ДОДАТОК А

Лістинги вихідного коду

Лістинг А.1 – Конфігураційний файл (config.py). Визначає гіперпараметри фільтрації, спеціальні токени та параметри навчання.

```
from dataclasses import dataclass
from pathlib import Path

ROOT_DIR = Path("/tf-knugs/llm-translator")
DATA_DIR = ROOT_DIR / "data"

TEXT_REPLACEMENTS = {
    "\n": "[NL]", "\u201c": '"', "\u201d": '"', "\u201e":
'"',
    "\u2018": "'", "\u2014": "-", "\u2026": "...",
    "`": "'", "'": '"',
    "$": "dollars", "\u20ac": "euros", "\u00bd": "1/2"
}

@dataclass(frozen=True)
class FilterConfig:
    min_char_len: int = 512
    min_num_of_words: int = 128
    header_check_len: int = 200
    footer_check_len: int = 200
    max_digit_ratio: float = 0.18
    max_newline_ratio: float = 0.02
    min_alpha_word_ratio_en: float = 0.70
    min_alpha_word_ratio_de: float = 0.60
    allowed_uppercase_ratio_en: float = 0.07
    allowed_uppercase_ratio_de: float = 0.13
    ttr_threshold_small: float = 0.4
    ttr_threshold_mid: float = 0.35
    ttr_threshold_large: float = 0.3
    max_avg_token_id: float = 7000.0
    min_avg_token_len: float = 3.2
    max_avg_token_len: float = 5.0
    max_unk_count: int = 1

class SpecialTokenIDs:
    PAD = 0; UNK = 1; EN = 2; DE = 3; EOS = 4; MASK = 5
```

```
VOCAB_SIZE = 30000
BATCH_SIZE = 100
```

Лістинг А.2 – Архітектура моделі (model.py). Містить реалізацію RoPE, SwiGLU, GQA та encoder-decoder Transformer.

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class RotaryPositionalEmbeddings(nn.Module):
    def __init__(self, dim, max_seq_len=4096,
base=10000):
        super().__init__()
        inv_freq = 1.0 / (base ** (torch.arange(0, dim,
2, dtype=torch.float32) / dim))
        freqs = torch.outer(torch.arange(max_seq_len,
dtype=torch.float32), inv_freq)
        self.register_buffer("freqs_cos",
torch.cos(freqs))
        self.register_buffer("freqs_sin",
torch.sin(freqs))

    def forward(self, x, start_pos=0):
        seq_len = x.shape[1]
        x_reshaped = x.float().reshape(*x.shape[:-1], -1,
2)
        x_r, x_i = x_reshaped[..., 0], x_reshaped[..., 1]
        cos =
self.freqs_cos[start_pos:start_pos+seq_len].unsqueeze(0).
unsqueeze(2)
        sin =
self.freqs_sin[start_pos:start_pos+seq_len].unsqueeze(0).
unsqueeze(2)
        out_r = x_r * cos - x_i * sin
        out_i = x_r * sin + x_i * cos
        return torch.stack([out_r, out_i], dim=-
1).flatten(3).type_as(x)

class SwiGLU(nn.Module):
    def __init__(self, dim, hidden_dim=None,
```

```
multiple_of=256, bias=False):
        super().__init__()
        if hidden_dim is None:
            hidden_dim = int(2 * 4 * dim / 3)
            hidden_dim = multiple_of * ((hidden_dim +
multiple_of - 1) // multiple_of)
        self.w1 = nn.Linear(dim, hidden_dim, bias=bias)
        self.w2 = nn.Linear(hidden_dim, dim, bias=bias)
        self.w3 = nn.Linear(dim, hidden_dim, bias=bias)

    def forward(self, x):
        return self.w2(F.silu(self.w1(x)) * self.w3(x))

