

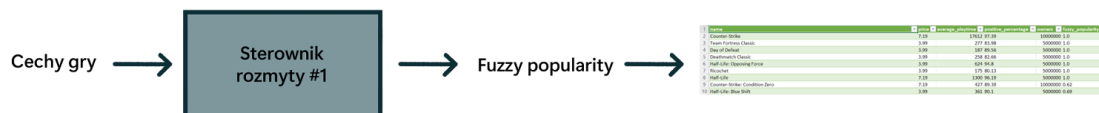
# Dokumentacja projektu

## Opis rozwiązywanego problemu

Steam jest najpopularniejszą i największą platformą z grami zawierającą ponad 50 000 gier. Mimo to, jej system do szukania gier wciąż nie oferuje wystarczająco dobrego narzędzia do znajdowania podobnych gier na podstawie preferencji użytkownika. Celem naszego projektu było utworzenie programu, który pozwoli użytkownikowi wpisać nazwę lubianej przez niego gry i na podstawie dostępnych informacji wyszukać w naszej bazie danych podobne tytuły.

## Szczegółowy opis zastosowanego algorytmu

Nasze rozwiązanie wykorzystuje dwa sterowniki rozmyte. Pierwszy sterownik jest aktywowany raz, przed uruchomieniem programu, i modyfikuje bazę danych o nowe parametry. Wylicza on dla każdej gry wartość „fuzzy popularity”. Parametry brane pod uwagę przy obliczaniu wartości to liczba pozytywnych recenzji, liczba osób posiadających grę, cena oraz długość gry.



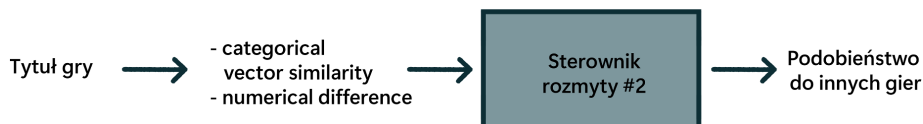
Powyższy rysunek pokazuje w uproszczony sposób jak działa algorytm obliczający wartość „fuzzy popularity”.

Zasady użyte w pierwszym sterowniku:

- 1) IF (Price IS average) OR (Game\_length IS average) OR (Rating IS average) OR (Number\_of\_owners IS average) THEN (Popularity IS average)
- 2) IF (Price IS expensive) AND (Game\_length IS long) AND (Rating IS positive) THEN (Popularity IS big)
- 3) IF (Price IS expensive) AND (Game\_length IS short) THEN (Popularity IS small)
- 4) IF (Price IS cheap) THEN (Popularity IS big)
- 5) IF (Rating IS negative) THEN (Popularity IS small)
- 6) IF (Rating IS positive) AND (Number\_of\_owners IS small) THEN (Popularity IS average)

7) IF (Rating IS average) AND (Price IS cheap) THEN (Popularity IS big)

Drugi sterownik uruchamiany jest wielokrotnie podczas działania programu. Dla wpisanego przez użytkownika tytułu algorytm po kolei porównuje grę do wszystkich dostępnych tytułów w bazie danych. Podczas tej operacji obliczane są dwa parametry – „*categorical vector similarity*” oraz „*numerical difference*”. „*Categorical vector similarity*” jest podobieństwem kosinusowym pomiędzy semantycznymi wektorowymi reprezentacjami wszystkich kategorii przyporządkowanych porównywanym grom w bazie danych. Wektorowa reprezentacja uzyskiwana jest poprzez sumę wektorów słów dla każdej nazwy kategorii przypisanej do danej gry, gdzie wartości pojedynczych wektorów zaczerpnięte są z pliku z pretrenowanymi wektorami uzyskanymi metodą *word2vec* udostępnianego przez firmę Google (szczegóły implementacji dostępne pod adresem: <https://code.google.com/archive/p/word2vec/>). Przykładowo, do pewnej gry może być przypisany następujący ciąg kategorii: “Multiplayer”, “FPS”, “Action”, “Sci-fi”. Dla każdego z tych słów odnajdywany jest odpowiadający mu wektor *word2vec* z pliku, a następnie są one wszystkie sumowane do postaci pojedynczego wektora, co daje nam ogólną semantyczną reprezentację jednocześnie charakteryzującą grę pod względem typu rozgrywki, tematyki, zawartości itp. Natomiast „*numerical difference*” porównuje wartości „*fuzzy popularity*” obliczone w pierwszym sterowniku i odejmuje je od siebie. Następnie obie te wartości wchodzi do drugiego sterownika, który skupia się na określeniu tego, czy dane gry faktycznie są do siebie podobne.



Powyższy rysunek pokazuje w uproszczony sposób jak obliczane jest podobieństwo dwóch gier (gry wybranej przez użytkownika i tytułu z bazy danych).

