Zoning Tests

Created November 12, 2026
Updated November 16, 2026

Table of contents

1 Provenance **1**

2 Setup **2**

3 Chipmunk benchmarks **3**

4 Okapi rebuild **3**

5 Recall by zone on Enron **7**

6 Five emails **7**

The zoning tests ran on the lab's GPU machine in `~/models/email-zoning`. The folder lives on the CITF analysis server, and the GPU machine mounts it over SSHFS. Chipmunk runs on the CPU. Okapi runs on one GPU.

1 Provenance

Item	Source	Identifier
Chipmunk code and annotations	github.com/webis-de/acl20-crawling-mailing-lists	commit `ecc2c70bb9b38a989fa64e6b6edf51990df52253`
`fasttext-model.bin` (1.78 GB)	files.webis.de/webis-gmane19-model	SHA-256 `c5320b1f93113c8cc5fc86f88abd06891232572451cc97`
`segmenter_hinge.epoch-11.val_loss-0.093.h5` (9.9 MB)	files.webis.de/webis-gmane19-model	SHA-256 `5e32645c43c814d145aaac280b588c4dc410ce44753a69`
XLM-RoBERTa base	huggingface.co/FacebookAI/xlm-roberta-base	revision `e73636d4f797dec63c3081bb6ed5c7b0bb3f2089`

`chipmunk-packages.txt` and `okapi-packages.txt` in the same folder record every installed package version.

2 Setup

The setup installs uv in the home directory and keeps its Python builds and both download caches on the mount.

```
curl -LsSf https://astral.sh/uv/install.sh | sh
mkdir -p ~/models/email-zoning && cd ~/models/email-zoning
export UV_CACHE_DIR=~/models/email-zoning/.cache/uv
export UV_PYTHON_INSTALL_DIR=~/models/email-zoning/.python
export HF_HOME=~/models/email-zoning/.cache/huggingface
git clone https://github.com/webis-de/acl20-crawling-mailing-lists.git
git -C acl20-crawling-mailing-lists checkout ecc2c70bb9b38a989fa64e6b6edf51990df52253
mkdir -p weights && cd weights
curl -fSLO https://files.webis.de/webis-gmane19-model/fasttext-model.bin
curl -fSLO https://files.webis.de/webis-gmane19-model/segmenter_hinge.epoch-11.val_loss-0.09
sha256sum * > SHA256SUMS
cd ..
```

The Chipmunk environment needs four adjustments to the repository's `requirements.txt`. It installs only the packages the segmenter imports. `fasttextmirror` needs `pybind11` to compile without declaring it, so it builds without isolation. TensorFlow 2.2 declares no upper bound on `protobuf` or `numpy` but fails with current releases of both, so the last line pins their 2020 releases.

```
uv venv --python 3.8 chipmunk-env
source chipmunk-env/bin/activate
uv pip install pybind11 setuptools wheel
uv pip install --no-build-isolation "fasttextmirror==0.8.22"
uv pip install "tensorflow==2.2.0" "click==8.1.8" "tqdm==4.70.1" "elasticsearch==6.8.2" "pys
uv pip install "protobuf==3.20.3" "numpy==1.18.5"
uv pip freeze > chipmunk-packages.txt
deactivate
```

The Okapi environment uses PyTorch's CUDA 12.6 build, since the server's driver (565.57) supports CUDA up to 12.7.

```
uv venv --python 3.11 okapi-env
source okapi-env/bin/activate
uv pip install --torch-backend cu126 "torch==2.14.1" "transformers==5.22.2" "pytorch-crf==0.
uv pip freeze > okapi-packages.txt
echo e73636d4f797dec63c3081bb6ed5c7b0bb3f2089 > xlmr-revision.txt
timeout 900 python3 - <<'EOF'
from huggingface_hub import snapshot_download
```

```
rev = open("xlmr-revision.txt").read().strip()
snapshot_download("FacebookAI/xlm-roberta-base", revision=rev, allow_patterns=["*.json", "*.
EOF
deactivate
```