class GroupedQueryAttention(nn.Module):
    def __init__(self, d_model, n_head, num_kv_heads,
rope_kwargs, bias=False):
        super().__init__()
        self.n_head, self.kv_head = n_head, num_kv_heads
        self.head_dim = d_model // n_head
        self.rope =
RotaryPositionalEmbeddings(**rope_kwargs) if rope_kwargs
else None
        self.wq = nn.Linear(d_model, n_head *
self.head_dim, bias=bias)
        self.wk = nn.Linear(d_model, num_kv_heads *
self.head_dim, bias=bias)
        self.wv = nn.Linear(d_model, num_kv_heads *
self.head_dim, bias=bias)
        self.wo = nn.Linear(n_head * self.head_dim,
d_model, bias=bias)

    def forward(self, x_q, x_kv=None, mask=None,
kv_cache=None, is_causal=False):
        if x_kv is None: x_kv = x_q
        B, S_q, _ = x_q.shape
        xq = self.wq(x_q).view(B, S_q, self.n_head,
self.head_dim)
        xk = self.wk(x_kv).view(B, -1, self.kv_head,
self.head_dim)
        xv = self.wv(x_kv).view(B, -1, self.kv_head,
self.head_dim)
        if self.rope:
            sp = kv_cache[0].shape[1] if kv_cache and
kv_cache[0] is not None else 0
```

```
            xq, xk = self.rope(xq, sp), self.rope(xk, sp)
        if kv_cache and kv_cache[0] is not None:
            xk = torch.cat([kv_cache[0], xk], dim=1)
            xv = torch.cat([kv_cache[1], xv], dim=1)
        updated_kv = (xk.clone().detach(),
xv.clone().detach())
        xq, xk, xv = [t.transpose(1, 2).contiguous() for
t in (xq, xk, xv)]
        if is_causal:
            L = xk.shape[2]
            cm = torch.tril(torch.ones(1,1,S_q,L,
device=xq.device, dtype=torch.bool))
            mask = mask & cm if mask is not None else cm
        out = F.scaled_dot_product_attention(xq, xk, xv,
attn_mask=mask, enable_gqa=True)
        return self.wo(out.transpose(1,
2).contiguous().view(B, S_q, -1)), updated_kv

class ModernEncoderLayer(nn.Module):
    def __init__(self, d_model, n_head, num_kv_heads,
rope_kwargs, dropout=0.1):
        super().__init__()
        self.self_attn = GroupedQueryAttention(d_model,
n_head, num_kv_heads, rope_kwargs)
        self.feed_forward = SwiGLU(dim=d_model)
        self.attention_norm = nn.LayerNorm(d_model)
        self.ffn_norm = nn.LayerNorm(d_model)
        self.attn_dropout, self.ffn_dropout =
nn.Dropout(dropout), nn.Dropout(dropout)

    def forward(self, x, mask):
        h, _ = self.self_attn(x_q=self.attention_norm(x),
mask=mask)
        x = x + self.attn_dropout(h)
        return x +
self.ffn_dropout(self.feed_forward(self.ffn_norm(x)))

class ModernDecoderLayer(nn.Module):
    def __init__(self, d_model, n_head, num_kv_heads,
rope_kwargs, dropout=0.1):
        super().__init__()
        self.self_attn = GroupedQueryAttention(d_model,
n_head, num_kv_heads, rope_kwargs)
        self.cross_attn = GroupedQueryAttention(d_model,
```

```
n_head, num_kv_heads, None)
        self.feed_forward = SwiGLU(dim=d_model)
        self.sa_norm, self.ca_norm, self.ffn_norm =
[nn.LayerNorm(d_model) for _ in range(3)]
        self.sa_drop, self.ca_drop, self.ffn_drop =
[nn.Dropout(dropout) for _ in range(3)]

    def forward(self, x, memory, src_mask, tgt_mask,
kv_cache=None):
        h, kv = self.self_attn(self.sa_norm(x),
mask=tgt_mask, kv_cache=kv_cache, is_causal=True)
        x = x + self.sa_drop(h)
        h, _ = self.cross_attn(x_q=self.ca_norm(x),
x_kv=memory, mask=src_mask)
        x = x + self.ca_drop(h)
        return x +
self.ffn_drop(self.feed_forward(self.ffn_norm(x))), kv

