```
Data: A text with annotations,stored as a String-Array
Result: A Graph (as described above)
initialization
foreach w in words do
    if w is annotation then
        v ←new Vertex(previousWord.getCenter())
        v.setAnnotation($w$)
        Graph.addVertex(v)
        UpperVerticesList.addVertex(v)
    else
        if w is too big for the line then
            startNewLine()
            connectBasedOnPosition(UpperVerticesList)
            UpperVerticesList ←LowerVerticesList
            emptyList(LowerVerticesList)
        end
        v1 ←new Vertex(w.getTopLeft())
        v2 ←new Vertex(w.getTopRight())
        v3 ←new Vertex(w.getBottomLeft())
        v4 ←new Vertex(w.getBottomRight())

        Graph.addAll(v1,v2,v3,v4)
        UpperVerticesList.addAll(v1,v2)
        LowerVerticesList.addAll(v3,v4)
        Graph.createEdgeBetween(v1,v3)
        Graph.createEdgeBetween(v2,v4)
    end
end
```

Algorithm 1: TODO

```
Data: A single annotation's source and its Graph
Result: A List of vertices describing the Leader's Path
initialization
while currentVertex not at right text border do
    if (Graph.getTopNeighbourOf(currentVertex)$\neq$ null) $\wedge \neg$backtracking then
        Path.addVertex(currentVertex)
        currentVertex $\leftarrow$ Graph.getTopNeighbourOf(currentVertex)

    else if Graph.getRightNeighbourOf(currentVertex)$\neq$ null then
        Path.addVertex(currentVertex)
        currentVertex $\leftarrow$ Graph.getTopNeighbourOf(currentVertex)
        backtracking $\leftarrow$ False
    else
        backtracking $\leftarrow$ True
        repeat
            oldVertex $\leftarrow$ currentVertex
            currentVertex $\leftarrow$ Path.getLastEntry()
            Path.RemoveVertex(currentVertex)
        until currentVertex's Position is below oldVertex or
          Path is Empty
        if currentVertex not below oldVertex then //No path found
            break
        end
    end
end
```

Algorithm 2: TODO