User's manual to the suffix tree

Tommy Petersen

December 29, 2020

Contents

Chapter 1

User's manual

1.1 Preface

This is an incomplete edition of the user's manual to the java suffix tree implementation by AI Agents. Still, it is published, because it contains useful information to the users. I expect that the readers know how to compile and run java programs grouped in packages. If you have questions or suggestions for additions to the manual, then write me at tp@ai-agents.com.

1.2 Context

A context of a given string s is a suffix of s that has occured earlier as a substring of s. An example is the string $s := 0110100110101101$. The largest context is the suffix 01101. Every other suffix, including the empty context, λ, of this largest context is also a context in s.

If a context has only one preceding substring it is said to have decoded a suffix, because only one suffix can follow as a future sequence when trusting what has been learned. On the other hand, if the context has more than one preceding substring, then there is more than one possibility for the future sequence, when using what has been learned. In the above example with s equal to the sequence 0110100110101101, none of the contexts decode a suffix. But with $s := 010101$, the largest context is 0101 and the context set is $\{\lambda, 1, 01, 101, 0101\}$. The contexts 0101 and 101 both decode a suffix, while the other three contexts do not decode a suffix.

In the java implementation, a context, *Context*, is given by a *baseVertex*, a *direction* and an *offset*. The *baseVertex* is a vertex in the suffix tree, the *direction* tells which direction to branch away from *baseVertex*, while the *offset* tells how many steps to take along the branch. If *direction* equals -1 then the context points at the *baseVertex* implying that no suffix is decoded and that if you use the *indexTo* from *baseVertex* then you get an index, i, where $s[i]$ is identical to the last symbol in the context. Note, that *indexTo* equals

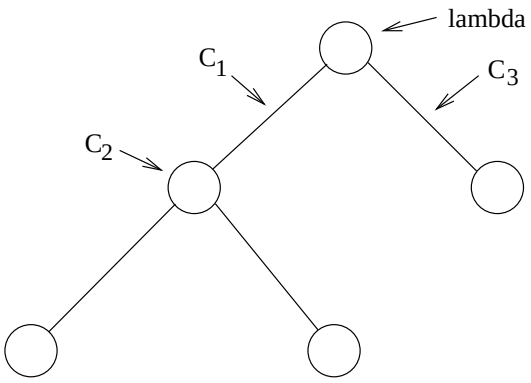

Figure 1.1: All possible types of contexts. The context λ is always in the context set. The contexts C_1, C_2 and C_3 are not necessarily in the same context set. For C_1, *direction* is greater than -1, but C_1 does not decode a suffix. For C_2, *direction* equals -1. Finally, for C_3, *direction* is greater than -1 and C_3 decodes a suffix.

infinity when the vertex containing it is a leaf. If *direction* is greater than -1, then the context points at the edge extending from *baseVertex* in direction *direction*. If you use the index $i :=$ *baseVertex.indexFrom*[*direction*]+ *offset*, then $s[i]$ is identical to the last symbol in the context. Note, that if *direction* is greater than -1 and *baseVertex.child*[*direction*] is a leaf, then the context decodes a suffix, and the symbol $s[i+1]$ should be the next symbol in the future sequence, if one can trust what has been learned about this particular context. Figure 1.1 illustrates the possible types of contexts. Also note, that a leaf can never be the base vertex of any context.

The string s can be retrieved by calling the method *getL* from any given context.

In the java implementation, a set of contexts, *contextSet*, is returned each time a new symbol is added to the suffix tree.

1.3 Average copy length

Definition 1.3.1 (Copy length for a context) *The copy length for a given context C is the sum of the length of all suffixes immediately following instances of C in the code word set. Note, that this omits suffixes in the context set.*

Definition 1.3.2 (Number of leaves for a context) *The number of leaves for a given context C is the number of leaves in the subtree rooted in either C's basevertex (if this basevertex has no direction or the child in the direction is a leaf) or its child (if the basevertex has a direction and the child in that direction is not a leaf).*

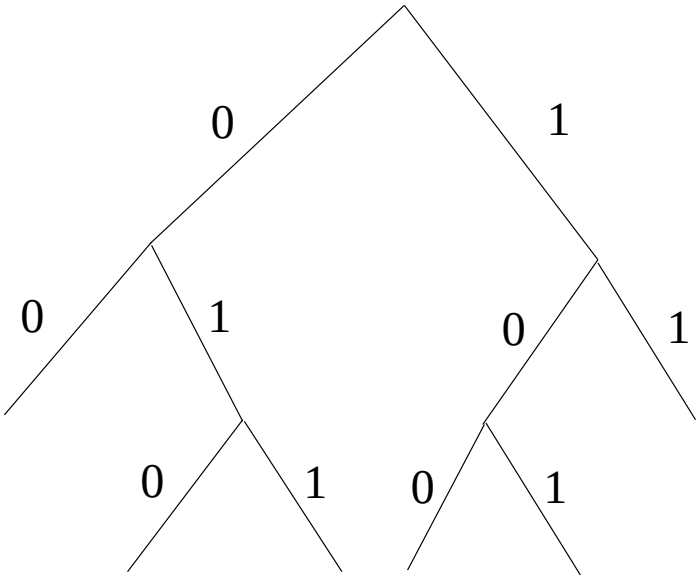

Figure 1.2: The suffix tree for the sequence 011010011010.

Definition 1.3.3 (Average copy length for a context) *The average copy length for a given context C equals the quotient of the copy length for the context to the number of leaves for the context.*

Example 1.3.1 (Average copy length for a context) *Let L be the sequence* 011010011010. *The context set is then* $\{\lambda, 0, 10, 010, 1010, 11010, 011010\}$, *and the code word set is* $\{011, 11, 101, 010, 100, 00\}$. *The suffix tree is shown in figure 1.2*

Context	Copy length	#leaves	Average copy length
λ	$12 + 11 + 10 + 9 + 8 + 7 = 57$	6	9.5
0	$11 + 8 + 6 = 25$	3	8.3334
10	$8 + 6 = 14$	2	7
010	6	1	6
1010	6	1	6
11010	6	1	6
011010	6	1	6

Note, that suffixes in the context set are omitted when computing the copy lengths. For example, the shortest copy length instance for the context λ is 7 even though λ is a prefix of every suffix in the sequence L.

Chapter 2

The java implementation

2.1 Source files

The package name is *AIAgents.SuffixTree.Java* and the source files are listed here:

- *Context.java*
- *Vertex.java*
- *codeWordSet.java*
- *contextSet.java*
- *suffixTree.java*
- *test.java*

test.java is not necessary in the implementation, but you can look in it in order to see how to use the suffix tree.