Computational Neuroscience, Digital Intelligence Laboratory

KONTEKSTOWE ASOCJACJE REGULARNE

Waldemar Wietrzykowski

Instytut Inteligencji Stosowanej, 27 lipca 2022

Streszczenie *w niniejszej pracy wprowadzono definicję kontekstowych asocjacji regularnych i opracowano program do ich realizacji. Uproszczono i zoptymalizowano mechanizm konsolidacji asocjacji regularnych. Zauważono też, że kontekstowe asocjacje regularne zastosowane do kolorów obrazu dokonują automatycznego jego rozbioru z punktu widzenia zawartych w nim elementów.*

Wstęp

Podstawą niniejszej pracy jest model asocjacyjnej maszyny Turinga zrealizowany w pracy „Implementacja asocjacyjnej maszyny Turinga" a wprowadzony w pracy „Asocjacje regularne". Na podstawie tego modelu opracowano definicję kontekstowej asocjacji regularnej a także uproszczono i zoptymalizowano mechanizm konsolidacji asocjacji.

Prawa kontekstowych asocjacji regularnych

Prawa asocjacji regularnych, podane w pracy „Asocjacje regularne", po dokonaniu pewnych zmian zostały przeniesione do praw kontekstowych.

Prawa te sprowadzają się po następujących praw:

1) Para produkcji z kontekstem αp i $p\beta$, gdzie α *i* β są lewo i prawostronnym kontekstem produkcji p, powiązana ze sobą relacją następstwa $\alpha p \rightarrow p\beta$, reprezentowana jest przez asocjację jednostkową $\alpha p\ p\beta$, gdzie αp jest poprzednikiem asocjacji a $p\beta$ jest następnikiem asocjacji.

$$\alpha p \rightarrow p\beta \Rightarrow \alpha p\ p\beta$$

2) Dwie pary produkcji z kontekstem αp i $p\beta 1$ oraz αp i $p\beta 2$ powiązanych ze sobą relacją następstwa $\alpha p \rightarrow p\beta 1$ i $\alpha p \rightarrow p\beta 2$ reprezentowane są przez asocjację αp $(p\beta 1, p\beta 2)$, gdzie αp jest poprzednikiem asocjacji a $(p\beta 1, p\beta 2)$ jest następnikiem asocjacji. Zapis $(p\beta 1, p\beta 2)$ oznacza, że produkcje $p\beta 1$ i $p\beta 2$ są alternatywne $(p\beta 1 \vee p\beta 2)$. Asocjacja αp $(p\beta 1, p\beta 2)$ oznacza, że lewostronnym kontekstem produkcji p jest α, a prawostronnym kontekstem p są $\beta 1$ lub $\beta 2$.

$$\alpha p \rightarrow p\beta 1, \alpha p \rightarrow p\beta 2 \Rightarrow \alpha p\ (p\beta 1, p\beta 2)$$

3) Dwie pary produkcji z kontekstem $\alpha 1p$ i $p\beta$ oraz $\alpha 2p$ i $p\beta$ powiązanych ze sobą relacją następstwa $\alpha 1p \rightarrow p\beta$ i $\alpha 2p \rightarrow p\beta$ reprezentowane są przez asocjację $(\alpha 1p, \alpha 2p)$ $p\beta$, gdzie $(\alpha 1p, \alpha 2p)$ jest poprzednikiem asocjacji a $p\beta$ jest następnikiem asocjacji. Zapis $(\alpha 1p, \alpha 2p)$ oznacza, ze produkcje $\alpha 1p$ i $\alpha 2p$ są alternatywne $(\alpha 1p \vee \alpha 2p)$. Asocjacja $(\alpha 1p, \alpha 2p)$ $p\beta$ oznacza, że lewostronnym kontekstem produkcji p są $\alpha 1$ lub $\alpha 2$, a prawostronnym kontekstem produkcji p jest β.

$\alpha 1p \rightarrow p\beta, \alpha 2p \rightarrow p\beta \Rightarrow (\alpha 1p, \alpha 2p)\ p\beta$

4) Ciąg produkcji: *p1*, *p2*, *p3*, *p4*, p5... reprezentowany jest przez ciąg kontekstowych asocjacji elementarnych: *p1p2 p2p3*, *p2p3 p3p4*, ...

p1,*p2*,*p3*,*p4*,*p5*... => *p1p2 p2p3*, *p2p3 p3p4*, ...

