Projekty – propozycje tematów

Zbiór 30 propozycji projektów zgodnych z tematyką materiałów laboratoryjnych: architektura Transformer, generowanie tekstu (GPT-2), BERT i MLM/NSP, fine-tuning, Question Answering oraz metryki ewaluacyjne.

————————————————————————————————

A. Architektura Transformera (podstawa teoretyczna)

1. „Od RNN do Transformera" — analiza porównawcza sieci sekwencyjnych

Cel: Zbudowanie kompleksowej analizy ewolucji architektur przetwarzania sekwencji — od prostych sieci rekurencyjnych po Transformer — na wspólnym zadaniu referencyjnym.

Zakres prac:

- Implementacja czterech modeli: klasyczny RNN, LSTM, GRU oraz mały Transformer (np. typ nanoGPT lub własny encoder-decoder) na tym samym zadaniu.
- Zaprojektowanie wspólnego zbioru danych umożliwiającego uczciwe porównanie.
- Eksperyment pokazujący problem znikającego gradientu: trening RNN na sekwencjach o rosnącej długości (od 10 do 200 kroków).
- Porównanie czasów treningu i możliwości zrównoleglenia.

Technologie: PyTorch / TensorFlow, matplotlib.

Efekt końcowy: Raport z tabelą porównawczą (dokładność, czas, pamięć, wrażliwość na długość), wykresy krzywych uczenia i wnioski o przyczynach przejścia na uwagę.

Kryteria oceny: Poprawność implementacji, jakość eksperymentów, głębia wnioskowania, estetyka wizualizacji.

————————————————————————————————

2. Wizualizacja mechanizmu uwagi (Attention Heatmaps)

Cel: Graficzne pokazanie, jak poszczególne głowy Multi-Head Attention wychwytują różne typy relacji językowych (syntaktyczne, semantyczne, referencyjne, lokalne).

Zakres prac:

- Wykorzystanie gotowego BERT (lub GPT) z opcją output_attentions=True.
- Wybór zdań z wyraźnymi relacjami: koreferencja, zależności składniowe, sąsiedztwo słów.
- Generowanie macierzy wag uwagi dla wszystkich warstw i głów.
- Stworzenie heatmap (matplotlib/seaborn) pozwalających porównać zachowanie różnych głów.
- Analiza specjalizacji głów i badania nad głowicami globalnymi vs lokalnymi.

Technologie: HuggingFace Transformers, PyTorch, matplotlib/plotly.

Efekt: Galeria heatmap z komentarzem opisującym rolę każdej wyróżnionej głowy.

———————————————————————————————

3. Eksperyment z Positional Encoding — kodowanie pozycji

Cel: Porównanie sinusoidalnego kodowania pozycji (Vaswani 2017) z uczeniowym oraz nowoczesnych metod (RoPE, ALiBi) pod kątem jakości i generalizacji.

Zakres prac:

- Implementacja obu klasycznych wariantów we własnym małym Transformerze.
- Test interpolacji dla pozycji spoza zakresu treningowego.
- Omówienie zasad działania RoPE i ALiBi.
- Eksperymenty z sekwencjami o różnej długości i pomiar dokładności/perplexity.
- Wizualizacja fal sinusoidalnych dla różnych wymiarów.

Technologie: PyTorch, numpy/matplotlib.

Efekt: Tabela porównawcza + wykresy generalizacji + raport wyjaśniający właściwości formuły trygonometrycznej.

———————————————————————————————

4. Złożoność obliczeniowa attention — Flash Attention i Sparse Attention

Cel: Zbadanie kosztu $O(n^2 \cdot d)$ mechanizmu uwagi i sposobów jego redukcji na długich sekwencjach.

Zakres prac:

- Pomiar czasu i pamięci standardowego attention dla n = 512–8192 tokenów.

- Implementacja lub użycie Flash Attention/PyTorch SDPA i porównanie wydajności.
- Przedstawienie idei sparse attention i redukcji złożoności do O(n·log n).
- Wykres skalowania pamięci/czasu dla obu podejść.

