Wrocławskie Centrum Akademickie

Ministerstwo Cyfryzacji

# Jak robić zadania na Olimpiadzie Sztucznej Inteligencji

**Filip Manijak**

**19.02.2025**

# Co będziemy dzisiaj omawiać?

- Prezentacja zadania Zagadki
- Prezentacja zadania Śledzenie Obiektów
- Ogólne spostrzeżenie dotyczące zadań na OAI

# Kim jestem?

Jestem złotym medalistą IOAI i srebrnym medalistą IAIO. Obecnie aktywnie wspieram społeczność olimpijską – współtworzę MIKO, największe w Polsce koło matematyczne, prowadzące także zajęcia z informatyki, sztucznej inteligencji i fizyki.

MIKO

Ministerstwo Cyfryzacji

WCA Wrocławskie Centrum Akademickie

# Zadanie zagadki - Treść

https://github.com/OlimpiadaAI/I-OlimpiadaAI/blob/main/first_stage/riddles/zagadki.ipynb

Ministerstwo Cyfryzacji

Wrocławskie Centrum Akademickie

# Zadanie zagadki - Analiza narzędzi

Szukamy najprostszego podejścia, żeby później iteracyjnie je ulepszać. Ograniczenia:

- brak GPU – Nie można odpalić żadnego modelu językowego
- 50 zagadek na minutę - nasza funkcja musi być dość szybka
- Rozwiązanie bez internetu - odpadają wszystkie API-based podejścia

Dane:

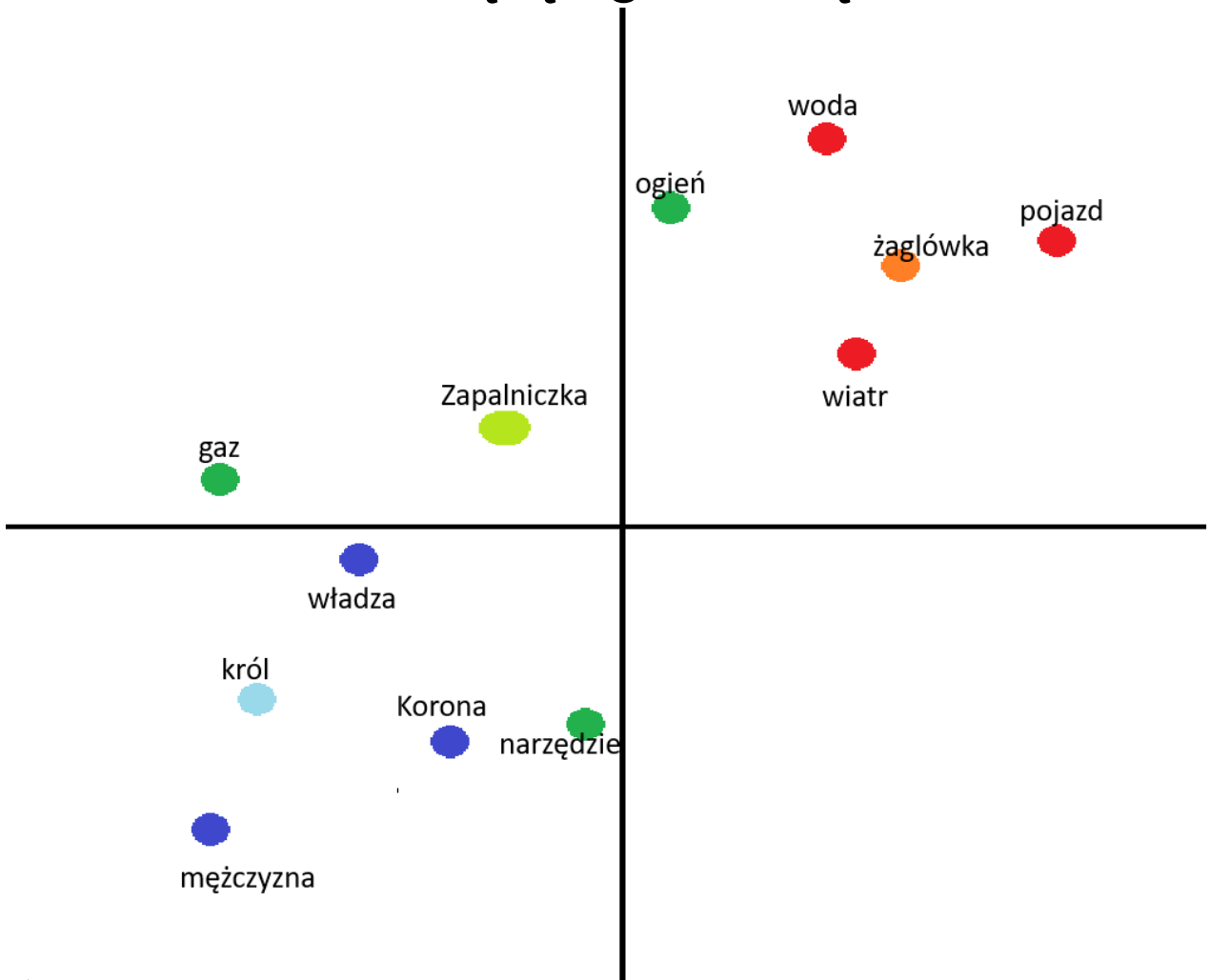
- test set
- Definicje słów z wikipedii – de facto train set
- superbazy_clean.txt – sugeruje, że dane trzeba będzie jakoś czyścić
- Embeddingi słów z Word2Vec