class ModernEncoderDecoder(nn.Module):
    def __init__(self, vocab_size=30000, d_model=512,
n_head=8, num_kv_heads=2,
                 num_encoder_layers=9,
num_decoder_layers=11, dropout=0.1, max_seq_len=1024):
        super().__init__()
        self.embedding = nn.Embedding(vocab_size,
d_model)
        rope = {"dim": d_model // n_head, "max_seq_len":
max_seq_len}
        self.encoder_layers = nn.ModuleList(
            [ModernEncoderLayer(d_model, n_head,
num_kv_heads, rope, dropout)
             for _ in range(num_encoder_layers)])
        self.decoder_layers = nn.ModuleList(
            [ModernDecoderLayer(d_model, n_head,
num_kv_heads, rope, dropout)
             for _ in range(num_decoder_layers)])
        self.encoder_norm = nn.LayerNorm(d_model)
        self.decoder_norm = nn.LayerNorm(d_model)
        self.fc_out = nn.Linear(d_model, vocab_size,
bias=False)
```

Лістинг А.3 – Основна функція конвеєра фільтрації (preparation.py, фрагмент). Реалізує рівні 1-2 каскадної фільтрації.

```
import re
from config import FilterConfig

def filter_texts(example):
    text = example.get("text", "")
    lang = example.get("lang", "")
    if not text: return False

    # Рівень 1: Регулярні вирази
    has_poison = bool(POISON_PATTERNS.search(text))
    has_bad_header =
bool(BOILERPLATE_PATTERNS.search(text[:FilterConfig.heade
r_check_len]))
    has_bad_footer =
bool(BOILERPLATE_PATTERNS.search(text[-
FilterConfig.footer_check_len:]))

    if has_bad_header:
        text = text[trim_text(text):]
        has_bad_header =
bool(BOILERPLATE_PATTERNS.search(text[:FilterConfig.heade
r_check_len]))
    if has_bad_footer:
        text = text[:trim_text(text, footer=True)]
        has_bad_footer =
bool(BOILERPLATE_PATTERNS.search(text[-
FilterConfig.footer_check_len:]))

    text_len = len(text) or 1
    if text_len < FilterConfig.min_char_len or has_poison
or has_bad_header or has_bad_footer:
        return False

    # Рівень 2: Статистичні евристики
    words = text.lower().split()
    num_words = len(words) or 1
    if num_words < FilterConfig.min_num_of_words: return
False

    cleaned = [w.strip(".,!?;:'\"") for w in words]
    ttr = len(set(cleaned)) / len(cleaned)
```

```
    threshold = (FilterConfig.ttr_threshold_small if
num_words < 500
                 else FilterConfig.ttr_threshold_mid if
num_words < 2000
                 else FilterConfig.ttr_threshold_large)
    if ttr < threshold: return False

    num_digits = sum(c.isdigit() for w in words for c in
w)
    if num_digits / text_len >
FilterConfig.max_digit_ratio: return False

    mean_word_len = sum(len(w) for w in words) /
num_words
    if not (3 < mean_word_len < (15 if lang == "de" else
12)): return False

    return True
```

Лістинг А.4 -- Токенайзерна фільтрація рівня 3 (preparation.py, фрагмент). Оцінює якість тексту через статистики навченого токенайзера.

```
from config import TEXT_REPLACEMENTS, SpecialTokenIDs

def tokenize_and_evaluate(data, tokenizer, max_len=128,
min_len=128,
                          max_avg_token_id=7000,
min_avg_token_len=3.2,
                          max_avg_token_len=5.0,
max_unk_count=1,
                          max_examples_per_text=16,
batched=True):
    raw_texts = data.get("text", [] if batched else "")
    langs = data.get("lang", [] if batched else "")
    if not batched: raw_texts = [raw_texts]

    texts = []
    for text in raw_texts:
        for old, new in TEXT_REPLACEMENTS.items():
            text = text.replace(old, new)
        texts.append(text)
```

