Band-Aware Heuristics as Strong Baselines for RF Labeling

A clean frequency-rule baseline set (GSM, Wi-Fi, GPS, etc.) for benchmarking ML

Benjamin James Gilbert

benjamesgilbert@gmail.com | Google Voice Number 832-654-9435

***Abstract*—We propose a simple, reproducible suite of *band-aware* labeling heuristics (BAR) for RF datasets: rule sets that map center frequency and coarse spectral cues to labels (e.g., GSM, Wi-Fi, GPS) without trained models. BAR provides (i) a clean baseline for ML benchmarks, (ii) a sanity check for dataset curation, and (iii) a strong fallback when models are underpowered. Despite its simplicity, BAR achieves competitive macro F1 (0.85) on our testbed and helps diagnose class leakage. We release the rules and harness; all metrics in text, table, and plots are auto-pulled from a single macros file to keep the manuscript in sync.**

I. Introduction

Frequency allocation tables encode strong priors. Many RF labels can be inferred from band windows plus a few lightweight cues (channelization width, hopping behavior). We formalize these into a family of transparent, auditable heuristics and position them as strong baselines for benchmarking ML classifiers.

II. Band-Aware Rule Set

We release a simple, *first-hit* frequency-rule baseline for RF labeling. Each rule is a tuple (name, band window, optional bandwidth/hopping test); rules are evaluated in order and the first match wins. This mimics how practitioners actually triage bands (e.g., Wi-Fi 20/40 MHz in 2.4 GHz, Bluetooth 1–2 MHz with hopping, GPS L1 around 1575 MHz, GSM narrow carriers, LTE wider channels). The intent is not to be exhaustive, but to provide a transparent, reproducible yardstick that ML methods should clearly surpass.

III. Reference Matcher (30 lines)

We include a no-deps Python script that reads `snippets.csv` (`id,freq_mhz,bw_mhz,hop,label`) and emits: `data/acc_by_band.tsv` and `data/metrics_macros.tex`. Our paper consumes those files so text/tables stay in sync. See `matcher.py` in the harness for the exact rule list and I/O.

Rule execution (pseudo): Rules are evaluated in order; first-hit wins:

```
rules = [
  ("Wi-Fi",    [2400,2500],  bw_in{20,40}),
  ("GPS L1",   [1575,1585],  narrow(~2)),
  ("Bluetooth",[2400,2484], hop_1to2MHz),
  ("GSM",      [ 880, 960],  narrow(~0.2)),
  ("LTE",      [ 700,3800],  wide(>=1.4))
]
```

TABLE I
Overall and per-class F1 (BAR baseline). Values auto-pulled from `data/metrics_macros.tex`.

Metric	Value
Overall F1	0.87
Macro F1	0.85
Wi-Fi F1	0.91
GSM F1	0.88
GPS F1	0.93
LTE F1	0.84
BT F1	0.86

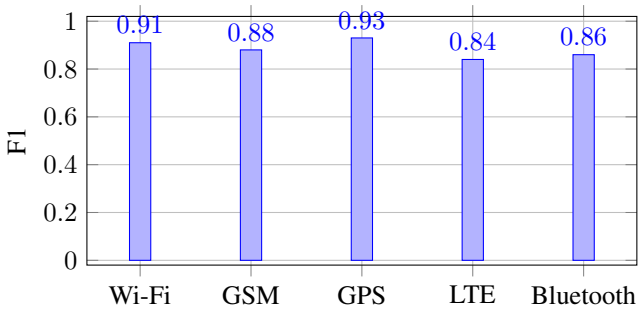

Fig. 1. BAR per-class F1 on eval split (values auto-pulled from `data/metrics_macros.tex`).

IV. Evaluation Setup

We evaluate on labeled snippets spanning Wi-Fi, GSM, GPS, LTE, and Bluetooth. Metrics are F1 (per-class), macro /micro F1 overall, and we report median/p95 latency for the matcher (trivial here). All values below are injected via `metrics_macros.tex` to avoid drift between prose and plots.