Ministerstwo Cyfryzacji

WCA Wrocławskie Centrum Akademickie

# Zadanie zagadki - Rozwiązanie

Dla każdej zagadki, jako odpowiedź zwracamy 20 najbliższych wektorów, do wektora będącego średnią wektorów z zdania



# Zadanie zagadki - Rozwiązanie

Poprawki

1. usunięcie "Stop Words", czyli np: a, aby, i, jeśli
2. Dodanie TFIDF score'a
   `https://pl.wikipedia.org/wiki/TFIDF`

pomysł wyliczenie wag, o które należy przeskalować wymiary embeddingów w funkcji kosztów

`https://colab.research.google.com/drive/10jLtN1rK6oZa17wKSIte6zRjLepMCj-d`

# Zadanie śledzenie obiektów - treść

https://github.com/OlimpiadaAI/I-OlimpiadaAI/tree/main/first_stage/object_tracking

# Zadanie śledzenie obiektów - Analiza narzędzi

Szukamy najprostszego podejścia, żeby później iteracyjnie je ulepszać. Ograniczenia:

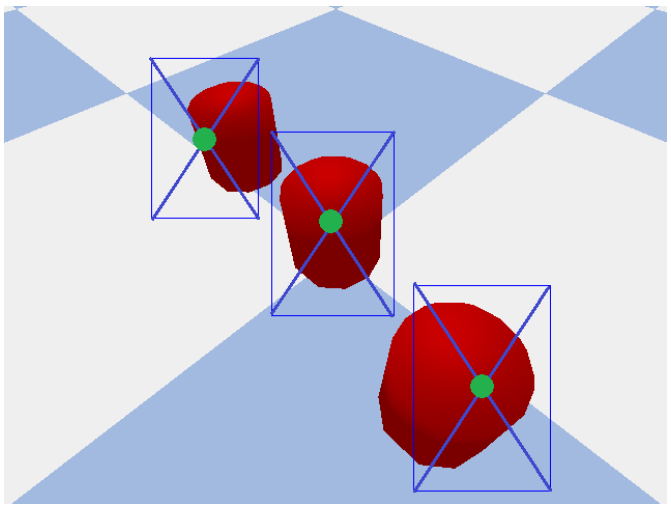
- 5 minut na środowisku bez GPU - odpadają wszystkie metody typu YOLO, czy naturalnego object detection.

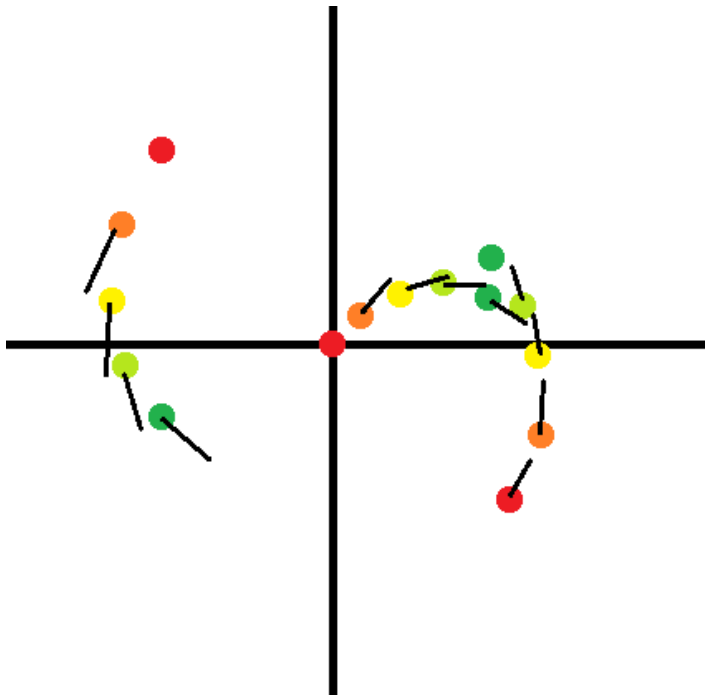
Dane:

- zaledwie 2 trasy przykładowe - duży problem z treningiem jakiś modeli

# zadanie śledzenie obiektów 1 i 2

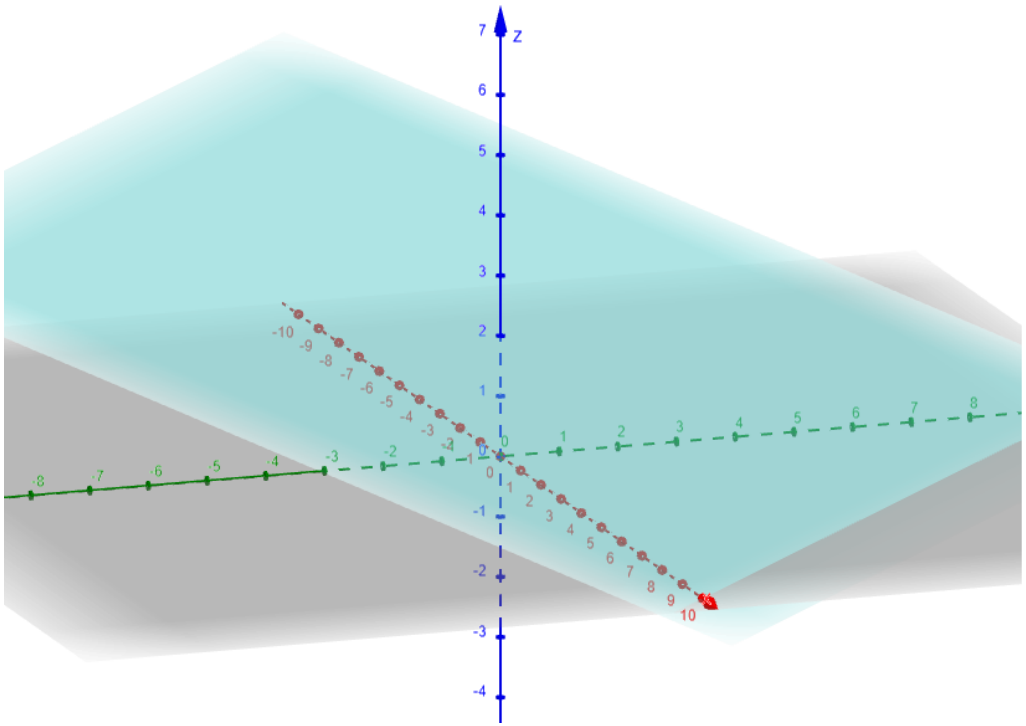
Predykcje mając już bounding boxy (a dokładniej ich środki), można zrobić używając pochodnej ruchu i poprzedniej pozycji.