Technologie: PyTorch, flashattn, profilowanie GPU.

Efekt: Raport ilościowy ze skalowalności attention + demonstracja praktycznego znaczenia optymalizacji.

———————————————————————————————

5. Tokenizacja i jej wpływ na jakość modelu

Cel: Systematyczne porównanie algorytmów tokenizacji (BPE, WordPiece, SentencePiece) i ich wpływu na działanie modeli, ze szczególnym uwzględnieniem języka polskiego.

Zakres prac:

- Analiza podziału tego samego polskiego tekstu przez różne tokenizery.
- Replikacja przykładów z materiałów: „Rzeszów", „unhappiness", „transformers".
- Badanie liczby tokenów dla rzadkich słów polskich (fleksja, diakrytyki).
- Wpływ tokenizacji na realną maksymalną długość sekwencji.
- Eksperyment: czy lepsza tokenizacja poprawia wyniki?

Technologie: HuggingFace AutoTokenizer, PyTorch.

Efekt: Tabela porównawcza tokenizacji + wnioski o efektywnym wykorzystaniu limitu kontekstu.

———————————————————————————————

B. Generowanie tekstu (GPT-2)

6. Eksperyment z parametrami generowania tekstu

Cel: Dogłębne zbadanie wpływu temperature, top-k, top-p i beam search na kreatywność, spójność i różnorodność.

Zakres prac:

- Generowanie wielu wariantów tego samego promptu dla temperatury 0.1–1.5 oraz pozostałych strategii.
- Obliczenie metryk różnorodności: unikalne tokeny, TTR, entropia dystrybucji.
- Subiektywna ocena spójności względem kreatywności.
- Wykres zależności metryk od wartości parametrów.
- Wnioski o degeneracji (powtarzalności) i „bezpieczeństwie" vs „szaleństwie".

Technologie: HuggingFace pipeline/GPT-2, matplotlib, pandas.

Efekt: Interaktywny notebook z porównaniami i wnioskami o doborze parametrów.

———————————————————————————————

7. Polski generator opowiadań / chatbot konsolowy

Cel: Budowa działającego chatbota/generatora narracji po polsku opartego na modelu wielojęzycznym (BLOOM, OPT lub polski Bielik).

Zakres prac:

- Konfiguracja modelu wielojęzycznego.
- Dodanie system promptu i pętli dialogowej input()/output() do 'quit'.
- Obsługa tokenów specjalnych przy dekodowaniu.
- Opcjonalnie zapis historii rozmowy.
- Ocena jakości polskich odpowiedzi.

Technologie: HuggingFace Transformers, opcjonalnie Gradio.

Efekt: Działający czat z modelem + wnioski o mocnych/słabych stronach modelu wielojęzycznego.

———————————————————————————————

8. Statystyczna analiza generowanych tekstów

Cel: Narzędzie do automatycznej statystycznej analizy wygenerowanych tekstów.

Zakres prac:

- Wygenerowanie N odpowiedzi na ten sam prompt różnymi strategiami.
- Obliczenie: liczba słów/zdań, unikalne słowa, type-token ratio.
- Średnia i odchylenie standardowe metryk w grupach strategii.
- Porównanie zróżnicowania deterministycznych vs stochastycznych strategii.
- Prezentacja jako tabele i wykresy.

Technologie: Python, NLTK/spaCy, pandas, matplotlib.

Efekt: Kompletna analiza statystyczna charakteru każdej strategii dekodowania.

———————————————————————————————

9. Kreator treści marketingowych / poetyckich oparty na LLM

Cel: Aplikacja generująca hasła reklamowe, limeryki, krótkie opowiadania i analiza wpływu parametrów na styl.

Zakres prac:

- Określenie trybów: hasło promocyjne, slogan, limeryk, mini-opowiadanie, tweet.
- Przygotowanie promptów dla każdego trybu.
- Badanie wpływu temperature/top-p na styl.
- Zebranie przykładów z subiektywną oceną.
- Prezentacja galerii wyników z parametrami.

Technologie: HuggingFace pipeline, opcjonalnie Gradio.

