Polski benchmark etnologiczny

for Speakleash aka Spichlerz

Wstęp

Pomysł stworzenia benchmarku etnologicznego dla modeli językowych narodził się z rozmów z Jankiem Sową – o tym, jak sztuczna inteligencja „rozumie” i co “wie” w temacie kultury i społeczności regionalnych.

Narzędzie ma pozwolić ocenić, jak modele radzą sobie z danymi osadzonymi w lokalnym, polskim kontekście.

Szukamy osób zainteresowanych współtworzeniem tego projektu badawczego.

Założenia

Opracowanie metody testowania?

Cel

Stworzenie testu zamkniętego w formie pytań i porównanie modeli językowych.

Rodzaje pytań

W tym miejscu warto byłoby dodać nasze przemyślenia dotyczące rodzajów pytań, np.: szczegółowe dotyczące jednego etnosu, przekrojowe, dotyczące pewnych wspólnych cech etnosu, pytania z zaprzeczeniem itp.

Bibliografia i źródło

Pytania i odpowiedzi nie powinny budzić wątpliwości, czy są prawidłowe, dlatego warto chociaż pokusić się o podstawy na których powstało pytanie testowe.

Proces

Testowanie modeli językowych => Scalanie wyników testów => Tworzenie publikacji.

Testowanie modeli językowych

Ten etap…

Scalanie wyników testów

Z poszczególnych testów

Tworzenie publikacji

Zadania do wykonania

Benchmark

- Test
    - Uzgodnienie rodzaju testu, kategorii pytań
    - Opracowanie pytań
- Oprogramowanie
    - skrypt wykonujący test dla określonego LLM-a - **benchmark_test_llm.py**
    - skrypt scalający wyniki testów dla każdego LLM-a w jeden rezultat porównawczy - **benchmark_merge_results.py**
    - skrypt generujący publikację pdf, html itp. na podstawie rezutaut

Formaty plików

Test - pytania

Format: CSV?

Dane:

- liczba porządkowa - [num]
- pytanie - [string]
- prawidłowa odpowiedź - [A | B | C | D]
- odpowiedź A - [string]
- odpowiedź B - [string]
- odpowiedź C - [string]
- odpowiedź D - [string]

Wynik - Odpowiedzi na pytania

Format: JSON?

Dane:

- identyfikator modelu; [string]
- nazwa modelu; [string]
- api; [vllm | openAI]
- url; [string]
- liczba wszystkich pytań; [num]
- podsumowanie - może dobrze byłoby to skorelować z typami ocen
    - suma prawidłowych odpowiedzi; [num]
    - suma nieprawidłowych odpowiedzi;[num]
    - suma z brakiem odpowiedzi; [num]
    - suma odpowiedzi niezgodna z oczekiwaniami; [num]
- Odpowiedzi; [list]
    - nr pytania z testu - [liczba porządkowa]
    - pytanie z testu - [string]
    - prawidłowa odpowiedź z testu - [A | B | C | D]
    - ocena - [prawidłowa | nieprawidłowa | brak odpowiedzi | odpowiedź niezgodna z oczekiwaniami]
    - odpowiedź udzielona przez badany llm - [A | B | C | D]
    - uzasadnienie udzielone przez badany llm - [string]
- Meta
    - wersja testu; [string]
    - domena testu; [string]
    - data; [datetime]
    - czas wykonania całego testu; [num; duration in ms]
    - średni czas odpowiedzi; [num; duration in ms]
    - konfiguracja sprzętowa; [string] jeśli dostępna

Podsumowanie - porównanie wyników

Format: JSON?

Dane:

- benchmark
    - meta dane o benchmarku; *do uzgodnienia. Dane o benchmarku, konieczne do wyprodukowania dokumentów końcowych typu html/pdf itp.*
- podsumowanie [list]; *element listy to właściwie przeniesienie danych z pojedynczego pliku odpowiedzi*
    - identyfikator modelu; [string]
    - nazwa modelu; [string]

- api; [vllm | openAI]
- url; [string]
- liczba wszystkich pytań; [num]
- Odpowiedzi; [object] - może dobrze byłoby to skorelować z typami ocen
  - suma prawidłowych odpowiedzi; [num]
  - suma nieprawidłowych odpowiedzi;[num]
  - suma z brakiem odpowiedzi; [num]
  - suma odpowiedzi niezgodna z oczekiwaniami; [num]

Skrypty - Instrukcje

Testowanie modelu językowego

```
python benchmark_test_llm.py --help
--test=”testPathName”         optional; default ./test.csv; can be url
--results=”resultPathName”    optional; default ./results/llmModelId.json;
--llm=”llmModelId”;           required; unique name of LLM e.g.
--llm-name=”model name”       optional; product name of LLM
--api=[vllm | openAI]         optional; default vllm
--url=”url”                   required only for API openAI;
--key=”key”                   required only for API openAI;
--interval=number             optional;
```

```
E.g.
python benchmark_test_llm.py --llm=”” --llm-name=”” --api=openai
--url=”https://” --key=”12345”
```

Scalanie wyników w jeden benchmark

```
python benchmark_merge_results.py --help
--results=resultFolderName    optional; default ./results/
--benchmark=benchmarkPathName optional; default ./benchmark/benchmark.json;
```