```
    encodings = tokenizer.encode_batch(texts,
add_special_tokens=False)
    batch_filtered, batch_token_ids, batch_langs = [],
[], []
    max_tokens_per_chunk = max_len - 2

    for doc_idx, enc in enumerate(encodings):
        full_ids = enc.ids
        full_strings = enc.tokens

        for ix, start in enumerate(range(0,
len(full_ids), max_tokens_per_chunk)):
            if ix >= max_examples_per_text: break
            chunk_ids = full_ids[start:start +
max_tokens_per_chunk]
            chunk_strs = full_strings[start:start +
max_tokens_per_chunk]
            tok_len = len(chunk_ids)

            if tok_len < min_len - 2:
                batch_filtered.append(True)
                batch_token_ids.append([])
                batch_langs.append(langs[doc_idx])
                continue

            unk_count = sum(1 for t in chunk_ids if t ==
SpecialTokenIDs.UNK)
            avg_tok_id = sum(chunk_ids) / tok_len
            avg_tok_len = sum(len(t) for t in chunk_strs)
/ tok_len

            is_filtered = (
                unk_count >= max_unk_count or
                avg_tok_id > max_avg_token_id or
                not (min_avg_token_len <= avg_tok_len <=
max_avg_token_len)
            )
            batch_filtered.append(is_filtered)
            batch_token_ids.append(chunk_ids)
            batch_langs.append(langs[doc_idx])

    return {"filtered_by_tokenizer": batch_filtered,
            "token_ids": batch_token_ids, "lang":
batch_langs}
```

Лістинг А.5 - Data Collator з динамічним span corruption (preparation.py, фрагмент). Реалізує T5-style denoising objective для передтренування.

```
import math

def get_mask_prob(step, start_prob, end_prob, max_steps):
    if step >= max_steps: return end_prob
    progress = step / max_steps
    cosine_decay = 0.5 * (math.cos(math.pi * progress) +
1.0)
    return end_prob + (start_prob - end_prob) *
cosine_decay

def generate_span_lengths(num_spans, p, min_len,
max_len):
    lengths =
torch.distributions.Geometric(torch.tensor([p])).sample((
num_spans,)).squeeze(-1) + 1
    return torch.clamp(lengths, min_len, max_len).long()

class DataCollatorDynamic:
    def __init__(self, start_prob, end_prob, steps,
en_token_id=2,
                 de_token_id=3, eos_token_id=4,
pad_token_id=0, mask_token_id=5):
        self.start_prob, self.end_prob, self.steps =
start_prob, end_prob, steps
        self.en_token, self.de_token = en_token_id,
de_token_id
        self.eos_token, self.pad_token, self.mask_token =
eos_token_id, pad_token_id, mask_token_id
        self.mean_span, self.p, self.step = 6.0, 1.0 /
6.0, 0

    def __call__(self, batch, step=None):
        current_step = step if step is not None else
self.step
        actual_prob = get_mask_prob(current_step,
self.start_prob, self.end_prob, self.steps)
        batch_size = len(batch)
        batch_len = max(len(item["token_ids"]) for item
in batch) + 2
```

```
        padded_inputs = torch.full((batch_size,
batch_len), self.pad_token, dtype=torch.long)
        padded_dec_inputs = torch.full((batch_size,
batch_len), self.pad_token, dtype=torch.long)
        padded_labels = torch.full((batch_size,
batch_len), self.pad_token, dtype=torch.long)

        for b_idx, item in enumerate(batch):
            lang_tok = self.en_token if item["lang"] ==
"en" else self.de_token
            raw_ids = torch.tensor(item["token_ids"],
dtype=torch.long)
            labels = torch.cat([raw_ids,
torch.tensor([self.eos_token], dtype=torch.long)])
            dec_inputs =
torch.cat([torch.tensor([lang_tok], dtype=torch.long),
raw_ids])
            seq_len = len(raw_ids)

