TranceLate — Abschluss- & Übergabebericht (Self-host MT Stack)

Datum: 26.08.2025 (Europe/Vienna)
Projekt: Self-hosting Übersetzungsplattform „TranceLate“ (Guard + TM + Polish + Crawling)
Beteiligte: Marisa (TranceLate), TranceLate (GPT-Assistenz)

---

1) Executive Summary

Wir haben eine lokal laufende, robuste Übersetzungs-Pipeline aufgebaut, die ohne Cloud-Provider auskommt und trotzdem Provider-Niveau bei UI-Texten liefert. Kernprinzip: **Modelle machen Grammatik, wir machen Governance** – also Regeln & Daten (TM/Glossar/OrgCard) plus technische Schutzmechanismen.

Highlights - **Self-host MT** (Uvicorn/Transformers) + **Guard** (FastAPI) mit Invariantenprüfungen.
- **Translation Memory (TM)**: exact + **fuzzy** (rapidfuzz, Schwelle 0.92), Live-Reload via `/admin/reload`.
- **Polish** (Tone-only) mit `POLISH=auto`: läuft nur, wenn TM „miss“ ist; Platzhalter/HTML garantiert unverändert.
- **Quality-Checks** in Responses (JSON): Platzhalter/Zahlen/HTML/Klammern/Längen-Verhältnis, inkl. **AM/PM→24h**-Heuristik.
- **Crawling/Parsing**-Tooling (Homepage-tauglich), plus **satzweise Segmentierung** für verlustfreie Übersetzung.
- **Reproduzierbare Umgebung** (`env_tl311.yml`), **Snapshot** (tgz) & **README** (Ops Guide).

Ergebnisse (Auszug) - Proof-Set & Acceptance-Smoke: Platzhalter 100% korrekt; TM exact/fuzzy hoch; Polish verändert TM-Treffer nicht.
- Crawl python.org (20 URLs → 776 Segmente): 7 problematische Fälle (Zeiten, Kurz-Titel, Zahlenerhalt) → Guard-Checks erweitert, Satz-Splitter implementiert; Restprüfungen stehen als letzte Validierung an.

---

2) Architektur & Komponenten

Übersicht - **mt_server (8090)**: Self-host MT (Transformers Pipeline).
- **mt_guard (8091)**: FastAPI-Gateway mit TM/Glossar, Fuzzy-Matching, Invarianten-Checks, Cleanup & Normalisierung.
- **Polish**: LLM-Endpoint (lokal/remote), deterministisch (`temperature=0`).
- **Tools**: CLI (`t.sh`, `tp.sh`, `tq.sh`), Batch (`batch_tq.py`), Quality-Gate (`qgate.py`), Crawling/Parsing (`fetch_clean.py`, `list_urls.py`, `crawl_fetch_clean.py`, `split_sentences.py`).

Datenflüsse 1. Request → **Guard** `/translate`
2. **TM exact** → Treffer? → Response
3. **TM fuzzy** (0.92, Placeholder-Set identisch) → Response
4. Sonst: **Placeholder & never_translate schützen** → **MT** (8090) → **Restore & Cleanup**
5. **EN sentence-case** nach Placeholder/Tags/Dash
6. **Checks** berechnen → Response (`translated_text`, `provenance`, `checks`).

Governance-Bausteine - **TM** (`tm.csv`): verifizierte Einträge, inkl. DU/SIE-Stil; Fuzzy erkent nahe Varianten sicher.
- **Glossar** (`glossary.json`): `never_translate`-Liste (Marken/Produkte) – garantiert unveränderte Terme.
- **OrgCard** (`orgcard.json`): Stil/Anrede; Polish nutzt Regeln, ohne Platzhalter/HTML zu verändern.

---

3) Installation & Umgebung

- Conda-Env `tl311` (Python 3.11), **Torch ≥ 2.6** (Fix für CVE-2025-32434; required by Transformers), Transformers 4.55.x.
- Snapshot erstellt: `trancelate_selfhost_20250826_0203.tgz` (Configs, TM, Scripts, Reports).
- Repo initialisiert (Git) + `.gitignore`.

