60-second takeaway

`openai/privacy-filter` runs comfortably on a 16 GB M2 MacBook Pro (2.8 GB BF16, ~1 s median inference on MPS, zero false positives across nine decoy prompts). It catches OpenAI, Anthropic, GitHub, Stripe, Slack, Google, Clerk, and Postgres-with-password secrets, plus names/emails/phones in English and common Asian languages. It misses AWS `AKIA...` keys, MongoDB and Redis connection URIs, Arabic and West-African names, and UK-style landline numbers. **Do not use it as the sole defense for credentials.** It earns its lane as a second layer behind deterministic regex, not as a replacement for one.

Where this fits

- **For founders:** If you are worried about pasting internal docs (meeting notes, support tickets, customer emails) into a cloud LLM, this model is a cheap, on-device first pass that catches most incidental PII. It is **not** safe as the only layer protecting API keys or database credentials.
- **For engineers:** Use this page to decide whether `openai/privacy-filter` fits in your redact-before-send pipeline, what you still need regex for, and what it costs to run on Apple Silicon.

1 What we tested and why

OpenAI released `openai/privacy-filter` as a 1.5B-parameter mixture-of-experts token classifier (50M active per token, 128 experts top-4) that tags spans into eight privacy categories: `account_number`, `private_address`, `private_email`, `private_person`, `private_phone`, `private_url`, `private_date`, and `secret`. The model card positions it for "high-throughput data sanitization workflows" and claims it can run in a browser.

The interesting question for us is narrower: **can it be the redact-before-read layer in front of a cloud coding assistant, so that secrets never leave the laptop?** That is a strictly harder bar than "can it find names in a document" - a single missed API key is a rotation event.

We built a 52-case suite covering:

- **Credentials** - OpenAI, Anthropic, GitHub classic + fine-grained PATs, AWS access key + secret, Stripe live key, Slack bot token, Google API key, Clerk secret, RSA and OpenSSH private key blocks, signed JWT, Postgres/MongoDB/Redis connection URIs with embedded passwords.
- **PII** - names across six cultures (Western, Chinese-Singaporean, Indian, Arabic, Nigerian, Japanese), emails with `+alias` and `.co.uk` TLDs, phones in Singapore/UK/US formats, addresses in three jurisdictions, birthdays, credit card, UK IBAN, US SSN, routing/account, reset links with tokens, S3 presigned URLs.
- **Decoys** - public executive names, corporate support emails, toll-free numbers, Apple Park HQ, tech acronyms, env variable names (without values), and plain business prose.
- **Mixed documents** - a realistic `.env.local` excerpt and a meeting note combining names, emails, phones, addresses, dates, and card numbers.

2 Setup on an M2 MacBook Pro, 16 GB RAM

Three environment notes worth calling out:

1. **`transformers 5.6.0` or newer is required.** The model ships `model_type: "openai_privacy_filter"` in its config and carries no `modeling_*.py` fallback. Stock `transformers 5.4.0` throws `KeyError: 'openai_privacy_filter'`. Upgrade via `pip install --upgrade "transformers>=5.6.0"`. PyPI has 5.6.1 as of this writing.
2. **MPS is picked automatically.** Passing `device_map="auto"` (or the default `pipeline()` behavior) selects `mps:0` on Apple Silicon. CPU also works but is roughly 2-3x slower.
3. **Cold load is slow, warm load is fast.** First run downloads 2.8 GB from HuggingFace Hub and spent 123 s end-to-end on our connection. Warm load (weights cached in `~/.cache/huggingface/`) completed in 7.5 s.

Minimal invocation:

```
from transformers import pipeline
clf = pipeline(
    "token-classification",
    model="openai/privacy-filter",
    aggregation_strategy="simple",
    trust_remote_code=True,
)
clf("Email me at alice@acme.com")
```

Memory footprint on the machine during inference stayed well under 6 GB, leaving plenty of headroom on a 16 GB laptop.

3 Category results

Hit rate counts a case as **HIT** only if the expected category fired on the correct span. Decoy cases count as **OK** when nothing fires. Mixed documents count as HIT when any detection fires.

