Guida Operativa — Progetto NanoSocrates (Opzione A)

Baseline unificata: Text↔RDF + Completion (film domain)

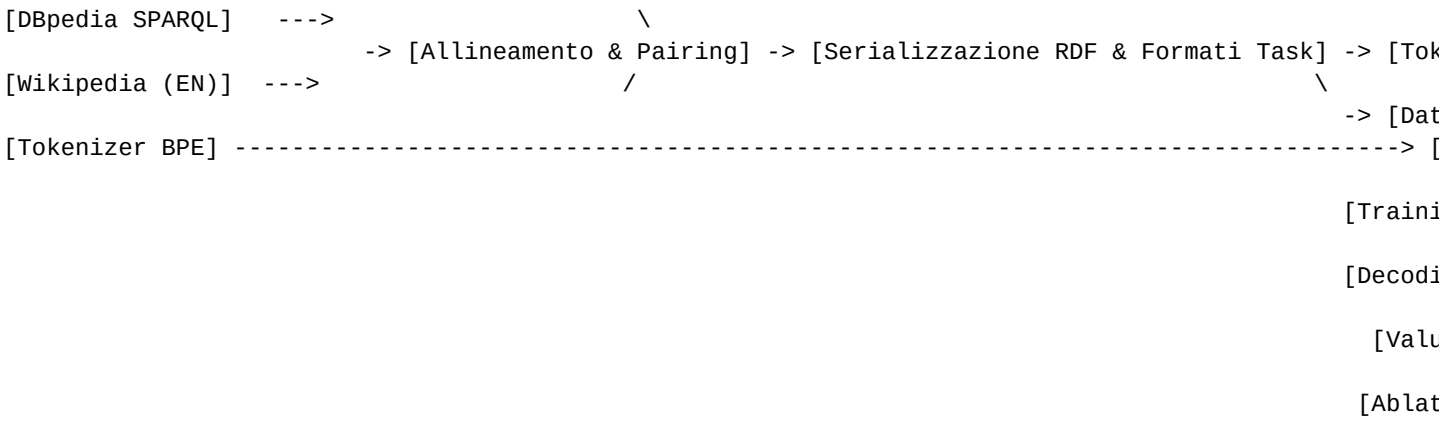
Data: 01 October 2025 — Setup target: GPU 16GB VRAM

Contesto: pipeline end-to-end conforme alla traccia (token speciali, tokenizer from scratch, encoder–decoder, multi-task training, metriche per task).

0) Obiettivo (controllo rotta)

Costruire un unico modello Transformer encoder–decoder capace di eseguire quattro task nel dominio dei film: Text2RDF, RDF2Text, RDF Completion 1 (masked) e RDF Completion 2 (continuation). Usare un tokenizer BPE addestrato da zero su (testo + RDF linearizzato) e una serie di token speciali <SOT>, <EOT>, <SUBJ>, <PRED>, <OBJ>, <RDF2Text>, <Text2RDF>, <CONTINUERDF>, <MASK>.

1) Vista d'insieme (pipeline & connessioni)



Passaggi obbligati: raccolta dati allineati (DBpedia↔Wikipedia) → serializzazione RDF con token speciali e prefisso di task → tokenizer from scratch → modello encoder–decoder → training multi-task → valutazione per task.

2) Data Layer (raccolta, allineamento, qualità)

- Entity sourcing (DBpedia) — obbligatorio: interrogazioni SPARQL per risorse di tipo dbo:Film e vicini 1-hop; estrarre triple rilevanti.
- Testo (Wikipedia EN) — obbligatorio: per ogni film, abstract introduttivo (primo paragrafo consigliato).
- Pairing & filtro qualità — obbligatorio: unire {text, triples}; selezionare predicati informativi (director, starring, writer, genre, releaseDate, runtime, country, language); scartare film con poche triple; split train/val/test per film (no leakage).
- Artefatto: pairs.jsonl con campi {film, text, triples:[(s,p,o)…]}.

3) Serializzazione & Formati per-task (I/O contratto)

Token speciali — obbligatorio: <SOT> <EOT> <SUBJ> <PRED> <OBJ> <RDF2Text> <Text2RDF> <CONTINUERDF> <MASK>.

RDF linearizzato — obbligatorio:

```
<SOT> <SUBJ> dbr:Inception <PRED> dbo:director <OBJ> dbr:Christopher_Nolan <EOT>
<SOT> <SUBJ> dbr:Inception <PRED> dbo:genre    <OBJ> dbr:Science_fiction    <EOT>
```

Prefisso di task (input routing) — obbligatorio:

```
Text2RDF:      input = text + " <Text2RDF>",        target = rdf_serialized
RDF2Text:      input = rdf_serialized + " <RDF2Text>", target = text
RDF Comp. 1:   input = rdf_serialized_con_<MASK> + " <MASK>", target = span mascherato (spanned mask
RDF Comp. 2:   input = context_rdf + " <CONTINUERDF>", target = triple successive
```

Artefatti: text2rdf.jsonl, rdf2text.jsonl, rdfcomp1.jsonl, rdfcomp2.jsonl (campi: input, target).

4) Tokenizer Layer (BPE from scratch)

- Addestrare BPE sul mix (testo + RDF linearizzato); vocab consigliato 16k–32k (baseline: 24k).

- Inserire i token speciali; validare OOV vicino a 0 su prefissi dbr:/dbo:.
- Artefatti: vocab.json, merges.txt, mapping ID token speciali.