Restore (Kurz)

```
# im Projektordner
conda env create -f env_tl311.yml || conda env update -f env_tl311.yml
./start_local.sh   # startet 8090 + 8091
curl -s http://127.0.0.1:8091/health  # => {"ok":true}
```

---

4) Betrieb (Runbook)

Start/Stop

```
./start_local.sh    # MT (8090) & Guard (8091)
./stop_local.sh     # beendet beide
```

Einfacher Call (Guard)

```
./t.sh de en 'TranceLate Pro synchronisiert {{COUNT}} Seiten.'
```

Mit Polish (auto)

```
./tp.sh de en '<strong>Jetzt starten</strong> – {{COUNT}} Seiten
synchronisiert.'
# Ergebnis: HTML/Platzhalter unverändert, sentence case korrigiert („pages“)
```

Quality-erzwungene Übersetzung

```
./tq.sh de en '<strong>…</strong>'  # bricht ab, wenn checks.ok=false
```

Endpoints - `POST /translate` → `{ translated_text, provenance, checks }`
- `GET /health` → `{ ok: true }`
- `GET /meta` → `{ tm_entries, tm_soft_threshold, provider_configured, never_terms }`
- `POST /admin/reload` → TM/Glossar ohne Neustart laden

---

5) Qualitätssicherung

Checks (Response) - `ph_ok` **(Platzhalter)**: exakt gleich, unverändert.
- `num_ok` **(Zahlen)**: digits-only-Vergleich; **AM/PM→24h** wird toleriert (z. B. „4pm“→„16“).
- `html_ok` **(Tags)**: gleiche Multimenge der HTML-Tags.
- `paren_ok` **(Klammern)**: keine neuen Klammern bei Quellen ohne Klammern.
- `len_ratio` **(Länge)**: 0.5–2.2 (≥20 Zeichen), 0.4–4.0 bei sehr kurzen Quellen.

Polish - `POLISH=auto` (Default in `tp.sh`): nur bei TM-Miss.
- Strikte Policies: **keine** neuen Fakten, **keine** Klammern, Platzhalter/HTML tabu.
- **DU-Fix optional** via `DU_FIX=on` (sonst Stil via TM/OrgCard).

TM fuzzy - Schwelle 0.92 (anpassbar: `TM_SOFT_THRESHOLD`).
- Nur, wenn Placeholder-Set identisch bleibt.

---

6) Crawling & Parsing

Einzel-Seite - `fetch_clean.py`: Readability + jusText, Segmentierung & Dedupe, JSONL-Output.

Mehrere Seiten - `list_urls.py` → `urls.txt` (gleicher Host, Menü/Breadcrumbs ausgeschlossen).
- `crawl_fetch_clean.py` → kombiniertes JSONL (Pro-Seite `meta` + Segmente).
- `split_sentences.py` → **satzweise** Segmente (verhindert Verlust von Zahlen/„Python 3“ etc.).

Batch-Übersetzung

```
jq -c 'select(has("text")) | {source:"en",target:"de",text:.text}'
python_crawl.jsonl > req.jsonl
./batch_tq.py req.jsonl out.jsonl  # schreibt je Zeile Ergebnis + checks
```

---

7) Messwerte & Akzeptanz

Smoke/Proof - Platzhalter OK (MT): 100% - Platzhalter OK (Polish): 100% - TM exact: sehr hoch (UI-Microcopy), TM fuzzy: sinnvoll strenger Match - Polish unverändert bei TM-Treffern

Leistung - MT p50 ~17 ms pro Segment (lokal, Einzelaufrufe).
- Polish p50 ~4.6 s (nur bei TM-Miss aktiv, deshalb gesamt selten).
- Batch (776 Segmente) sequentiell: einige Minuten erwartbar → **Parallel-Runner** geplant.

Großer Crawl - Ausgang: 776 Segmente, 7 Fails (kurze Titel, AM/PM, Zahlenerhalt).
- Maßnahmen: Numbers-Check + AM/PM-Heuristik, Satz-Splitter → **Recheck-Batch vorbereitet**.

---

8) Sicherheit & Datenschutz

- **Vulnerabilities**: Torch ≥ 2.6 verpflichtend (Transformers-Safety beim Laden).
- **Lokalbetrieb**: Keine Daten an Dritte; optionaler Provider-Fallback derzeit deaktiviert.
- **Glossar**: Nie-übersetzen-Liste verhindert Brand-/PII-Verfälschung.
- **Logs**: `logs/` in `.gitignore` (keine sensiblen Texte im Repo).

---

9) Risiken & bekannte Grenzen

- **Parallelität**: Batch aktuell sequentiell → Laufzeit höher; Parallel-Runner noch nicht integriert.
- **Nummern-Heuristik**: AM/PM behandelt, andere Formate (Zeitzonenstrings) ggf. erweitern.
- **HTML**: Tag-Multimenge wird gesichert; Attribut-Normalisierung/Links (href) werden derzeit nicht validiert.

---

10) Empfehlungen (Next Steps, priorisiert)

1. **Parallel-Batchrunner** (8–16 Threads) → 5–10× schneller.
2. **Memo-Cache** im Guard (In-Memory/SQLite) → doppelte Sätze ≈ 0 ms.
3. **Provider-Backup** als Failover setzen (nur bei 8090-Down).
4. **Glossar** weiter füllen (Marken, Produktnamen, kritische Phrasen).
5. **TM-Kern** aus UI-Microcopy erweitern; Import aus CSV/Sheets.
6. **QE-Erweiterung**: Zahl/Datum/Währung-Normalisierungen (z. B. 1,234.56 ↔ 1.234,56).
7. **Sitemap-Support** im Crawler; Robots/Rate-Limit; Retry/Backoff.
8. **Monitoring**: einfache Metriken (p50/p95) & Fehlerzähler als Prometheus-Endpoint.