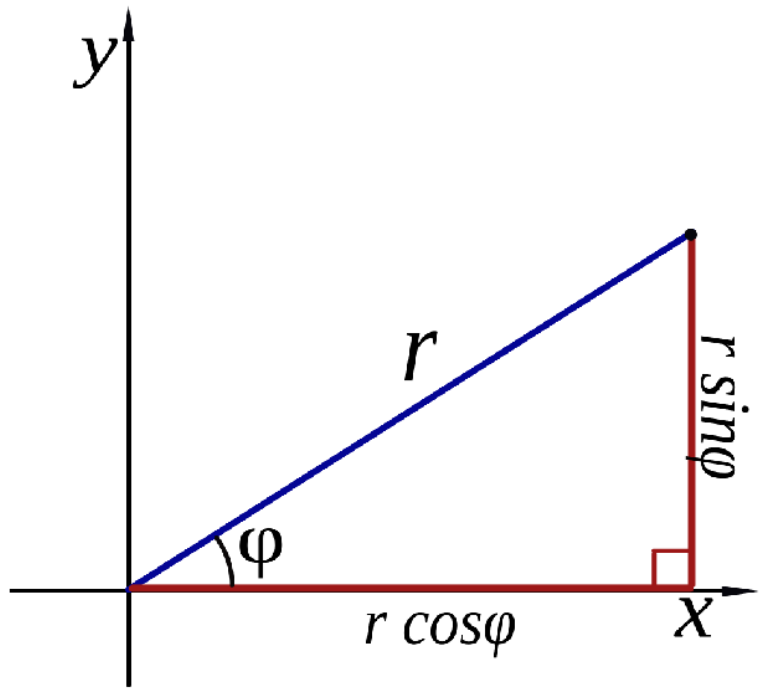




# zadanie śledzenie obiektów 1 i 2

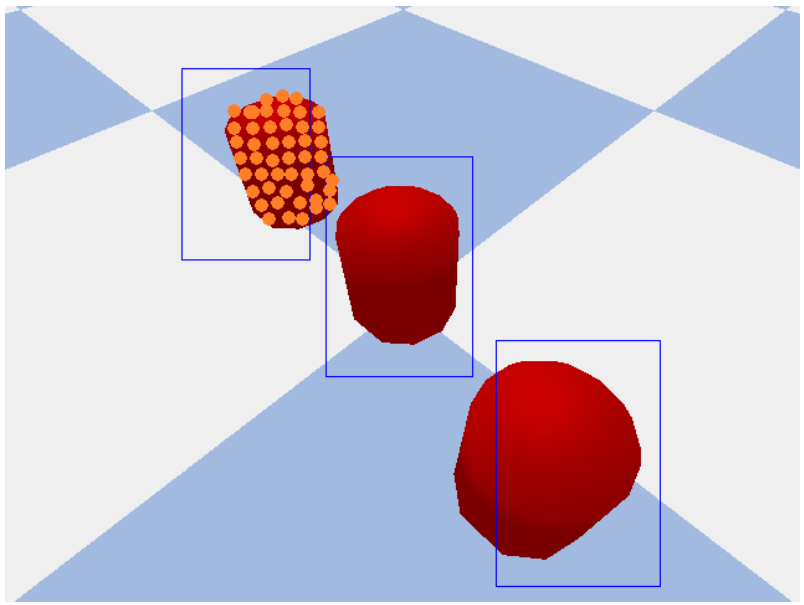
Można to wszystko z rzutować i opisać jako współrzędne biegunowe



# zadanie śledzenie obiektów 3

Pixele można wrzucić w algorytm K-means, i dostaniemy środki walców - co nam wystarcza. Dla optymalizacji można wziąć np. co 4 pixel.



# Kodzik

```python
def your_algorithm_task_3(images): # nie zmieniaj nazwy funkcji
    from sklearn.cluster import KMeans
    def odl(A, B):
        return (A[0] - B[0]) ** 2 + (A[1] - B[1]) ** 2
    def find_pixels_with_low_blue(image, threshold=20):
        # Otwórz obrazek
        # Pobierz szerokość i wysokość obrazka
        width, height = image.size
        # Zainicjalizuj listę do przechowywania współrzędnych pikseli
        low_blue_pixels = []
        # Iteruj przez każdy piksel obrazka
        for y in range(0, height, 3):
            for x in range(0, width, 3):
                # Pobierz wartości kolorów RGB piksela
                r, g, b, a = image.getpixel((x, y))
                # Sprawdź czy wartość koloru niebieskiego jest mniejsza niż próg
                if b < threshold:
                    # Jeśli tak, dodaj współrzędne piksela do listy
                    low_blue_pixels.append((x, y))
        # Zwróć listę współrzędnych pikseli z wartością koloru niebieskiego mniejszą niż próg
        return low_blue_pixels
    i=0
    Positions=[]
    for image in images:
        red_pix = find_pixels_with_low_blue(image, 20)
        all_perms = [[0, 1, 2], [1, 0, 2], [0, 2, 1], [1, 2, 0], [2, 0, 1], [2, 1, 0]]
        kmeans = KMeans(n_clusters=3, random_state=1, max_iter=10000,n_init=10).fit(red_pix)
        centers=kmeans.cluster_centers_
```

# Ogólne obserwacje i rady

- Zadania są podzielone na 2 główne sekcje – Deep Learning i klasyczny Machine Learning. Jak je rozpoznać? Sprawdzić czy w zadaniu można używać GPU.
- Na 1 etapie ważny jest time-management – w przeciwieństwie do OM czy OI zadań jest więcej, i często nie mają "sufitu" jeśli chodzi o liczbę punktów. Można zmarnować kilka dni np. na tuning hiperparametrów. (patrz zadanie prunning `https://github.com/OlimpiadaAI/I-OlimpiadaAI/tree/main/first_stage/pruning`)
- W zadaniach często są "Magicki" – prosta zmiana, lub nieszablonowe podejście daje bardzo dużo punków, trochę jak bruty na OI-u. Patrz kwantyzacja kolorów `https://github.com/OlimpiadaAI/I-OlimpiadaAI/tree/main/first_stage/color_quantization`.