Efekt: Biblioteka „przepisów" (prompt + parametry) dla różnych gatunków tekstu.

———————————————————————————————

C. BERT i reprezentacje językowe (MLM/NSP)

10. Badanie kontekstowych embeddingów — polisemia

Cel: Demonstracja różnicy między statycznymi a kontekstowymi reprezentacjami na słowach polisemicznych.

Zakres prac:

- Wybór słów: „bank", „zamek", „mysz".
- Uzyskanie embeddingów kontekstowych BERT i statycznych word2vec/FastText.
- Obliczenie podobieństwa cosinusowego między wystąpieniami.
- Klasteryzacja (PCA/t-SNE) pokazująca rozdzielenie znaczeń przez BERT.
- Wnioski o sile reprezentacji kontekstualnych.

Technologie: HuggingFace, gensim, scikit-learn, matplotlib.

Efekt: Wykres projekcji 2D z klastrami + raport o rozwiązaniu polisemii.

———————————————————————————————

11. Graficzna aplikacja do zabawy z Masked Language Modeling

Cel: Interaktywne narzędzie maskujące słowa i przewidujące je przez BERT z prezentacją prawdopodobieństw.

Zakres prac:

- Back-end: BERT fill-mask pipeline.
- API zwracające top-5 propozycji wraz z wagami.
- Front-end: pole tekstowe, zaznaczanie słowa do [MASK], paski prawdopodobieństwa.
- Testy na zdaniach polskojęzycznych (multilingual BERT).
- Domyślne przykłady ilustrujące siłę kontekstu.

Technologie: HuggingFace Pipeline, Flask/FastAPI + HTML/JS lub Gradio.

Efekt: Działająca aplikacja demonstrująca działanie MLM.

————————————————————————————————————

12. Analiza roli tokenów [CLS] i [SEP]

Cel: Badanie wpływu struktury wejścia ([CLS]/[SEP], segmenty A/B) na wyniki NSP i klasyfikacji.

Zakres prac:

- Dostęp do stanów ukrytych tokenów specjalnych.
- Testy: przesunięcie/usunięcie [CLS], zmiana kolejności segmentów, usunięcie [SEP].
- Porównanie embeddindów [CLS] dla par IsNext vs NotNext.
- Wnioski o informacji używanej przez model.

Technologie: HuggingFace, torch, sklearn, matplotlib.

Efekt: Seria eksperymentów z interpretacją roli tokenów specjalnych.

————————————————————————————————————

13. Porównanie MLM vs NSP w pre-treningu (eksperyment RoBERTa)

Cel: Sprawdzenie, czy usunięcie NSP poprawia wyniki modelu (jak w RoBERTa).

Zakres prac:

- Budowa dwóch wariantów: MLM+NSP oraz samo MLM.
- Fine-tuning na wspólnym zadaniu docelowym (sentyment SST-2 lub QA SQuAD).

- Porównanie wyników.
- Omówienie hipotez: NSP zbyt łatwe, mylące sygnały, dłuższe sekwencje.
- Własne wnioski, czy NSP jest niepotrzebne.

Technologie: HuggingFace Trainer, datasets.

Efekt: Raport porównawczy i odpowiedź na pytanie kontrolne nr 4.

———————————————————————————————

D. Fine-tuning i klasyfikacja tekstu

14. Fine-tuning DistilBERT do klasyfikacji sentymentu

Cel: Pełny pipeline fine-tuningu z ewaluacją accuracy/F1 i porównaniem z TF-IDF+regresja logistyczna.

Zakres prac:

- Pobranie zbioru (SST-2 lub polskie recenzje).
- Podział train/test (np. 160/40).
- Tokenizacja i konfiguracja AutoModelForSequenceClassification.
- Trening przez 3 epoki (Trainer), zapis checkpointu.
- Ewaluacja accuracy i F1.
- Baseline TF-IDF + regresja logistyczna.
- Porównanie i dyskusja o transfer learningu przy małych danych.

Technologie: HuggingFace, datasets, scikit-learn, numpy.