            if seq_len > 0:
                mask_flags = torch.rand(seq_len) <
actual_prob
                num_spans = mask_flags.sum().item()
                if num_spans > 0:
                    span_lengths =
generate_span_lengths(int(num_spans), self.p, 2, 20)
                    final_mask = torch.zeros(seq_len,
dtype=torch.bool)
                    span_idx, force = 0, 0
                    for i in range(seq_len):
                        if force > 0:
                            final_mask[i] = True; force -
= 1
                        elif mask_flags[i]:
                            final_mask[i] = True
                            force =
span_lengths[span_idx].item() - 1; span_idx += 1
                    raw_ids[final_mask] = self.mask_token

                is_mask = raw_ids == self.mask_token
                if is_mask.any():
                    shifted = torch.roll(is_mask,
shifts=1); shifted[0] = False
```

```
                    collapsed_ids = raw_ids[~is_mask |
(is_mask & ~shifted)]
                else:
                    collapsed_ids = raw_ids
            else:
                collapsed_ids = raw_ids

            inputs = torch.cat([torch.tensor([lang_tok],
dtype=torch.long),
                                collapsed_ids,
torch.tensor([self.eos_token], dtype=torch.long)])
            padded_inputs[b_idx, :len(inputs)] = inputs
            padded_dec_inputs[b_idx, :len(dec_inputs)] =
dec_inputs
            padded_labels[b_idx, :len(labels)] = labels

        return {"input_ids": padded_inputs,
                "attention_mask": (padded_inputs !=
self.pad_token).long(),
                "decoder_input_ids": padded_dec_inputs,
                "decoder_attention_mask":
(padded_dec_inputs != self.pad_token).long(),
                "labels": padded_labels}
```

Лістинг А.6 – Приклад запуску генерації перекладу. Демонструє завантаження моделі та генерацію з KV Cache.

```
import torch
from tokenizers import Tokenizer
from model import ModernEncoderDecoder
from config import SpecialTokenIDs, VOCAB_SIZE

model = ModernEncoderDecoder(vocab_size=VOCAB_SIZE,
d_model=512, n_head=8,
    num_kv_heads=2, num_encoder_layers=9,
num_decoder_layers=11, dropout=0.0)
model.load_state_dict(torch.load("checkpoint.pt",
map_location="cpu"))
model.eval()

tokenizer = Tokenizer.from_file("tokenizer.json")
```

```
text = "Ich fahre nach Berlin."
encoded = tokenizer.encode(text)
input_ids = [SpecialTokenIDs.DE] + encoded.ids +
[SpecialTokenIDs.EOS]
input_tensor = torch.tensor([input_ids])

decoder_start = torch.tensor([[SpecialTokenIDs.EN]])
output_ids = model.generate(
    input_ids=input_tensor,
decoder_input_ids=decoder_start,
    max_new_tokens=200, temperature=0.8, top_k=40,
top_p=0.8
)

