word

return 0

**boolean f17():**

find all noun phrases in sentence

for every noun phrase in sentence:

if this phrase exists after “it”:

if the word immediately preceding the start of this noun phrase is an adjective:

return true

return false

**ArrayList<String> f18():**

generate a list of all dependency relations in sentence using CoreNLP

give an empty arrayList called “a”

for each dependency relationship:

if the dependent of this relationship is “it”:

add the governor of this relationship into “a”

sort “a” based on the distance from its elements to “it”

“target governor” = the first element in “a”

return an arrayList that contains all dependencies the “target governor” has in sentence

**boolean f19():**

for every word after “it”:

if this word is a verb:

get this verb’s senses by its lemma in WordNet

for every sense of this verb:

if this sense is related to weather:

return true

return false

**boolean f20():**

for every word after “it”:

if this word is a verb:

get this verb’s senses by its lemma in WordNet

for every sense of this verb:

if this sense is related to cognition:

return true

return false