Efekt: Działający klasyfikator + tabela porównawcza + wnioski.

———————————————————————————————

15. Detektor toksyczności komentarzy online

Cel: Wieloetykietowy system klasyfikacji toksyczności z optymalizacją progu decyzyjnego.

Zakres prac:

- Użycie modelu unitary/toxic-bert.
- Przygotowanie 10–20 komentarzy.
- Wyodrębnienie prawdopodobieństw kategorii: toxic, severe_toxic, obscene, insult, threat, identity_hate.
- Tabela wyników.
- Badanie progu odcięcia (precision/recall curves) i znalezienie równowagi.

- Agregacja statystyk globalnych.

Technologie: HuggingFace pipelines, scikit-learn, pandas.

Efekt: Wieloetykietowy klasyfikator + raport progu decyzyjnego.

————————————————————————————————

16. Analiza learning curve — wpływ rozmiaru danych treningowych

Cel: Zbadanie zależności F1 od liczby przykładów treningowych podczas fine-tuningu.

Zakres prac:

- Stały zbiór testowy.
- Fine-tuning na 50, 100, 200, 400 przykładach.
- Ewaluacja F1 dla każdego rozmiaru.
- Powtórzenia z różnymi seedami (średnia ± odchylenie).
- Wykres learning curve z marginesami błędów.
- Interpretacja wypłaszczenia krzywej i minimalnej ilości danych.

Technologie: HuggingFace Trainer, scikit-learn, matplotlib.

Efekt: Learning curve + wnioski o wymaganej ilości danych.

————————————————————————————————

17. System klasyfikacji wieloklasowej newsów

Cel: Klasyfikator kategorii artykułów prasowych (Świat/Sport/Biznes/Technologia) z interfejsem testowym.

Zakres prac:

- Użycie AG News z HuggingFace Datasets.
- Model baseline textattack/bert-base-uncased-ag-news.
- Opcjonalnie własny fine-tuning DistilBERT.
- Predykcja nowych artykułów.
- Macierz pomyłek i analiza mylonych klas.
- Bonus: GUI (gradio).

Technologie: HuggingFace, scikit-learn, gradio (opcjonalnie).

Efekt: Gotowy klasyfikator + analiza błędów + demo.

————————————————————————————————

18. Analiza warstw uwagi przed i po fine-tuningu

Cel: Zbadanie, jak fine-tuning zmienia specjalizację głów uwagi BERT/DistilBERT.

Zakres prac:

- output_attentions=True.
- Wizualizacja warstwy 6, głowy 0 dla konkretnego zdania.
- Przechwycenie wag przed fine-tuningiem.
- Fine-tuning na zadaniu.
- Ponowna wizualizacja po dostrojeniu.
- Porównanie i interpretacja reorganizacji uwagi.

Technologie: HuggingFace Transformers, matplotlib (matshow), torch hooks.

Efekt: Pary heatmap przed/po + komentarz o zmianach.

———————————————————————————————

E. Question Answering

19. System extractive QA dla własnych dokumentów (Wikipedia PL)

Cel: System odpowiadający na pytania faktograficzne na podstawie pobranego artykułu z polskiej Wikipedii.

Zakres prac:

- Pobranie artykułu biblioteką wikipedia (PL).
- Extractive QA z BERT (deepset/bert-base-cased-squad2).
- Min. 8 pytań różnych typów.
- Zapis wyników z oceną i pewnością.
- Identyfikacja źródeł błędów.
- Statystyki trafności według typu pytania.

Technologie: HuggingFace, wikipedia, PyTorch.

Efekt: Działający system QA + raport o źródłach błędów.

———————————————————————————————

20. Konfrontacja: BERT (extractive) vs FLAN-T5 (generative)

Cel: Systematyczne porównanie podejścia ekstrakcyjnego i generatywnego.

Zakres prac:

- Przygotowanie 10 pytań z kontekstami.
- Generowanie odpowiedzi z obu modeli.
- Obliczenie EM i Token-F1.
- Znalezienie po 3 przykłady, gdzie każdy model jest lepszy, z wyjaśnieniem.
- Dyskusja nad ryzykiem halucynacji vs sztywnością extractive.

