IFT Encoder Updates

April 2026

Notable Changes

1. Zero Config Mode
2. Public API and Util Name Cleanup
3. Dependency Graph Condition Extraction (DEP_GRAPH_ONLY)
4. Patch map entry list compilation improvements
5. Performance optimizations

Zero Config Mode

- Previously generating fonts was multiple step process:
  - Hand write “segmenter config”
  - Run segmenter w/ config to generate segmentation plan.
  - Run compiler w/ segmentation plan to create IFT font.
- **Zero config allows you to go from input font to IFT font with no provided config.**
  - Automatically generates a segmenter config and runs all steps
- Optional “quality” parameter which can select a performance/quality tradeoff.
- Old way still works too if more control is needed.

Zero Config Mode

```
font2ift [--auto_config_quality=n]
  --input_font="font.ttf"
  --output_path="out/"
  --output_font="font-ift.woff2"
```

Public API and Util Cleanup

- Renamed many of the utilities:
  - gen_ift_segmenter_config: font -> segmenter config
  - gen_ift_segmentation_plan: font -> segmentation plan
  - font2ift: font -> ift font.
  - Every util supports “auto config” or can be given an explicit config.
- bazel build has been set up to make it explicit what’s public api and not. New APIs for auto config, and running the compiler (mirrors the above utils).

Dependency Graph Condition Extraction

- Full per glyph conditions can now be extracted from the dependency graph, including composite conditions:

  if ((AE OR ae OR AEacute OR aeacute)
  AND (AE OR AEacute OR smcp)
  AND (ae OR aeacute OR c2sc)
  AND (c2sc OR smcp)) then p0

  VS

  if ((AE OR ae OR AEacute OR aeacute OR smcp OR c2sc)) then p0

Dependency Graph Condition Extraction

- Typically faster and more detailed then the closure approach,
  but may overestimate when contextual lookups are involved.

- Default for auto config, can be enabled with new condition analysis mode:
  DEP_GRAPH_ONLY

- Still struggles with some fonts that have highly complex shaping due to condition complexity explosion. More work is needed.

Patch map entry list compilation improvements

- Fixed long standing issue with patch maps that have more than 127 child entries.
- Optimize the compiled entry list for smaller number of encoded bytes.
- 10-15% reduction in raw sizes, and minor improvements in post woff2 sizes.

Performance Optimizations

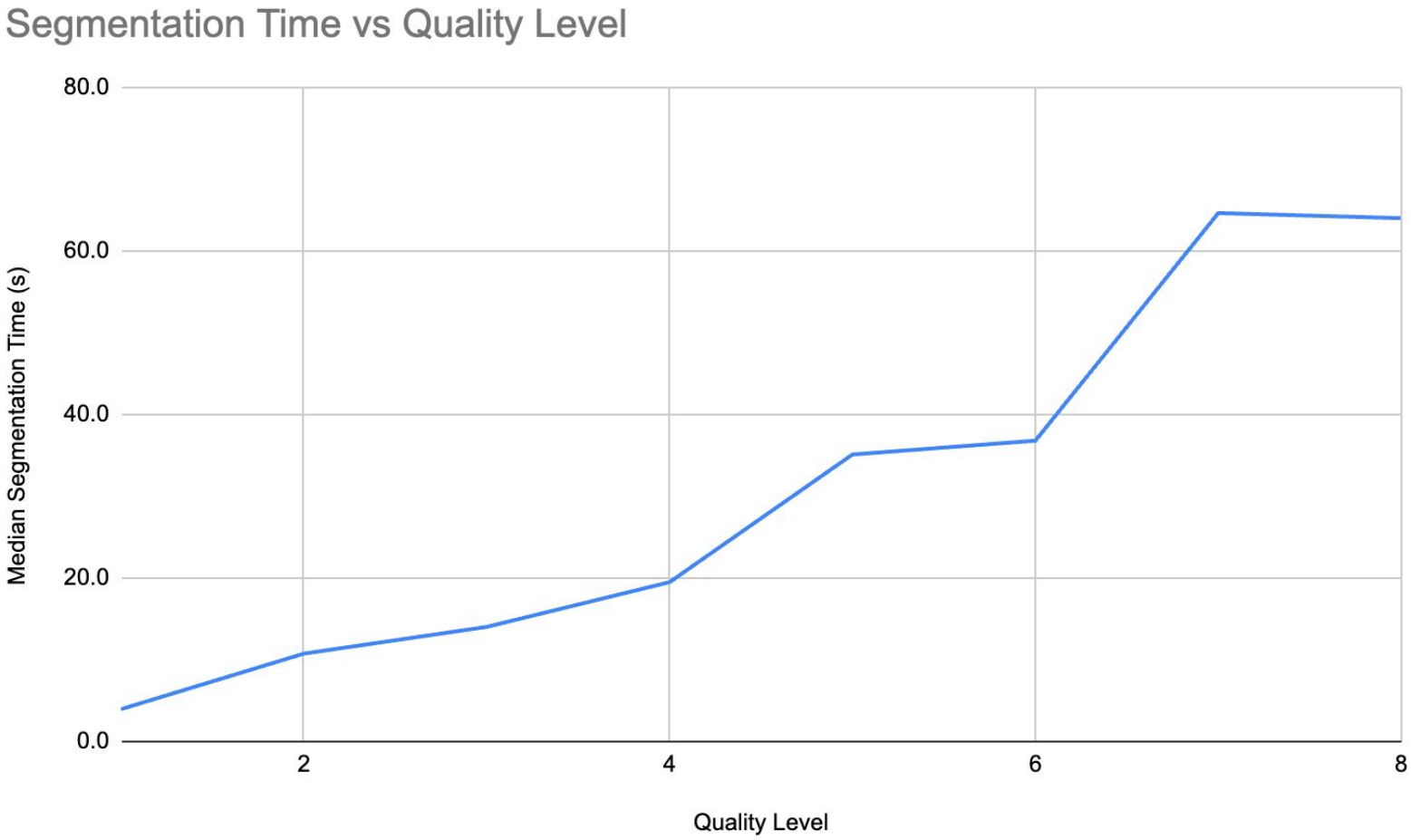
- I’ve got the encoder running across the open source font collection.
- Using that to identify fonts with long encoding times and target optimization work.
- Sped up probability calculations
- Sped up dep graph based condition analysis
- Reduced memory usage