5) Model Layer (encoder–decoder Transformer)

Architettura — obbligatoria:

Encoder–Decoder Transformer con Multi-Head Self-Attention. Config micro (16GB): 4 encoder + 4 decoder, d_model=512, n_heads=8, d_ff=2048, dropout=0.1, Pre-LN, max_len=384, weight tying, label smoothing=0.1.

Varianti incoraggiate (ablation) — opzionali: RoPE, MLA, Interleaved Attention; Spanned Masking in Comp-1.

6) Training Layer (multi-task, ottimizzazione, sanity)

- Loop multi-task con mixing per batch: ratio iniziale 3:3:2:2 (Text2RDF:RDF2Text:Comp1:Comp2).
- AdamW (lr 3e-4), warmup (circa 2k steps) + cosine decay; gradient accumulation per batch effettivo 64.
- Sanity workflow: toy set (10–20 film) + overfit su 1 batch; max seq 256–512 con troncatura.
- Early-stopping su macro-F1 (Text2RDF+Comp2) e ROUGE-L (RDF2Text).

7) Decoding & Post-processing (robustezza output)

- Decoding base: beam search o top-k.
- Vincoli leggeri (consigliati): dopo limitare il vocab a dbo: noti; dopo / preferire dbr:/letterali.
- Validatore di tripla: parser dell'output → lista triple, dedup, normalizzazione URI.

8) Evaluation Layer (per-task, obbligatorio)

- RDF2Text → BLEU, ROUGE, METEOR.
- Text2RDF → Precision / Recall / F1 su set di triple (match esatto con normalizzazione URI/prefissi).
- RDF Completion 1 → Accuracy sullo slot mascherato.
- RDF Completion 2 → Precision / Recall / F1 su triple generate.
- Artefatti: predictions_{task}.jsonl, metrics_{task}.json.

9) Governance esperimenti (riproducibilità)

- Config unico (seed, lr, ratio task, maxlen, vocab).
- Logging per task (loss/metriche) + esempi qualitativi a checkpoint.
- Split fissi con seed deterministico.

10) Roadmap step-by-step (con dipendenze)

1. 1) SPARQL + Wikipedia fetch → pairs.jsonl (pilot ■ 5k).
2. 2) Serializzazione RDF + costruzione 4 dataset (text2rdf/rdf2text/rdfcomp1/rdfcomp2).
3. 3) Tokenizer BPE from scratch (24k + special token) → vocab/.
4. 4) Scheletro encoder–decoder micro (4e+4d, d=512) + data loaders.
5. 5) Sanity: toy set + overfit 1 batch.
6. 6) Training multi-task su 5k; maxlen 384; AdamW + scheduler.

7) Decoding & post (parser triple, vincoli leggeri opz.).

8) Valutazione per-task (BLEU/ROUGE/METEOR; F1; Accuracy).

9) Ablation breve (RoPE on/off o sampling differente).

10) Analisi errori & raccomandazioni.

11) Criteri di Go/No-Go per checkpoint

- Dopo Step 2: ≥90% delle triple parse-abili dal serializzatore.
- Dopo Step 3: OOV ≈ 0 su dbr:/dbo/; lunghezza media ≤ maxlen.
- Dopo Step 5: overfit-batch raggiunto → via libera a 5k.
- Durante Step 6: loss in calo su dev; nessun collapse su un task.
- Dopo Step 8: metriche coerenti (ROUGE-L sopra baseline; F1 > random).

12) Rischi e mitigazioni

- Allineamento rumoroso: usare primo paragrafo, predicati comuni; scartare entità con poche triple.
- Out-of-memory: maxlen 256–512, truncation, gradient accumulation, vocab ≤ 32k.
- Output RDF invalidi: vincoli leggeri + validatore locale.
- Instabilità training: mixing fisso, warmup, label smoothing.

Appendice A — Template SPARQL (predicati informativi)

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT ?film ?p ?o WHERE {
  ?film a dbo:Film .
  VALUES ?p { dbo:director dbo:starring dbo:writer dbo:musicComposer dbo:releaseDate dbo:runtime dbo
  ?film ?p ?o .
}
LIMIT 100000
```

Appendice B — Contratti I/O (esempi minimi)

```
pairs.jsonl record:
{
  "film": "dbr:Inception",
  "text": "Inception is a 2010 science fiction film ...",
  "triples": [
    ["dbr:Inception","dbo:director","dbr:Christopher_Nolan"],
    ["dbr:Inception","dbo:starring","dbr:Leonardo_DiCaprio"]
  ]
}

Formati task:
Text2RDF.input:  "<TEXT> ... <Text2RDF>"
Text2RDF.target: "<SOT> <SUBJ> dbr:Inception <PRED> dbo:director <OBJ> dbr:Christopher_Nolan <EOT> .

RDF2Text.input:  "<SOT> <SUBJ> dbr:Inception <PRED> dbo:genre <OBJ> dbr:Science_fiction <EOT> ... <R
RDF2Text.target: "Inception is a science fiction film ..."

Comp1.input:     "<SOT> <SUBJ> dbr:Inception <PRED> dbo:director <OBJ> <MASK> <EOT> ... <MASK>"
Comp1.target:    "dbr:Christopher_Nolan"

Comp2.input:     "<SOT> <SUBJ> dbr:Inception <PRED> dbo:director <OBJ> dbr:Christopher_Nolan <EOT> <
```

```
Comp2.target:    "<SOT> <SUBJ> dbr:Inception <PRED> dbo:genre <OBJ> dbr:Science_fiction <EOT> ..."
```