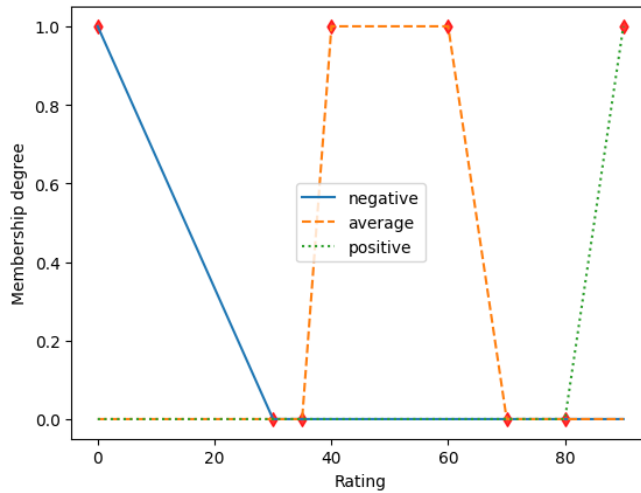
Zasady użyte w drugim sterowniku:

- 1) IF (Categorical\_similarity IS average) AND (Numerical\_difference IS average) THEN (Similarity IS average)
- 2) IF (Categorical\_similarity IS small) AND (Numerical\_difference IS big) THEN (Similarity IS small)
- 3) IF (Categorical\_similarity IS big) AND (Numerical\_difference IS small) THEN (Similarity IS big)

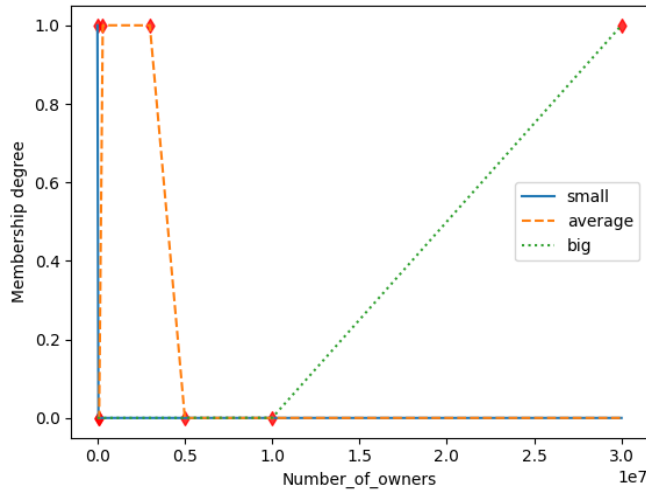
## Funkcje opisujące zmienne lingwistyczne

1) Sterownik obliczający wartość "fuzzy popularity"

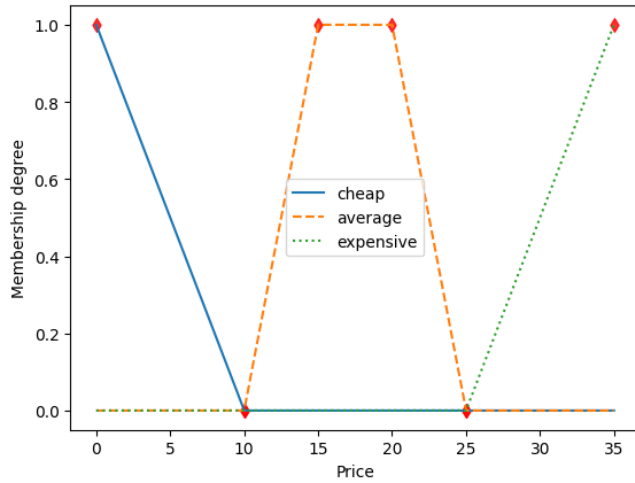
- wykres przedstawiający wartości zmiennej lingwistycznej opisującej ilość pozytywnych opinii o grze (Rating)



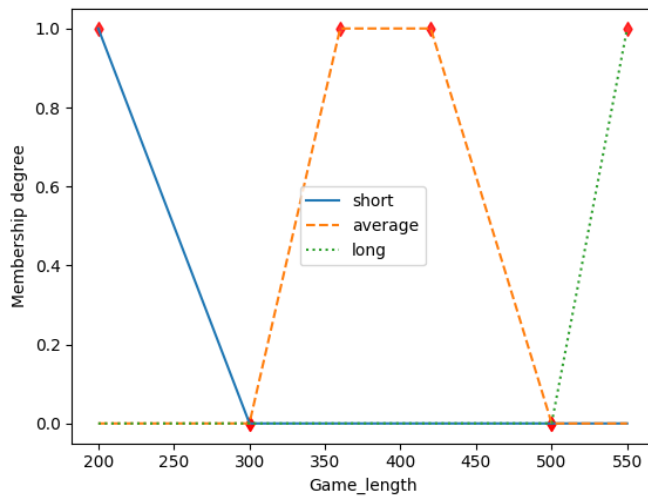
- wykres przedstawiający wartości zmiennej lingwistycznej opisującej ilość osób posiadających daną grę (Number of owners)



- wykres przedstawiający wartości zmiennej lingwistycznej cena (Price)