Category	n	Hit rate	Notable outcome
API keys and tokens	10	80.0%	AWS access key + secret both missed
PEM private key blocks	2	100.0%	RSA and OpenSSH both caught
JWT	1	100.0%	Full Bearer token flagged
Connection URIs with creds	3	33.3%	Mongo SRV and Redis URIs missed
Personal names	6	66.7%	Arabic and Nigerian names missed
Emails	3	66.7%	Two adjacent emails only catches the first
Phones	3	66.7%	UK landline mis-tagged as account + secret
Addresses	3	66.7%	London "221B Baker Street" missed standalone
Dates (birthdays)	2	100.0%	ISO and long-form both caught
Account numbers	4	100.0%	Card, IBAN, SSN, ACH all caught
URLs with sensitive params	2	50.0%	S3 presigned URL missed
Non-English PII	2	100.0%	Chinese name + Japanese email caught
Decoy (public + code + prose)	9	100.0%	Zero false positives
Mixed documents	2	100.0%	Multi-category in-context recall

Overall: 38 of 50 single-category cases classified correctly (76%), plus 9 of 9 decoys rejected cleanly and 2 of 2 mixed documents hit on multiple categories.

4 What it caught well

4.1 Modern SaaS API key formats

OpenAI project keys (`sk-proj-...`), Anthropic keys (`sk-ant-api03-...`), Stripe live keys (`sk_live_...`), Slack bot tokens (`xoxb-...`), Google API keys (`AIzaSy...`), Clerk

secrets, GitHub classic (`ghp_...`) and fine-grained (`github_pat_...`) tokens all fire at confidence 1.00. The model has clearly seen these prefixes during training.

4.2 Cryptographic blocks

Both `-----BEGIN RSA PRIVATE KEY-----` and `-----BEGIN OPENSSH PRIVATE KEY-----` blocks are flagged cleanly, as is a signed JWT in `Authorization: Bearer ...` form. The fixed structural prefix seems to be doing the work here.

4.3 Zero false positives on decoys

This is the quietest but most important finding. Across nine decoy prompts - public executives, corporate support emails, a 1-800 number, the Apple Park HQ address, `NODE_ENV=production`, and plain revenue prose - the model fired zero spurious detections. In redaction pipelines, false positives are often worse than false negatives because they corrode user trust until people disable the filter.

4.4 In-context recall beats standalone

The meeting-note test caught "221B Baker Street" as an address, "Wei Jie Chee" as a person, "+91 98450 12345" as a phone, and "2026-04-30" as a date all at confidence 1.00. Many of these same spans in isolation (tested alone without surrounding context) performed worse or missed. The model uses neighboring tokens as evidence, which is the right behavior but also means short snippets underperform longer documents.

5 What it missed - and why that matters

5.1 AWS access keys (the big miss)

```
AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

Neither the `AKIA...` access key id nor the 40-character base64-ish secret fires in either test. AWS credentials are the single most common cloud secret in infrastructure code. For anyone thinking of using this model to strip secrets from `.env` files or Terraform state before pasting into a chat, this is a disqualifying gap. The regex `AKIA[0-9A-Z]{16}` would have caught it deterministically in microseconds.

5.2 MongoDB and Redis connection URIs

```
mongodb+srv://dba:Mx7!pQ@cluster0.mongodb.net/app
redis://default:HeLloW0rld@redis-123.cloud.com:6379
```

Both fail to fire. The Postgres URL `postgresql://admin:s3cretP4ss@...` was caught - the model appears trained on that specific prefix but generalizes poorly to other URI schemes. Any production environment runs more than one datastore.

5.3 Names outside the apparent training distribution

"Alice Smith", "Wei Jie Chee", "Rajesh Venkataraman", and "Akira Tanaka" all fire. "Fatima Al-Rashid" and "Chinedu Okonkwo" do not. This matches the model card's warning about uncommon names and regional conventions, but the practical consequence is that redaction coverage is uneven across your user base. A European-Chinese-Japanese-Indian-skewed corpus will look cleaner than it really is.

5.4 UK-style phone numbers without a country prefix

```
Dial 020 7946 0018 for the London office.
```

The model tags "020" as `account_number` and "7946 0018" as `secret`. Complete mis-categorization. The Singapore and US formats with a leading `+` or parenthesized area code worked. Evidently the model uses the `+XX` prefix as a strong signal, and absent it falls back on formatting heuristics that fail on UK layouts.

5.5 S3 presigned URLs

```
https://bucket.s3.amazonaws.com/file.png?X-Amz-Signature=ABCDEF
```

No detection. A presigned URL is effectively a short-lived credential - missing it here is concerning for any workflow that handles shared upload links.

5.6 Two adjacent emails