Technologie: HuggingFace, PyTorch, datasets.

Efekt: Tabela wyników, analiza przypadków i wnioski.

————————————————————————————————————

21. Pełnoprawny system RAG z rankingiem TF-IDF vs BM25

Cel: System Retrieval-Augmented Generation z porównaniem dwóch metod wyszukiwania.

Zakres prac:

- Korpus ~20 dokumentów.
- Implementacja retrieverów: TF-IDF oraz BM25 (rank-bm25).
- Dla 10 pytań: top-3 dokumenty z współczynnikami podobieństwa.
- Generowanie odpowiedzi na bazie pobranych dokumentów.
- Porównanie jakości zależnie od retrievera.
- Omówienie zasady „garbage in garbage out".

Technologie: scikit-learn, rank_bm25, HuggingFace, numpy.

Efekt: Działający system RAG z pokazywaniem źródeł + porównanie metod.

————————————————————————————————————

22. Multi-hop QA — odpowiadanie z wielu dokumentów

Cel: System łączący informacje z co najmniej dwóch dokumentów.

Zakres prac:

- Przygotowanie 5 pytań multi-hop (np. „W którym roku urodził się twórca Pythona?").

- Architektura iteracyjnego dzielenia pytania na podpytania i łączenia wyników.
- Ocena, jak radzi sobie pojedynczy BERT.
- Mechanizm chaining: pierwsza odpowiedź jako część kolejnego zapytania.
- Porównanie: pojedynczy BERT vs system wieloetapowy.

Technologie: HuggingFace, PyTorch, własna logika wnioskowania.

Efekt: Demonstracja ograniczeń pojedynczego modelu + prototyp multi-hop.

———————————————————————————————

23. Nowoczesny RAG z bazą wektorową (FAISS/Chroma)

Cel: Zaawansowany system RAG z semantycznym wyszukiwaniem wektorowym.

Zakres prac:

- Model embeddingów sentence-transformers.
- Podział na chunki (256–1024 tokeny z nakładką).
- Indeksowanie w FAISS lub Chroma (ANN).
- Pipeline: embed → wyszukaj → wzbogać prompt → generuj (FLAN-T5/GPT).
- Grounding/provenance odpowiedzi.
- Porównanie z systemem leksykalnym z Projektu 21.

Technologie: sentence-transformers, faiss-cpu/chromadb, HuggingFace, numpy.

Efekt: Funkcjonalny RAG z semantyczną bazą wiedzy i groundingiem.

———————————————————————————————

F. Metryki ewaluacyjne

24. Panel porównania metryk automatycznych (BLEU, ROUGE, METEOR, BERTScore)

Cel: Jednoczesne obliczanie i wizualizacja głównych metryk ewaluacji.

Zakres prac:

- Funkcje obliczające BLEU, ROUGE-1/2/L, METEOR, BERTScore.
- Zestaw hipotez o zróżnicowanej jakości.
- Zebranie wyników w DataFrame.
- Wizualizacja porównawcza (grouped bar chart).
- Dyskusja, dlaczego metryki się różnią.

Technologie: nltk, rouge_score, bert_score, pandas, matplotlib.

Efekt: Panel demonstrujący mocne/słabe strony każdej metryki.

———————————————————————————————

25. Badanie korelacji metryk z oceną ludzką

Cel: Sprawdzenie, która metryka najlepiej oddaje subiektywną ocenę człowieka.

Zakres prac:

- 10 par (referencja, hipoteza) o zróżnicowanej jakości.
- Zebranie ocen ludzkich (2–3 osoby, skala 1–5).
- Obliczenie BLEU-2, ROUGE-L, BERTScore-F1.
- Korelacja Spearmana (scipy.stats.spearmanr).
- Ranking metryk według korelacji.
- Wnioski, która metryka najlepiej przewiduje ocenę ludzką.

Technologie: nltk, rouge_score, bert_score, scipy, matplotlib.