---

11) Datei-Inventar (Kern)

- `mt_guard.py` — Guard-API (TM, Glossar, Fuzzy, Checks, Cleanup, sentence-case).
- `polish.py` — Tone-only Polish, placeholder/HTML-sicher, DU-Fix optional.
- `t.sh` / `tp.sh` / `tq.sh` — CLI (Guard / Polish / Quality-erzwungen).
- `tm_upsert.py` — TM-Insert/Update; Validierung des Placeholder-Sets.
- `tm.csv` — Translation Memory.
- `orgcard.json` — Stil/Voice/Anrede.
- `glossary.json` — `never_translate`-Liste.
- `fetch_clean.py`, `list_urls.py`, `crawl_fetch_clean.py`, `split_sentences.py` — Crawling/Parsing.
- `batch_tq.py` — Batch-Übersetzung (JSONL in/out).

- `qgate.py` — Offline-Quality-Gate über JSONL.
- `start_local.sh`, `stop_local.sh` — Start/Stop.
- `env_tl311.yml` — Conda-Umgebung (reproduzierbar).
- `README_selfhost.md` — Ops-Guide.
- `trancelate_selfhost_20250826_0203.tgz` — Snapshot/Backup (Configs, Reports).

---

12) Troubleshooting (Kurzmatrix)

- `Internal Server Error` **beim Start** → Torch < 2.6: Env mit Torch ≥ 2.6 reaktivieren.
- `address already in use (8091)` → laufenden Guard killen (`lsof -tiTCP:8091 … | xargs kill -9`).
- `NotOpenSSLWarning` **(LibreSSL)** → irrelevant für lokalen Betrieb; Conda-Env nutzen.
- **cURL „unknown file attribute: h“** → Zeilenumbrüche/Backslashes in zsh korrigieren (einzeilig).
- **Here-Doc hängt** → `PY` / `MD` **allein** in neuer Zeile eingeben.

---

13) Übergabe-Checkliste (Heute)

- [x] MT & Guard starten/stoppen
- [x] Health/Meta/Reload-Endpoints
- [x] TM exact/fuzzy inkl. Live-Reload
- [x] Glossar `never_translate` aktiv
- [x] Polish (auto) aktiv
- [x] Quality-Checks im Response
- [x] Crawling/Parsing & Satz-Split
- [x] Batch-Übersetzung + Reports
- [x] Snapshot & README & Git

---

14) To-Dos (kurzfristig)

- Re-Check der 7 Restfails aus `python_crawl_en2de_out.jsonl` (Schritt vorbereitet).
- Parallel-Runner hinzufügen (Param `-j 8`).
- Memo-Cache (Guard) aktivieren.

---

15) Anhang — Quick Commands

```
# Health & Meta
curl -s http://127.0.0.1:8091/health
curl -s http://127.0.0.1:8091/meta

# TM & Glossar neu laden
curl -s -X POST http://127.0.0.1:8091/admin/reload

# Beispiel Guard-Call
```

```
./t.sh en de 'Save changes'            # → TM exact: „Änderungen speichern“
./t.sh de en '<strong>…</strong>'        # HTML/PH unverändert

# Batch mit Checks
./batch_tq.py input.jsonl output.jsonl

# Snapshot
tar -czf trancelate_selfhost_$(date +%Y%m%d_%H%M).tgz
  mt_guard.py polish.py t.sh tp.sh tq.sh tm_upsert.py tm.csv
  orgcard.json glossary.json start_local.sh stop_local.sh
  env_tl311.yml README_selfhost.md
```

Fazit: Du hast jetzt eine **lokale, auditierbare** MT-Plattform mit **starkem Stil-Versprechen** (Tonalität, Placeholder-Sicherheit, HTML-Treue) und Werkzeugen für **skalierbare Website-Übersetzungen**. Die letzten Optimierungen (Parallel-Runner/Memo-Cache) sind klar umrissen und unabhängig umsetzbar.

Gute Nacht! 🌙

---

16) Transcreation — Sinn statt Übersetzung (Konzept & Umsetzung)

Ziel: Nicht Wortlaut übertragen, sondern **Sinn + Wirkung** in der Zielsprache neu schreiben (Copywriting), bei strenger Einhaltung von Marken-/Platzhalter-/Zahlen-/HTML-Invarianten.