```
Invoices go to billing@startup.io, ar@startup.io.
```

Only `billing@startup.io` is flagged. The second email is dropped. This is fixable with post-processing (split on `,` and re-run), but out of the box it is a coverage gap worth knowing about.

6 Latency on Apple Silicon MPS

Metric	Value

Device	`mps:0` (Apple M2 GPU)
Weights	2.8 GB (BF16)
Cold load (with download)	123 s
Warm load (cache hit)	7.5 s
Per-case median	958 ms
Per-case p90	1,725 ms
Per-case max	3,039 ms
Total for 52 cases	58.3 s

Roughly 60 documents per minute at our sample length on a 16 GB M2. Long enough that you would not want it inline in a streaming chat response, short enough that batch-scanning a folder of meeting notes before feeding them to a cloud model is fine. CPU-only runs were 2-3x slower in quick spot checks; if you are on Apple Silicon, keep MPS on.

7 The architecture this benchmark argues for

The clean signal from 52 cases is that `openai/privacy-filter` has two genuine strengths (soft PII in well-formed English and discipline against false positives) and two genuine weaknesses (novel secret formats and names outside common distributions). The right way to use it is as the second layer in a two-stage redactor:

Layer 1 - Deterministic regex:

```
AKIA[0-9A-Z]{16}                  # AWS access key id
[A-Za-z0-9/+=]{40}                # AWS secret (needs context signal)
ghp_[A-Za-z0-9]{36}               # GitHub PAT
github_pat_[A-Za-z0-9_]{82}       # GitHub fine-grained PAT
sk-(?:proj-|ant-api03-|live_)...  # provider-specific key prefixes
xox[baprs]-[A-Za-z0-9-]+          # Slack tokens
AIza[0-9A-Za-z_-]{35}             # Google API key
eyJ[A-Za-z0-9_-]+\.eyJ[...]\....  # JWT shape
-----BEGIN [A-Z ]+PRIVATE KEY-----
mongodb(\+srv)?://[^\s]+@          # any Mongo URI with creds
redis(s)?://[^\s]+@                # any Redis URI with creds
```

These catch the failure modes the model has. They run in microseconds. They have no learned bias.

Layer 2 - `openai/privacy-filter`:

The model then handles the soft PII the regex layer cannot - names, addresses, free-form phone numbers, birthdays, and in-document correlations that a regex will never catch without exploding in false positives.

Layer 0 (always): Never feed known-secret files (`.env*`, plists with `EnvironmentVariables`, keychain exports) to either layer. Ask for the specific non-secret fields instead. This is strictly safer than any classifier.

8 When to use this model

Use it for:

- Meeting notes, support tickets, customer emails, transcripts, social DMs - content that is not expected to contain credentials but often contains incidental PII.
- Batch redaction of a document set before feeding any of it into a cloud LLM.
- In-browser redaction (via Transformers.js + WebGPU + q4 quantization) where the data cannot leave the tab.

Do not use it for:

- `.env` files, Terraform state, Kubernetes manifests, CI secret exports, database dumps. Regex and "do not read this file at all" are safer.
- Anything involving AWS credentials as the sole layer.
- Non-English corpora without explicit per-language validation.
- Short, context-free snippets - the model leans on neighboring tokens.

9 Reproducibility

Full 52-case suite (Python, no external deps beyond `transformers` and `torch`) and the JSON result dump are available on request. Machine: M2 MacBook Pro, 16 GB unified memory, macOS 15.4. Stack: `torch==2.11.0`, `transformers==5.6.1`, Python 3.10, PyTorch MPS backend. All model runs used `aggregation_strategy="simple"` with default thresholds.

10 The honest read

OpenAI shipped a well-calibrated, laptop-sized PII classifier that refuses to cry wolf and handles common English PII cleanly. That is genuinely useful. The framing on

the model card - "privacy filter" - oversells what a single neural model can do against a determined secret-format. The model's own documentation admits this ("may miss novel secret formats") and the test suite confirms it on the two formats that matter most in practice, AWS keys and non-Postgres connection URIs.

Treat it as a smart second pass, not a firewall. Put deterministic regex in front of it for credentials. Keep your "never read that file" rules in place for the files you already knew not to touch. Inside those guard rails, this is a small, fast, on-device model that earns its spot in a privacy-by-design pipeline.