Uczenie Nienadzorowane - projekt

Piotr Łuczak M.Sc.Eng.
piotr.luczak.1@p.lodz.pl

Zima 2022

1 Opis

Celem projektu jest próba zdekodowania "zakodowanego" przy użyciu nieznanych symboli pisma odręcznego. W oparciu o metody uczenia nienadzorowanego należy zbudować system pozwalający na konwersję tekstu pisanego kodem do alfabetu łacińskiego.

System powinien operować na reprezentacji ukrytej symboli i budować mapowanie w oparciu o częstotliwości występowania symboli. Ponadto, na bazie dostępnych zaszumionych i zniekształconych symboli należy zbudować listę "czystych" znaków oraz model pozwalający na odszumianie i eliminację zniekształceń symboli.

Ze względu na brak łatwo dostępnych tekstów odręcznych w zakodowanym języku angielskim, w ich miejscu wykorzystany zostanie tekst "Kubusia Puchatka"[1]. W tym celu konieczne jest przygotowanie programu automatycznie generującego odręczny tekst w wersji kodowanej oraz jawnej. Rolę kodowanych znaków pełnić będą znaki Hiragana z Kuzushiji-49 (podzbiór KMNIST), a alfabetu łacińskiego EMNIST w wariancie "ByMerge".

2 Szczegółowe wymogi

2.1 Organizacyjne

Rozwiązania mogą być przygotowywane w grupach maksymalnie 4 osobowych.

Prezentacje końcowe odbędzie się zgodnie z planem zajęć w dniach **2023-01-16 (grupa SIIUM 2)** oraz **2023-01-23 (grupa SIIUM 1)**, chyba że dojdzie do modyfikacji terminarza uczelni. W przypadku duże różnicy w ilości studentów w grupach SIIUM 1 i SIIUM 2, prezentacje zostaną rozłożone równomiernie pomiędzy oba terminy. W przypadku gdy całkowita ilość grup projektowych nie przekroczy 8, wszystkie prezentacje będą mogły się odbyć w jednym terminie. Sprawozdania powinny być dostarczone do **2023-01-30** (początek sesji).

Możliwe jest również zaproponowanie własnego tematu, pod warunkiem że proponowane rozwiązanie będzie wykorzystywało metody wymienione w sekcji 2.3.

[1] Książka tym roku dołączyła do domeny publicznej.

2.2 Generator danych

Pierwszym krokiem przygotowania danych jest wygenerowanie losowego mapowania znaków łacińskich na Hiragana. Zbiór Kuzushiji-49 posiada o jeden znak więcej niż EMNIST, przez co jeden znak oraz "znak iteracyjny" Hiragana pozostaną niewykorzystane.

Dane wejściowe mają mieć postać strony mającej 80 kolumn na 114 wierszy (proporcja zgodna z serią A), składające się z komórek o wymiarach 32x32 piksele. Przed umieszczeniem symbolu w komórce, powinien on być poddany losowemu zniekształceniu składającemu się z obrotu o maksymalnie $\pm 30°$ lub rozciągnięciu/zmniejszeniu w jednej z osi o maksymalnie $\pm 15\%$. Zniekształcenia losowane są niezależnie (znak może być poddany zarówno obrotowi i rozciągnięciu) z prawdopodobieństwem 0.3, a ich parametry (kąt, poziom rozciągnięcia, kierunek rozciągnięcia) powinny byc każdorazowo próbkowane z rozkładu jednostajnego. Wpisywanie symboli w komórki powinno uwzględniać występujące w tekście znaki białe - odpowiednio pozostawiając pustą komórkę lub przechodząc do wpisywania znaków do kolejnej linii. Na wypełnioną stronę należy dodatkowo nałożyć szum typu "sól i pieprz".' Możliwe, ale nieobowiązkowe, jest również dodanie krzywych o szerokości jednego piksela i losowej długości, symulujących zagniecenie kartki. Gotowa strona powinna być w formacie binarnym, tj. piksele mogą przyjmować tylko wartości 0 lub 1. Jeżeli w toku przetwarzania dojdzie do konwersji na format ciągły (float lub uint), należy dokonać progowania na 50% jasności.