A) Zwei Betriebsmodi

- **Translate-Modus (bestehend):** formtreue Übertragung mit Guard-Checks.
- **Transcreate-Modus (neu):** 1) **Meaning Capture** (sprachagnostisch) → kompaktes Briefing. 2) **Target Authoring** (Copywriter-Persona je Zielsprache) → Text in eigenen Worten. 3) **Gates** (Guard): Must-Keep/PH/Numbers/HTML/Legal unverändert.

B) Datenfluss (Transcreate)

1. **/transcreate** Request (source, target, raw_text, orgcard, options).
2. **Meaning Capture**: extrahiert Zweck, Publikum, Key Points, Must-Keeps.
3. **Copywriter**: erzeugt Zieltext nur aus Briefing + OrgCard (sieht Original nicht direkt).
4. **Guard-Gates**: Platzhalter/Marken/Zahlen/HTML prüfen; Coverage & Ton verifizieren.
5. **Response**: `{ text, provenance: {mode:"transcreate", engine}, checks }`.

C) Briefing-Schema (sprachagnostisch)

```
{
  "purpose": "CTA Button",
  "audience": "SMB, non-technical",
  "key_points": ["quick start", "no risk", "free trial"],
  "must_keep": ["{{COUNT}}", "TranceLate Pro"],
  "tone": {"voice":"confident","warmth":"medium","formality":"low"},
```

```
  "constraints": {"placeholders": true, "numbers": "preserve", "legal": []}
}
```

D) API-Entwurf

Endpoint: `POST /transcreate`

```
{
  "source": "de",
  "target": "sv-SE",
  "text": "<strong>Jetzt starten</strong> – {{COUNT}} Seiten
synchronisiert.",
  "orgcard": {...},
  "options": {"polish":"off|light", "max_len": 140}
}
```

Antwort (Beispiel):

```
{
  "text": "<strong>Starta nu</strong> – {{COUNT}} sidor synkroniserade.",
  "provenance": {"mode":"transcreate","engine":"copywriter-sv"},
  "checks":
{"ph_ok":true,"num_ok":true,"html_ok":true,"paren_ok":true,"len_ratio":0.92}
}
```

E) Copywriter-Prompt (Vorlage)

System (Zielsprache):

> Du bist leitende*r Werbetexter*in in {target_locale}. Du schreibst idiomatisch, präzise, ohne unnötige Längung. Du veränderst **keine** Platzhalter/Marken/Zahlen/HTML. Keine erfundenen Fakten.

Aufgabe:

> Schreibe nur auf Basis dieses **Briefings** (keine Wort-für-Wort-Übernahme der Quelle): `{brief_json}` **Constraints:** `must_keep` & Platzhalter exakt, Zahlen/Datum unverändert, HTML-Tags intakt.
> **Self-Check vor Abgabe:** Sind alle `key_points` abgedeckt? Alle Must-Keeps enthalten? Wenn nein → korrigieren.

F) Qualität in Transcreation

Hard Gates (Guard): Platzhalter 1:1, Zahlen/Datum unverändert (Unicode-Ziffern normalisiert), Marken/never-translate unverändert, HTML-Tags identisch.
Soft Gates: Coverage (`key_points`), Ton (OrgCard), Länge (±Toleranz), Leseniveau.
Human Gate: A/B-Präferenz vs. Referenz/Provider.

G) Skalierung ohne 50 Einzel-Tweaks

- **Model-Router (Tiered):** 5–7 Backend-Familien (z. B. Lateinisch, CJK, Indic, RTL, Multilingual-Fallback) statt 50+ Sprachen.
- **Familien-Splitter & Policies:**
- Lateinisch: `.?!` + Abkürzungsmaske; Polish „light“.
- CJK: `。！？；`; kein Extra-Spacing; Polish „off“.
- Indic: `।` / `॥`; Devanagari-Digits → ASCII-Digits.
- RTL: familientypische Endzeichen; keine westliche Quote-Normalisierung.
- **Guard universell:** Unicode-Digit-Normalizer; AM/PM-Heuristik nur für `source=en`.
- **TM/Glossar:** global + kundenspez.; fuzzy mit PH-Set; nie-übersetzen markensicher.

H) Umsetzung (kurzfristige Schritte)

1. **/transcreate-Route** implementieren (Meaning-Capture + Copywriter + Gates).
2. **Unicode-Digits & Familien-Splitter** im Guard aktivieren.
3. **Router-Map** (pair→backend) einziehen; Tier-Backends registrieren.
4. **Mini-Eval** je Familie (30 Sätze): Gates 100%, Präferenzscore.
5. **Go/No-Go je Paar** → Router pflegen; Polish-Policy pro Familie.

I) Grenzen & Hinweise

- **Transcreation nicht** für Legal/Medical/Verträge/Specs → dort Translate-Modus.
- Copywriter darf **keine neuen Claims** erfinden; nur Ton & Stil lokal anpassen.
- Monitoring: Coverage-Fehlschläge & Gate-Violations als Metriken ausweisen.