Modele językowe
Pracownia 2
Zajęcia 4 i 6

Na skosie znajdzie się link do materiałów potrzebnych do rozwiązywania zadań z tej listy. W każdym zadaniu powinieneś korzystać z modelu językowego dla języka polskiego, ze strony huggingface. Sugerowane są dwa stosunkowo niewielkie modele: PapuGaPT oraz polka, ale nie musisz się do nich ograniczać. Na stronie wykładu będą przykładowe programy, które możesz dowolnie wykorzystywać i modyfikować.

Zadanie 1. (**5+6p**) Zadanie to ma dość otwartą postać: masz 'zabawić się' w naukowca i sprawdzić, jak dobrze wybrany model językowy (w wielkości w przybliżeniu 100M-1B) radzi sobie z obliczeniami stosunkowo prostych wyrażeń arytmetycznych w scenariuszu few-shot learning.

To ,czym jest wyrażenie arytmetyczne, jak jest skomplikowane, jak konstruujemy prompty, jak dobieramy przykłady jest Twoją decyzją; aby dostać 5 punktów należy spełnić jedynie dwa wymagania:

1. Należy znaleźć scenariusz, który pokazuje, że wyniki modelu są w danym zadaniu są istotnie lepsze niż losowe

2. Wnioski z badania powinny być rzetelne, spisane w kodzie/notaniku.

Punkty dodatkowe przyznawane są przez prowadzących uznaniowo, mają one docenić osoby, które włożyły szczególnie dużo pracy w to zadanie, ewentualnie osiągnęły bardzo ciekawe wyniki. Do maksimum wlicza się 5 punktów.

Ciekawym pytaniem jest, czy da sie zaobserwować jakieś obciążenie (bias) modelu w tym zadaniu.

Zadanie 2. (**6p**) W zadaniu tym masz zmodyfikować procedurę generacji kolejnych tokenów w ten sposób, by:

1. Generowała dokładnie jedno słowo **z zadanego zbioru słów** (przy czym słowo to może mieć więcej niż jeden token) oraz

2. była ona wystarczająco efektywna (do wygenerowania w niezbyt długim czasie wielu kontynuacji prefiksu)

Oczywiście może to wymagać wygenerowania pewnej niezerowej liczby tokenów nadmiarowych. Następnie wykorzystaj tę procedurę do rozwiązania zadania Riddles[1] z Olimpiady Informatycznej w następujący sposób: utwórz prompt składający się z zagadki oraz jakiegoś łącznika, następnie wygeneruj (kilka razy?) możliwe jednosłowowe kontynuacje takiego prompta (oczywiście te kontynuacje powinny pochodzić ze zbioru możliwych odpowiedzi). Podaj, jaka jest dokładność takiego rozwiązania (dla więcej niż jednego modelu).

Uwaga: to zadanie będzie miało kontynuację, w której punkty będą przyznawane za osiągnięte rezultaty.

Zadanie 3. (**6p**) W zadaniu tym zajmiemy się ujednoznacznianiem tekstu. Zakładamy, że tekst z wariantami jest podany w następujący sposób:

> wprost|wyprosty|wyprostu|wyprost uwielbiała|wielbił|wielbiła|uwielbił|wielbiło|uwielbiał|uwielbiało|uwielbiały
> słuchać|osłuchać|słychać|usłuchać o|i|e|a|ó|ę|y|ą|u
> wartościach własnych|owłosionych macierzy|mocarz|macierzą|macierze|mocarza|mocarze|mocarzy|macierz

Tokenizację wykonujemy za pomocą metody split, ponadto (dla wygody) poprawne słowo zawsze jest na pierwszym miejscu (ale nie wolno Ci z tego w żaden sposób korzystać).

Napisz program, który znajduje poprawny wariant słowa. Twój program powinien wykorzystywać modele językowe, oraz jakiś niezachłanny algorytm optymalizacyjny. Może to być na przykład beam search, ale można wybrać też inne metody.

[1] `https://github.com/OlimpiadaAI/I-OlimpiadaAI/blob/main/first_stage/riddles/zagadki.ipynb`

Zadanie 4. (**6p**) W tym zadaniu będziemy generować zdania w języku naturalnym spełniające dodatkową właściwość: **wszystkie słowa powinny zaczynać się od tej samej litery**. Na SKOS będzie dostępna lista prefiksów, takich jak: „Prawdziwy piekarz przyprawia pieczywo pieprzem", Twoje generacje powinny zawsze zaczynać się od wylosowanego prefiksu i kończyć kropką (lub innym znakiem interpunkcjnym kończącym zdanie).

Twój program powinien:

1. Stosować zarówno top-k, jak i top-p.
2. Modyfikować rozkład prawdopodobieństwa tokenów
3. Generować „porządne" teksty, to znaczy z wyrazami składającymi się z liter, ze spacjami jako separatorami, z interpunkcją poprawnie formatowaną (na przykład przecinek powinien być doklejony do poprzedzającego go słowa)
4. Unikać powtórzeń
5. Generować kilka wariantów i wybierać z nich najlepszy, wg zdefiniowanego przez Ciebie kryterium

Wygenerowany tekst