3 Chipmunk benchmarks

The repository's own `evaluate` command scores the trained segmenter on the 300 Gmane validation emails and the 296 Enron emails. The paper reports its 96% on the same 300 emails.

```
cd ~/models/email-zoning
source chipmunk-env/bin/activate
cd acl20-crawling-mailing-lists
CUDA_VISIBLE_DEVICES="" timeout 3600 ./run.sh src/parsing/message_segmenter.py evaluate \
  ../weights/segmenter_hinge.epoch-11.val_loss-0.093.h5 ../weights/fasttext-model.bin \
  annotations/annotations-final-validation.jsonl 2>&1 | tee ../eval-gmane-validation.log
CUDA_VISIBLE_DEVICES="" timeout 3600 ./run.sh src/parsing/message_segmenter.py evaluate \
  ../weights/segmenter_hinge.epoch-11.val_loss-0.093.h5 ../weights/fasttext-model.bin \
  annotations/enron-annotated.jsonl 2>&1 | tee ../eval-enron.log
```

4 Okapi rebuild

The Okapi paper specifies the encoder and the sequence model. We chose the remaining settings.

Setting	Okapi paper	Our rebuild
Line embedding	XLM-RoBERTa base, frozen; last four layers concatenated per token and averaged over the line (3,072 features)	Same; lines cut at 128 subword tokens
Sequence model	BiLSTM, one layer, 64 units; CRF output	Same
Dropout and optimizer	0.25; RMSprop, learning rate 0.001	Same
Training emails	Chipmunk's English Gmane annotations	`annotations-final-train.jsonl` (2,733 emails), split 90% training and 10% development with seed 13

Setting	Okapi paper	Our rebuild
Labels	Chipmunk's 15 zones	The 15 zones plus `<empty>` for blank lines, assigned with the Chipmunk repository's `annotation_dict_to_lines`, so scores compare with Chipmunk's `evaluate`
Batch size and stopping	Not reported	8 emails per batch; up to 50 epochs, stopping after 5 epochs without improvement on the development emails (training stopped after epoch 14 and kept epoch 9)

`okapi.py` embeds every line, trains the model, saves it as `okapi-model.pt`, and writes `okapi-eval-gmane-validation.txt` and `okapi-eval-enron.txt`.

```
cd ~/models/email-zoning
export HF_HOME=~/models/email-zoning/.cache/huggingface
source okapi-env/bin/activate
CUDA_VISIBLE_DEVICES=0 timeout 7200 python3 okapi.py 2>&1 | tee okapi-train.log
```

okapi.py

5 Recall by zone on Enron

On Enron, the two models recall the zones that matter for us at similar rates. Both read about a quarter of mua_signature lines as paragraph, which would send banners and disclaimers into the text DocuScope tags.

Zone	Chipmunk	Okapi
paragraph	0.94	0.92
salutation	0.81	0.84
closing	0.90	0.90
personal_signature	0.85	0.79
mua_signature	0.71	0.72
quotation	0.97	0.94

6 Five emails

`five-emails.txt` holds the first five emails of the 30-email sample. This block splits it into one body per email in `emails/`, named by msgid2, and saves the header fields to `emails/headers.json`. Each body is everything below the line, unchanged.

```
cd ~/models/email-zoning
source okapi-env/bin/activate
timeout 60 python3 - <<'EOF'
import json, re
from pathlib import Path
raw = Path("five-emails.txt").read_text()
blocks = re.split(r"^(?=Email \d+ of \d+$)", raw, flags=re.M)
out, headers = Path("emails"), {}
out.mkdir(exist_ok=True)
for b in (b for b in blocks if b.strip()):
    head, body = re.split(r"^ {20,}\n", b, maxsplit=1, flags=re.M)
    fields = dict(l.split(": ", 1) for l in head.strip().split("\n")[1:])
    fields["Email"] = head.split("\n", 1)[0]
    msgid = fields["msgid2"]
    (out / f"{msgid}.txt").write_text(body.rstrip("\n") + "\n")
    headers[msgid] = fields
    print(msgid, fields["Email"], "|", len(body.strip().split("\n")), "lines")
(out / "headers.json").write_text(json.dumps(headers, indent=2, ensure_ascii=False))
EOF
deactivate
```