Ostatnim etapem generacji danych jest wycięcie par symboli (łaciński i Hiragana) z przygotowanych stron "pisma odręcznego". Pary te, są jedynymi danymi do jakich algorytmy uczenia mogą mieć dostęp.

2.3 Wykorzystywane metody

Projekt powinien wykorzystywać przedstawione na zajęciach klasyczne metody klasteryzacji (np. spektralne), neuronowe nienadzorowane metody redukcji wymiarowości (np. autoenkodery), reprezentacji i wizualizacji przestrzeni wysokowymiarowych (np. UMAP) oraz nienadzorowane metody odszumiania (np. pamięci autoasociatywne).

Sam proces mapowania pomiędzy symbolami Hiragana i łacińskimi powinien zostać zbudowany w oparciu o łączenie ze sobą klastrów Hiragana i łacińskich o podobnej liczności. Pomimo znanej a priori prawdziwej ilości klastrów, wartość ta powinna być wyznaczona w oparciu o posiadane dane, zgodnie z kryteriami przedstawionymi na wykładzie (np. w oparciu o wartości własne lub "metodę łokcia"). Końcowa wersja systemu może dodatkowo wykorzystać tak powstałe pseudoetykiety do nauczenia modelu (np. asociatywnego lub neuronowego) pozwalającego na mapowanie pomiędzy alfabetami (w przestrzeni ukrytej lub jawnej).

2.4 Interfejs

Zbudowany system powinien pozwalać na dynamiczne generowanie stron "rękopisu" Hiragana i poddawanie jej mapowaniu na oczyszczony "rekopis" łaciński, wraz z opcją oczyszczenia symboli na stronie źródłowej. Interfejs powinien pozwalać na dowolne przełączanie sie pomiędzy wytrenowanymi modelami (np. klasteryzacji) i pozwalać na wizualizację ich działania (np. poprzez wizualizację uzyskanych klastrów - rozkład próbek w przestrzeni oraz medoidy).

2.5 Dozwolone narzędzia

Aplikacja musi być napisana w języku Python 3.9 lub nowszym i może wykorzystywać:

- NumPy i/lub PyTorch (lub TensorFlow 2). Jeżeli posiadany sprzęt na to pozwala warto rozważyć obliczenia przy użyciu CuPy.
- Gotowe implementacje MDS, TSNE, UMAP [2], DBSCAN, GaussianMixture i SpectralClustering .
- scikit-image i imageio.
- Dowolną bibliotekę GUI. Szczególnie warte rozważenia są: Dash, Streamlit i QT.

3 Kryteria oceniania

Projekt oceniany będzie na podstawie dwóch komponentów - sprawozdania i prezentacji działania aplikacji. Oba komponenty traktowane są równoważnie.

3.1 Sprawozdanie

Sprawozdanie z projektu powinno zawierać analizę skuteczności przetestowanych metod (wymienionych w sekcji 2.3) wraz porównaniem ich zaobserwowanych zalet i wad oraz rekomendacjami która pozwala na uzyskanie najlepszych wyników. Sprawozdanie powinno zwierać wszystkie rozważane (niekoniecznie wszystkie podane na wykładzie) metody, nawet jeżeli uzyskane przez nie wyniki były niesatysfakcjonujące.

3.2 Aplikacja

Prezentacja powinna być oparta o interfejs zgodny z opisem z sekcji 2.4 (oraz ewentualne slajdy) i w czasie 10 minut przedstawić zaimplementowane funkcjonalności, wraz z wykorzystanymi metodami.

[2] Prezentacja algorytmu w wykonaniu jego autora: `https://www.youtube.com/watch?v=nq6iPZVUxZU`