Wall·E

An OpenStreetMap robot

Oliver Kaleske

PTV Group, Karlsruhe

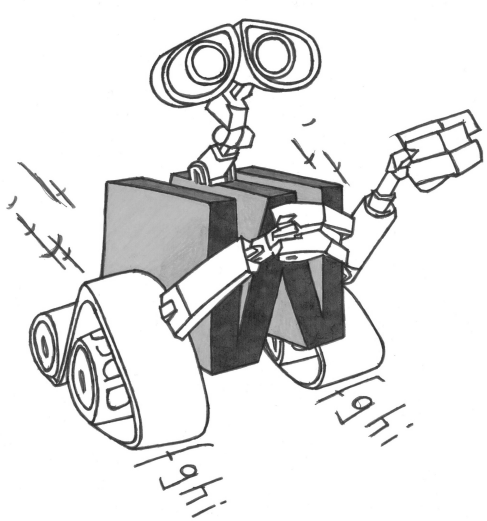
aka Oli-Wan

State of the Map EU 2014
20140614, Karlsruhe

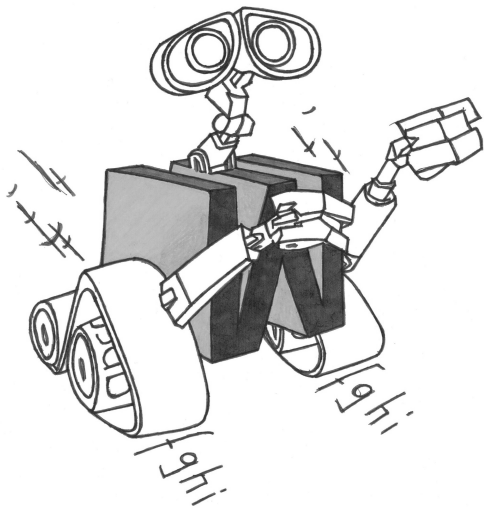
Outline

Wall·E – a short introduction

- robot focused on fixing (obvious) errors; no retagging etc.
- account: https://www.openstreetmap.org/user/Wall·E
- performs various spelling corrections (country-specific), tag corrections, geometry fixes
- started in late 2012, fully operational since early 2013
- currently triggered manually, but can operate fully automatically
- roughly 33k edits to date (cf. hdyc)



Special features



- written in Emacs Lisp (filtering tools in C++ using Osmium)
- running in Germany (and Austria)
- extensive documentation (though mostly in German): http://wiki.openstreetmap.org/wiki/User:Oli-Wan/Wall-E
- conservative: avoids unsafe edits
- goals: improve data quality, and relieve other mappers of trivial QA tasks

Problems addressed

- misspelt/abbrev'd road names
- various addr:* errors
- extra whitespace (tags, roles)
- ways with repeated nodes
- ways with only one node
- misspelt tags

Short history

- inspired by xybot (2008-2012)
- forum thread, informal question: should we set up a replacement for xybot? (Nov 12)
- first task: correction of street names (Dec 12)
- various extensions and new tasks, each discussed and approved on the German forum
- hosted by Wikimedia toolserver until its decommissioning

OSM editing with GNU Emacs

Internal representation: How to store OSM objects in Emacs Lisp?

```
(way 28245343 4 t 10522530 "KrilleOSM" 390669
"2012-01-28T16:34:58Z" (("building" . "yes")
("building:levels" . "6") ("name" . "B")) (310217110
310217160 310217161 310217162 310217163 310217126
310217111 310217112 310217117 310217113 310217110))
```

How to make an edit?

```
(setq object (osm-api-retrieve-node id))
(osm-obj-add-tag object "foo" "bar")
(osm-api-create-changeset "added foo=bar tag to baz")
(osm-api-update-object object) (osm-api-close-changeset)
```