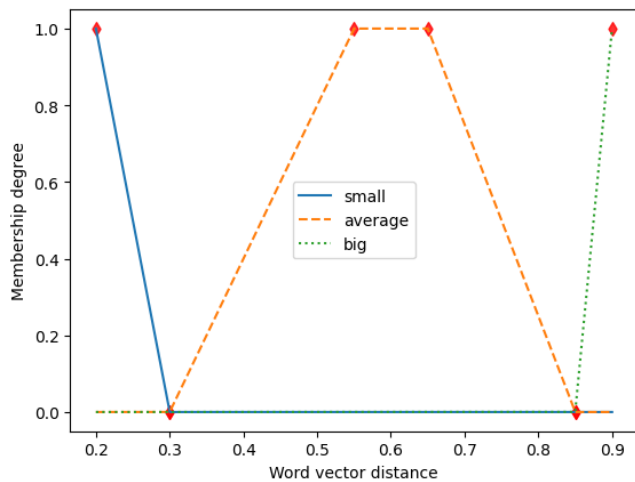


- wykres przedstawiający wartości zmiennej lingwistycznej długość gry (Game length)

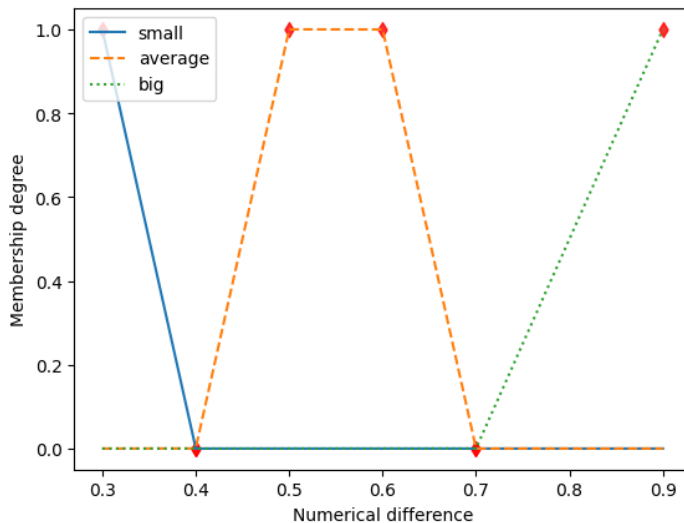


2) Sterownik obliczający podobieństwo dwóch gier

- wykres przedstawiający wartości zmiennej lingwistycznej dla podobieństwa między dwoma wektorami (Word vector distance)



- wykres przedstawiający wartości zmiennej lingwistycznej dla różnicy wartości parametru `fuzzy_popularity`, który obliczany jest w pierwszym sterowniku (Numerical difference)



## Opis wykonanej implementacji

Program został napisany w języku Python, przy użyciu biblioteki Simful. Reprezentacja wektorowa danych kategoryjnych została utworzona na podstawie pretrenowanych wektorów `word2vec`. Kod jest dostępny w repozytorium: <https://git.wmi.amu.edu.pl/s449288/fuzzy-game-recommender>.

## Ewaluacja systemu

Do oceny jakości zwracanych wyników posłużył współczynnik podobieństw Jaccarda, liczony ze wzoru

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

Dla każdej z 10 losowo wybranych gier ze zbioru danych system wygenerował 10 rekomendacji. Współczynnik podobieństwa Jaccarda został obliczony dla kategorii każdej z 10 gier oraz przypisanych do niej 10 rekomendacji, dając łącznie 100 wyników, następnie wynik został uśredniony.

```
Avg Jaccard score for random 10 games and 10 recomendations for
each game (100 total): 0.6191152683291523
```

Wynik zestawiony jest z baselinem, który powstał poprzez uruchomienie systemu w trybie generowania rekomendacji w sposób losowy

```
Avg Jaccard score for random 10 games and 10 recomendations for each game
(100 total): 0.2104311054230962
```

Dla 100 porównań, średnie podobieństwo zbiorów kategorii zestawionych gier wynosiło ok. 62%, dla gier wygenerowanych przez system oparty o logikę rozmytą, dla systemu losowego było to ok. 21%, co oznacza, że dobrane przez system gry mogą być uznane za podobne do gier wejściowych.

Przykład obiektu przechowującego wynik, wygenerowanego przez skrypt ewaluacyjny w pliku Fuzzy\_presentation.ipynb

```
{
  "fuzzy_rec":{
    "sel_game":["casual', 'indie', 'simulation', 'single-player']",
    "fuzzy_game":["casual', 'indie', 'multi-player', 'simulation', 'single-player']",
    "jaccard_tag_score":0.8
  }
}
```

Największym problem działania systemu jest ryzyko zwrócenia nie relewantnych rekomendacji w sytuacji, w której wybrana gra posiada niewiele przypisanych kategorii. W wykorzystywanym zestawie danych (liczba kategorii waha się w przedziale od kilku do kilkunastu w zależności od gry). W przypadkach takich zmienna lingwistyczna *categorical vector similarity* nie może być wyliczona z dostateczną precyzją, co przekłada się na działanie całego sterownika rozmytego #2. Jakkolwiek jest to przede wszystkim wada samego zestawu danych, to najpewniej możliwe jest również zniwelowanie jej w pewnym stopniu poprzez zaimplementowanie dodatkowych zmiennych lingwistycznych, na których podstawie swoją oceną podobieństwa pomiędzy grami mógłby obliczyć system.