Idiolect: A Reconfigurable Voice Coding Assistant

Breandan Considine, Nicholas Albion, Xujie Si

"To ascertain the meaning of an intellectual conception one should consider what practical consequences might result from the truth of that conception—and the sum of these consequences constitute the entire meaning of the conception."

–Charles Sanders Peirce, How to Make Our Ideas Clear (1878)

"The language is meant to serve for communication between a builder A and an assistant B. A is building with building-stones: there are blocks, pillars, slabs and beams. B has to pass the stones, in the order in which A needs them. For this purpose they use a language consisting of the words 'block', 'pillar', 'slab', 'beam'. A calls them out; — B brings the stone which he has learnt to bring at such-and-such a call. Conceive this as a complete primitive language."

–Ludwig Wittgenstein, Philosophical Investigations (1953)

Main Idea

- Verbally instruct a computer to program itself as a human being does.
- How do humans interact with computers? Using a screen and a keyboard.
- Visually parse the workspace and offer affordances as navigable choices.

AceJump

Helps developers search and navigate source code by assigning tags to search results in an unambiguous way while minimizing total keystrokes.

```
fun map(availabXDTags: List<String>, caches: Map<Editor, EditorOffsetCache>): Map<StringJJTag> {
  val eligibleSitesDVTag = HashMap<String, MHtableLisDDTag>>( initialCapacity: 100)
  vaKKtagsByFirstLetter : Map<Char, List<String>> = availabXATags.groupBy { it[0] }

  for ((editor : Editor , offsets : IntList ) in newResults) {
    val iter : IntListIterator = offsets.iterator()
    while (iter.hasNext()) {
      val site : Int = iter.nextInt()

      for ((firstLetter : CharMMtags : List<String> ) iXXtagsByFirstLetter.entries) {
        if (cDTTagBeginWithChar(editor, site, firstLetter)) {
          forQQtag : String iPPtags) {
            eligibleSitesMLTag.getOrPuFGtag, ::MUtableListOf).acFDTag(editor, site))
          }
        }
      }
    }
  }
```

Given a set of indices I in document d, and a set of tags T, find a bijection $f : T^* \subset T \leftrightarrow I^* \subset I$, maximizing $|I^*|$, such that:

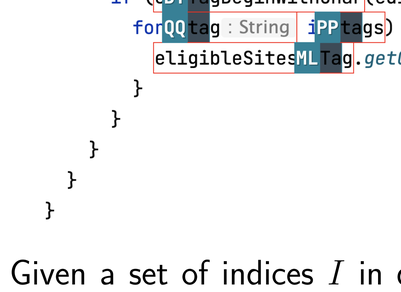
$$d[i \dots k] + t \notin d[i' \dots (k + |t|)], \forall i' \in I \setminus \{i\}, \forall k \in (i, |d| - |t|]$$

where $t \in T, i \in I$. This can be relaxed to $t = t[0]$ and $\forall k \in (i, i + K]$ for some fixed K, in most natural documents.

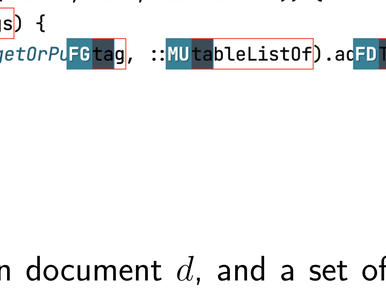
Natural language: *Maximizes tags assigned to search results in uniquely-selectable manner, i.e., should never be possible to select a tag by mistake.*

TraceJump

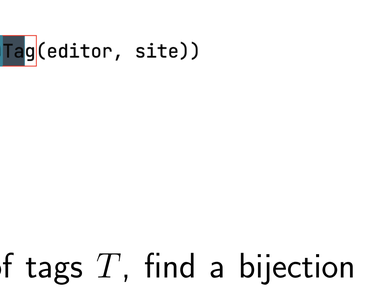
Screenreading, object detection and visual navigation from raw screen pixels.



Activate



Select



Jump

SourceJump

Learn to retrieve contextually relevant source code from an unindexed corpus by learning to write a query which captures semantically similar documents.

Tidyparse

Suggests minimal syntax repairs ranked by naturalness and Levenshtein distance.

Idiolect

Bind custom phrases in a reconfigurable IDE plugin with instant adaptation.