translation = tokenizer.decode(output_ids[0].tolist(),
skip_special_tokens=True)
print(translation)
```

ДОДАТОК Б

Приклади перекладів моделі

Таблиця Б.1 – Приклади перекладів дотренованої моделі (DE→EN) на текстах різної складності

Рівень	**Вхід (DE)**	**Переклад моделі (EN)**
A1	[DE] Ich bin Anna. Ich fahre nach Berlin. Ich buche ein Hotel. Das Hotel ist gut. Berlin ist groß. Ich mag Berlin. [EOS]	[EN] I'm a Anna. I go to Berlin. I'll walk on Berlin. I am writing Anna. i drive an hotel. I drive a Hotel. I ride at Berlin. I go by the busts. I've been some Santa..adyns is good. Berlin's big. Germany is large. [EOS]

A2	[DE] Hallo, mein Name ist Anna. Nächste Woche fliege ich nach Berlin, weil ich die Stadt besuchen will. Ich habe schon ein Hotelzimmer gebucht. Ich möchte die Sehenswürdigkeiten sehen und vielleicht in einem Restaurant essen. Kannst du mir helfen? [EOS]	[EN] Hi, my name is Anna. Next week I'm starting to camp I go to Berlin because I want the city on Tuesday because I want to visit towns. I have been booked a hotel room for Europe? Can't get one Hotel Room already this morning. I would like to see that sights and perhaps eat in restaurant. Can you help me with it and be sure about your Restaurant if you're eating. Do you need to assist myself? Don't You'll feel better than it! Are you helping me....If you are making any of them all the way! [EOS]

B1	[DE] Ich bereite gerade meine Reise nach Berlin vor und freue mich schon sehr darauf. Obwohl das Wetter nicht immer perfekt ist, hoffe ich, dass wir viele Parks erkunden können. Mir wurde gesagt, dass man unbedingt das Brandenburger Tor besuchen sollte. Falls du Zeit hast, könnten wir uns vielleicht treffen, um gemeinsam die Stadt zu entdecken. [EOS]	[EN] I just ready to go through my trip to Berlin right now and look forward to it. Although the weather isn't always perfect, I hope we can explore a lot of parking. It was said that if you have not ever been perfectly enjoying parks as well. For example it became said that when you are attending Brandenburger Tor, we should definitely visit here. If you're time for some time, they might make us meet together to discover the city together. Then in your event, please might tell us. [EOS]

B2	[DE] Bei der Planung meiner Berlin-Reise lege ich besonderen Wert darauf, nicht nur die touristischen Hauptattraktionen abzuklappern, sondern auch einen authentischen Eindruck vom Leben in den verschiedenen Stadtteilen zu bekommen. Es wäre schade, wenn man die kulturelle Vielfalt, die Berlin auszeichnet, verpassen würde. Deshalb habe ich vor, mich abseits der ausgetretenen Pfade zu bewegen und die Stadt auf eigene Faust zu erkunden. [EOS]	[EN] When planning my Berlin travel, I arrange for special value to not only deliberate the tourist attractions, but also have a authentic impression of life in its different areas. It would be shame when you get out about cultural diversity that was made up with London. That's why I'm thinking it would miss a lot more diverse than which Germany would like to come down. Therefore, by doing so before moving off through and exploring the city on your own side. [EOS]

C1	[DE] Die Faszination Berlins ergibt sich aus dem ständigen Spannungsfeld zwischen seiner turbulenten Vergangenheit und seiner dynamischen Gegenwart. Um die Stadt wirklich zu begreifen, genügt es nicht, die Überreste der Mauer zu betrachten; man muss sich mit den vielschichtigen Narrativen auseinandersetzen, die das heutige Stadtbild prägen. Der Facettenreichtum der Metropole offenbart sich erst bei der eingehenden Beschäftigung mit ihren subkulturellen Strömungen und der permanenten urbanen Transformation. [EOS]	[EN] The fascint Berlins come from the constant tension field between his turbulent past and its dynamic present. To make sure that it is really true to unexpected, they don't happen to look at irrigation of walls; you need to deal with the multinational Narratives, which reflected in this city's history. The facedness of neighborhood awareness seems only visible by their subcultural existingity using her subculture complexes & permanent urban transformations. [EOS]

Розмовна	[DE] Hey, was geht ab? Ich hab mir überlegt, wir könnten heute Abend ins Kino gehen. Der neue Film soll mega krass sein, oder? Ich hab aber keine Ahnung, worum es geht. Soll halt gut sein. Bei dir läuft's ja eh. Also, sag Bescheid! [EOS]	[EN] what about it? How does that come from now? I thought we could go to cinematir. The new movie is supposed to be there tonight. the new film shouldn't get super cool today or really? I've no idea where would it do this evening? So, but don't have any clues when questioned! If you run good at all now. So, say yes! I'm going to see it! [EOS]

Параметри генерації: `temperature=0.8, top_k=40, top_p=0.8`. Модель передтренована на фільтрованому корпусі (384M токенів).

ДОДАТОК В

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	76 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Аргунов М.К.			Дослідження повного циклу розробки генеративних мовних моделей: від підготовки даних до оптимізації інференсу	Шифр групи	Код напр./спец.
Перевірив		Стьопін О.С.				ІТШІ-22-3	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Чала Л.Е.					