

Wymagane pliki w repozytorium i do wgrania w platformie ChallengeRocket:

**Readme.md** – Co to jest, jak zainstalować i jak uruchomić (wymagane).

Do tych plików dodajcie nazwy zespołów bez polskich znaków a spacje zamieńcie na podkreślenie, czyli np. nazwa\_zespołu="czerwone ślimaki"

output\_czerwone\_slimaki.txt

**output\_[nazwa\_zespołu].txt** – Zanonimizowany plik wynikowy (wymagane, format 1:1 z wejściem).

**performance\_[nazwa\_zespołu].txt** – Czas i opis sprzętu (wymagane).

**preprocessing\_[nazwa\_zespołu].md** – Opis przetwarzania, pozyskiwania danych (opcjonalnie)

**synthetic\_generation\_[nazwa\_zespołu].md** – Opis metody i przykłady generacji nowych danych (opcjonalne/punktowane w sekcji związanej z skutecznym modułem do generacji danych syntetycznych).

**presentation\_[nazwa\_zespołu].pdf** – Prezentacja – max 5 slajdów (wymagane)

### **Opis plików do oceny:**

#### **1. Readme.md – Wasza wizytówka**

To pierwszy punkt styku z Waszym projektem

**O co chodzi:** Jedno zdanie o Waszym podejściu (np. "Hybryda RegEx + BERT").

- **Gdzie co leży:** Mapa repozytorium (gdzie jest kod, dodatkowe dane, wagi modelu).
- **Jak to działa:** Krótki opis procesu anonimizacji i przyjętych założeń.
- **Instrukcja obsługi:** Komenda do instalacji (np. pip install...) i uruchomienia.

#### **2. output.txt – Wynik działania**

To plik, który wygeneruje Wasz algorytm na podstawie dostarczonych danych testowych.

- **Jeden do jednego:** Liczba linii w output.txt musi być **identyczna** jak w pliku wejściowym. Nie usuwajcie zdań!.
- **Formatowanie:** Używajcie formatu określonego w zadaniu, czyli nawiasów kwadratowych (np. [name], [pesel], [city]).
- **Porządek:** Zachowajcie kolejność wierszy, żebyśmy mogli automatycznie porównać Wasz wynik ze Złotym Standardem.

### **3. performance.txt – Metryki i Środowisko**

Tutaj chwalicie się wydajnością. Żebyśmy mogli uczciwie porównać Wasze wyniki z innymi, trzymajcie się tego schematu:

- **Time:** Czas przetwarzania w sekundach (od załadowania tekstu do zapisu wyniku). **Nie wliczajcie** czasu ładowania modelu/bibliotek do pamięci.
- **Hardware:** Na czym to liczyliście?
  - Typ: CPU / GPU (to kluczowe dla nas).
  - Model: np. "NVIDIA RTX 3060" lub "Intel i5-12400".
- **API:**
  - Czy użyto API PLLUM lub własnego modelu postawionego na zewnętrznym serwerze? **TAK/NIE**.
  - Jeżeli model inny niż PLLUM podajcie jaki to model i w jaki sposób postawiony

### **4. preprocessing.md – Obrabianie danych**

Opis procesu zebrania danych, czyszczenia, generowania itd.

### **5. synthetic\_generation.md – Generowanie danych**

Jeżeli podjęliście się wyzwania generacji danych syntetycznych (czyli zamiany tagów {name}, {city} z powrotem na losowe, ale sensowne słowa), opiszcie to tutaj.

W tym pliku zawrzyjcie:

- **Mechanizm:** Skąd bierzecie nowe dane? (Słowniki, biblioteki, czy model językowy?).
- **Walka z fleksją:** To jest kluczowe. Jak radzicie sobie z odmianą?
  - *Przykład problemu:* Model widzi Mieszkam w {city}.
  - *Sukces:* Zamienia na Mieszkam w Radomiu.
  - *Porażka:* Zamienia na Mieszkam w Radom. Opiszcie krótko, jak Wasz kod zapewnia poprawność gramatyczną.
- **Dbałość o sens:** Czy Wasze rozwiązanie bierze pod uwagę początkowe dane? Czy randomizacja odbywa się na zasadzie stworzenia inputów podobnych do danych wejściowych, czy nie ma do nich nawiązania? W jaki sposób dbacie o rozwiązanie?

- **Log z przykładami (Showcase):** Pokażcie 3-5 par zdań w formacie (im bardziej ciekawe, wymagające przykłady tym lepiej):
  - **Szablon (zanonimizowany):** "Spotkałem się z {name} w {city}."
  - **Wynik (syntetyczny):** "Spotkałem się z Kasią w Gdańsku."

## 5. Presentation.pdf – Prezentacja

- Treść dowolna, pochwalcie się co się udało zrobić. Ocenimy w tym miejscu Pomysłowość podejścia 😊