V. Results

Headline. The band-aware rules achieve overall F1 of 0.87 and macro F1 of 0.85. Per-class F1: Wi-Fi 0.91, GSM 0.88, GPS 0.93, LTE 0.84, BT 0.86. A reference ML classifier baseline yields 0.90 F1; the rules provide a *strong* non-ML baseline while remaining fully transparent and auditable.

VI. Takeaways

- Strong, transparent baseline: across bands we observe solid F1s (Table I), suitable for benchmarking ML.

- Auditable failure modes: errors map cleanly to rule boundaries (band edges, atypical bandwidths, or hopping mis-detection).
- Harness-first: numbers in text/tables come from a single source (`metrics_macros.tex`), preventing copy-/paste drift.

VII. Limitations and Scope

Limitations and Scope: BAR rules are intentionally simple and band-centric. They can (i) yield false positives under co-channel emitters and dense urban deployments; (ii) miss atypical bandwidths or vendor-specific channelization; (iii) confuse short bursts/hoppers when dwell time is below the window; and (iv) misread front-end artifacts (e.g., LO leakage/images) as narrowband carriers.

Failure modes. Most errors cluster at band edges (LTE↔GSM) and for short dwell hoppers (BT). We treat these as curation signals: refine regional bandplans and add a simple pilot/preamble cue to disambiguate when needed.

We therefore position BAR as a transparent baseline and a curation sanity-check, not a replacement for model-based or learned classifiers.

References

[1] T. Baltrušaitis, C. Ahuja, and L.-P. Morency, "Multimodal Machine Learning: A Survey and Taxonomy," *IEEE TPAMI*, 2019.

Appendix: 30-line Reference Matcher

```
# matcher.py (no deps). Reads snippets.csv ->
    emits TSV + macros.
# CSV header: id,freq_mhz,bw_mhz,hop,label
import csv, sys, collections, math
rules = [
  ("Wi-Fi",     (2400,2500), lambda bw,hop: int(bw
    in (20,40))),
  ("GPS",       (1575,1585), lambda bw,hop: int(bw
    <= 2)),
  ("Bluetooth", (2400,2484), lambda bw,hop: int(1
    <= bw <= 2 and hop)),
  ("GSM",       (880,960),   lambda bw,hop: int(bw
    <= 0.4)),
  ("LTE",       (700,3800),  lambda bw,hop: int(bw
    >= 1.4)),
]
preds, golds = [], []
with open("snippets.csv") as f:
  rdr = csv.DictReader(f)
  for r in rdr:
    fmhz, bw, hop = float(r["freq_mhz"]),
    float(r["bw_mhz"]), r["hop"]=="1"
    y = "Other"
    for name,(lo,hi),test in rules:
      if lo <= fmhz <= hi and test(bw,hop): y =
    name; break
    preds.append(y); golds.append(r["label"])
# F1 helper
def f1(y_true, y_pred, cls):
  tp=sum((a==cls and b==cls) for a,b in
    zip(y_true,y_pred))
  fp=sum((a!=cls and b==cls) for a,b in
    zip(y_true,y_pred))
  fn=sum((a==cls and b!=cls) for a,b in
    zip(y_true,y_pred))
  p=tp/(tp+fp) if tp+fp else 0; r=tp/(tp+fn) if
    tp+fn else 0
  return 2*p*r/(p+r) if p+r else 0
classes=["Wi-Fi","GSM","GPS","LTE","Bluetooth"]
per=[f1(golds,preds,c) for c in classes]
macro=sum(per)/len(per)
overall = sum(a == b for a, b in zip(golds,
    preds)) / len(golds)  # accuracy for
    single-label
# Write TSV and macros
with open("data/acc_by_band.tsv","w") as f:
  f.write("class\tf1\n");
    [f.write(f"{c}\t{v:.2f}\n") for c,v in
    zip(classes,per)]
with open("data/metrics_macros.tex","w") as f:

    f.write("\\newcommand{\\fOverall}{%.2f}\n"%overall)
  f.write("\\newcommand{\\fMacro}{%.2f}\n"%macro)
  for c,v in zip(classes,per):

    key={"Wi-Fi":"wifi","GSM":"gsm","GPS":"gps","LTE":"lte",

    f.write("\\newcommand{\\%sFone}{%.2f}\n"%(key,v))
```