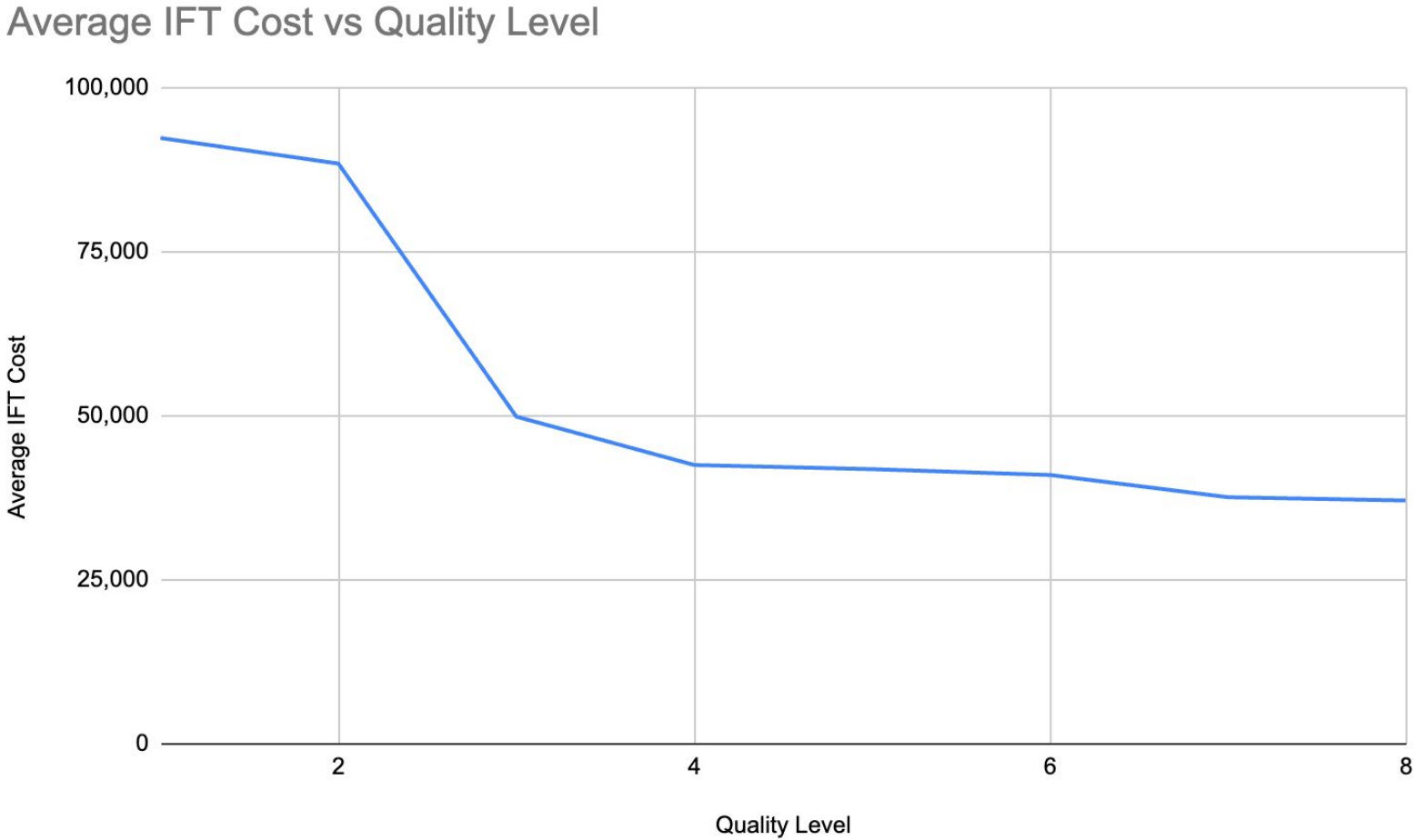
Encoder Stats

- From running encoder across https://github.com/google/fonts
- Excludes anything that takes longer than 30 minutes

  2% of cases timed out

- For those that did not time out, no errors.


Segmentation Time vs Quality Level
80.0
60.0
40.0
20.0
0.0
Median Segmentation Time (s)
2
4
6
8
Quality Level


Average IFT Cost vs Quality Level
100,000
75,000
50,000
25,000
0
Average IFT Cost
2
4
6
8
Quality Level

Auto Pick Quality