Efekt: Tabela korelacji + wykresy rozrzutu.

———————————————————————————————

26. Porównanie modeli summaryzacji (BART vs DistilBART)

Cel: Generowanie streszczeń dwoma modelami i ewaluacja przez ROUGE, z analizą trade-offu jakość/szybkość/rozmiar.

Zakres prac:

- Wczytanie 5 artykułów z CNN/DailyMail.
- Generowanie streszczeń: facebook/bart-large-cnn i sshleifer/distilbart-cnn-12-6.
- Ewaluacja ROUGE-1/2/L.
- Pomiar czasu i rozmiaru.
- Analiza quality-per-cost i rekomendacja.

Technologie: HuggingFace, datasets, rouge_score, timeit.

Efekt: Tabela ROUGE + analiza kompromisu jakości vs szybkości.

———————————————————————————————

27. Wpływ długości tekstu na metryki — pułapki ewaluacji

Cel: Zbadanie, jak długość odpowiedzi zniekształca metryki.

Zakres prac:

- Odpowiedzi o długości 20, 50, 100, 200 słów na ten sam prompt.
- Obliczenie ROUGE-1, ROUGE-2, BLEU.
- Hipoteza: czy dłuższa odpowiedź zawsze oceniana wyżej?
- Sprawdzenie brevity penalty (BLEU) i recall (ROUGE).
- Identyfikacja mylących sytuacji.
- Praktyczne wskazówki interpretacji.

Technologie: nltk, rouge_score, HuggingFace, matplotlib.

Efekt: Wykresy zależności metryk od długości + lista pułapek.

———————————————————————————————

28. Perplexity jako miara kalibracji modelu

Cel: Zbadanie, co naprawdę mierzy perplexity.

Zakres prac:

- Funkcja compute_perplexity dla GPT-2 i drugiego modelu (np. BLOOM).
- Zestaw: zdania poprawne, bezsensowne, specjalistyczne.
- Obliczenie perplexity dla każdego tekstu i modelu.
- Interpretacja wartości.
- Porównanie kalibracji między modelami.
- Dyskusja ograniczeń metryki.

Technologie: HuggingFace (AutoModelForCausalLM), torch, math.

Efekt: Tabela wartości perplexity + wnioski o tym, co metryka mierzy, a czego nie.

———————————————————————————————

Projekty łączące kilka tematyk

29. Kompletny pipeline: klasyfikacja → QA → RAG dla firmowej bazy FAQ

Cel: Integracja umiejętności z kilku laboratoriów w spójną aplikację obsługującą bazę wiedzy firmy.

Zakres prac:

- Etap 1 (klasyfikacja): fine-tuning BERT do kierowania zapytania.
- Etap 2 (retrieval): korpus FAQ, indeksowanie, wyszukiwanie.
- Etap 3 (QA): extractive BERT lub generatywny FLAN-T5.
- Spięcie w jeden pipeline z pokazaniem źródła.
- Demo i wizualizacja etapów.

Technologie: HuggingFace, datasets, LangChain/Haystack (opcjonalnie), scikit-learn.

Efekt: Kompletny system obsługi FAQ pokazujący synergię komponentów.

———————————————————————————————

30. Halucynacje w Generative QA — analiza i mitygacja przez RAG

Cel: Wykrycie halucynacji w modelach generatywnych i pokazanie, jak RAG redukuje ryzyko.

Zakres prac:

- Część A (diagnoza): pytania otwarte, identyfikacja wiarygodnie brzmiących fałszywych odpowiedzi; typologia halucynacji.
- Część B (weryfikacja): sprawdzenie względem wiarygodnego źródła.
- Część C (mitygacja): RAG z bazą dokumentów, ponowne zadanie pytań.
- Miary: odsetek odpowiedzi zweryfikowanych, stopień groundedingu.
- Wnioski o skuteczności RAG.

Technologie: HuggingFace, sentence-transformers, FAISS/Chroma.

Efekt: Raport empiryczny o skali halucynacji + dowód ograniczania ich przez RAG.