Example: geometry fixes

- reduce repeated nodes in ways: A-B-B-B-C→A-B-C (safe)
- remove ways consisting of only one node
- unsafe, might delete information
- identify safe cases without loss of information:
  - no tags, no history → no information
  - tags duplicated by adjacent way, no history → information conserved
  - way is member of a relation → check relation, other members sharing the node
  - history, but . . . → information conserved
  - . . .
- leave the rest to human mappers

Example: disentangling street and house number

	addr:street	addr:housenumber
correct	foo road	1
swapped	1	foo road
both in street	foo road 1	
both in hno		foo road 1

Example: disentangling street and house number

	addr:street	addr:housenumber
correct	foo road	1
swapped	1	foo road
both in street	foo road 1	
both in hno		foo road 1
duplication	foo road	foo road 1
duplication	foo road 1	1
contradiction	foo road 1	5

Example: disentangling street and house number

case: both in addr:street (no duplication)

- identify substring representing house number...
- basic idea: check for /[[:blank:]]*([[:digit:]]+)$/
- copy \1 to addr:housenumber, then remove \0
- realistic regex: /[[:blank:]]*([[:digit:]]+([[:blank:]]?[[:alpha:]])?|[[:digit:]]+[-\/][[:digit:]]+)[[:blank:]]*$/
- false positive matches: “Weg 1”, “Straße 115”, “An 44”, “R2”, ... ⇒ need cross-check
- solution: check for highway=* with corresponding name nearby (Overpass API)

Example: Misspelt tag keys

- analyze tag keys occurring in the OSM data and count them
- . . . Tracktype tarcktype teacktype trachtype track type tracktpe trackttype tracktxpe trackty tracktye tracktyp tracktype tracktyper tracktypr trakctype traxktype trycktape trycktype zracktype . . .
- form pairs: . . . , (trycktape, tracktype), . . .
- keep only pairs whose second element is much more frequent: ~~(highway, tracktype)~~
- keep only pairs with small editing distance: ~~(tracktype, highway)~~
- review individually $\rightarrow$ filtering and substitution rules
- . . . , Tracktype $\mapsto$ tracktype, tarcktype $\mapsto$ tracktype, . . .

A case for more (maintenance) robots

- my view: we need more robots for maintenance tasks – quality assurance/bugfixing, tagging cleanup?
- Central question: do we want this?
- Key objections:
  - It's easy to break the data. *(technical)*
  - It's easy to harm the community. *(social)*
  - This is not how I/we imagine OSM. *(ideological)*

Why not have mappers do all the maintenance work?

- very few mappers involved in QA
- fixing large numbers of trivial errors time and again can be frustrating
- rapidly growing amount of data; (grey) imports
- continuous and growing influx of erroneous data
  - inexperienced (or one-time) mappers
  - simple non-validating editors (wheelmap, mobile editors)
  - publicity $\rightarrow$ "we need to be in OSM" (or wheelmap)
- low conversion rate of beginners into regular mappers
- mappers alone cannot handle the maintenance work

“It’s easy to break the data” (with a robot)

- plenty of proof exists
- also manual edits/fixes can introduce errors
- robot strictly follows its program

“It’s easy to harm the community” (with a robot)

social risks of automated edits:

- tension within the community as a whole
- scaring away individual contributors

most controversial:

- arbitrary retagging, “deprecation” of tags
- poor execution
- edit wars
- name:*

pure bugfixing much less problematic – but what is an error?

positive community effects?

- make OSM a more enjoyable experience for mappers involved in QA
- having your mistakes fixed by a robot might be less embarrassing
  - no extra work for your fellow mappers
  - mistake was anticipated

A word on policies

- Mechanical Edit Policy in place (see wiki)
- often criticized as being too strict
- does not distinguish “mechanical edits” by
  - size/scope
  - find/replace in JOSM vs. robot
  - 1:1 tag replacement vs. pattern
  - obvious fixes vs. retagging
  - ...
- adequate for full-scale robots

Last slide

Thank you!

- for your attention
- for the constructive feedback I received on Wall·E