Chipmunk labels each line with the repository's own functions and writes `zones-chipmunk.jsonl`. `chipmunk_raw` holds the model's top zone for each

line; `chipmunk` holds the same after the repository's `_post_process_labels` corrections.

```
cd ~/models/email-zoning
source chipmunk-env/bin/activate
CUDA_VISIBLE_DEVICES="" TF_CPP_MIN_LOG_LEVEL=3 PYTHONPATH=acl20-crawling-mailing-lists/src P
import json
from pathlib import Path
import numpy as np
from tensorflow.keras import models
from util.mail_classification import load_fasttext_model, MailLinesSequence
from parsing.message_segmenter import CONTEXT_SHAPE, INF_BATCH_SIZE, _post_process_labels

load_fasttext_model("weights/fasttext-model.bin")
model = models.load_model("weights/segmenter_hinge.epoch-11.val_loss-0.093.h5")
headers = json.loads(Path("emails/headers.json").read_text())
rows = []
for msgid, h in headers.items():
    text = Path(f"emails/{msgid}.txt").read_text().rstrip("\n")
    seq = MailLinesSequence(text, CONTEXT_SHAPE, labeled=False, input_is_raw_text=True, batc
    soft = model.predict(seq)
    raw = [MailLinesSequence.LABEL_MAP_INVERSE[int(np.argmax(s))] for s in soft]
    post = [label for _, label in _post_process_labels(seq, soft.copy())]
    print(f"\n{h['Email']} (msgid2 {msgid})")
    for i, (line, r, p) in enumerate(zip(text.split("\n"), raw, post), 1):
        rows.append({"msgid2": msgid, "line": i, "text": line, "chipmunk_raw": r, "chipmunk"
        print(f"{i:>3}  {r:<19}{p:<19}{line[:70]}")
Path("zones-chipmunk.jsonl").write_text("".join(json.dumps(r, ensure_ascii=False) + "\n" for
EOF
deactivate
```

Okapi loads `okapi-model.pt`, labels each line, and merges its labels with Chipmunk's into `zones.jsonl`. Each embedding passes through half precision before the BiLSTM, as in training.

```
cd ~/models/email-zoning
source okapi-env/bin/activate
export HF_HOME=~/models/email-zoning/.cache/huggingface
CUDA_VISIBLE_DEVICES=0 timeout 900 python3 - <<'EOF'
import json
from pathlib import Path
import torch
from torch import nn
from torchcrf import CRF
from transformers import AutoTokenizer, AutoModel

LABELS = ["paragraph", "closing", "inline_headers", "log_data", "mua_signature", "patch", "p
```

```
          "quotation", "quotation_marker", "raw_code", "salutation", "section_heading", "tab
          "visual_separator", "<empty>"]
REV = Path("xlmr-revision.txt").read_text().strip()
dev = torch.device("cuda")
tok = AutoTokenizer.from_pretrained("FacebookAI/xlm-roberta-base", revision=REV)
enc = AutoModel.from_pretrained("FacebookAI/xlm-roberta-base", revision=REV).to(dev).eval()


class Okapi(nn.Module):
    """Same architecture as okapi.py."""
    def __init__(self):
        super().__init__()
        self.lstm = nn.LSTM(3072, 64, num_layers=1, batch_first=True, bidirectional=True)
        self.drop = nn.Dropout(0.25)
        self.out = nn.Linear(128, len(LABELS))
        self.crf = CRF(len(LABELS), batch_first=True)

    def emissions(self, x):
        return self.out(self.drop(self.lstm(x)[0]))


@torch.no_grad()
def embed(lines):
    b = tok(lines, padding=True, truncation=True, max_length=128, return_tensors="pt").to(de
    x = torch.cat(enc(**b, output_hidden_states=True).hidden_states[-4:], dim=-1)
    m = b["attention_mask"].unsqueeze(-1).float()
    return ((x * m).sum(1) / m.sum(1)).half().float()


model = Okapi().to(dev)
model.load_state_dict(torch.load("okapi-model.pt"))
model.eval()

rows = [json.loads(l) for l in open("zones-chipmunk.jsonl")]
headers = json.loads(Path("emails/headers.json").read_text())
for msgid, h in headers.items():
    mine = [r for r in rows if r["msgid2"] == msgid]
    x = embed([r["text"] for r in mine]).unsqueeze(0)
    pred = model.crf.decode(model.emissions(x), torch.ones(x.shape[:2], dtype=torch.bool, de
    print(f"\n{h['Email']} (msgid2 {msgid})")
    print(f"{'':>3}  {'chipmunk_raw':<19}{'chipmunk':<19}{'okapi':<19}text")
    for r, p in zip(mine, pred):
        r["okapi"] = LABELS[p]
        print(f"{r['line']:>3}  {r['chipmunk_raw']:<19}{r['chipmunk']:<19}{r['okapi']:<19}{r
Path("zones.jsonl").write_text("".join(json.dumps(r, ensure_ascii=False) + "\n" for r in row
```

```
EOF
```