Produkcje *p1* i *p3* są lewo i prawostronnym kontekstem produkcji *p2* a produkcje *p2* i *p4* są lewo i prawostronnym kontekstem produkcji *p3*.

5) Ciąg kontekstowych asocjacji jednostkowych *α1p pβ1*, *α1p pβ2*, *α2p pβ1*, *α2p pβ2* daje kontekstową asocjację złożoną: (*α1p*, *α2p*) (*pβ1*, *pβ2*). Ogólnie kontekstową asocjacją złożoną nazywamy asocjację *P N* = (*α1p*, *α2p*, *α3p*,...) (*pβ1*, *pβ2*, *pβ2*,...) gdzie: *P* jest zbiorem alternatywnych poprzedników zawierającym alternatywne lewostronne konteksty produkcji p, a *N* jest zbiorem alternatywnych następników zawierającym alternatywne prawostronne konteksty produkcji p. Kontekstowa asocjacja złożona dotyczy jednej produkcji p oraz jej lewostronnych i prawostronnych kontekstów.

Dowód:

$\alpha 1p\ p\beta 1, \alpha 1p\ p\beta 2, \alpha 2p\ p\beta 1, \alpha 2p\ p\beta 2 = \alpha 1p\ (p\beta 1, p\beta 2), \alpha 2p\ (p\beta 1, p\beta 2) = (\alpha 1p, \alpha 2p)\ (p\beta 1, p\beta 2)$

6) Dwie asocjacje *A B* i *C D* są ortogonalne (rozłączne) jeżeli iloczyn zbiorów $A \cap C$ jest zbiorem pustym i iloczyn zbiorów $B \cap D$ jest zbiorem pustym.

$(A \cap C = \emptyset) \wedge (B \cap D = \emptyset)$

7) Dwie asocjacje *A B* i *C D* są nieortogonalne (nierozłączne) jeżeli iloczyn zbiorów $A \cap C$ nie jest zbiorem pustym lub iloczyn zbiorów $B \cap D$ nie jest zbiorem pustym.

$(A \cap C \neq \emptyset) \vee (B \cap D \neq \emptyset)$

8) Prawo konsolidacji asocjacji.

Dwie nieortogonalne (nierozłączne) asocjacje *P1 N1* oraz *P2 N2* można połączyć w jedną asocjację *P N*, gdzie *P* jest sumą zbiorów *P1* i *P2*, a *N* jest sumą zbiorów *N1* i *N2*, co można przedstawić następująco:

Jeżeli

$(P1 \cap N1 \neq \emptyset) \vee (P2 \cap N2 \neq \emptyset)$

to

$P1\ N1, P2\ N2 \Rightarrow (P1 \cup P2)\ (N1 \cup N2)$

Asocjacje regularne kolorów

Jednym z zastosowań kontekstowych asocjacji regularnych są asocjacje regularne kolorów. Podczas szeregowego skanowania pikseli obrazu aktualnie skanowany piksel posiada swój kontekst, jakim są piksel poprzedni - lewy i piksel następny - prawy, i jest reprezentowany on przez kontekstową asocjację jednostkową, jaką jest: (kontekst lewy, piksel), (piksel, kontekst prawy)

Asocjację jednostkową możemy przedstawić przy pomocy następującego zestawu pikseli:

e = [(pixels[x,y], pixels[x+1,y]), (pixels[x+1,y], pixels[x+2,y])]

posiadającego wartość:

[((Rp, Gp, Bp), (R, G, B)), ((R, G, B), (Rn, Gn, Bn))]

Przykład wartości asocjacji jednostkowej:

e = [((128, 255, 255), (128, 128, 0)), ((128, 128, 0), (76, 61, 175))]

Kontekstowa asocjacja złożona

Jeżeli zbiór kontekstowych asocjacji jednostkowych poddamy procesowi konsolidacji uzyskamy listę rozłącznych kontekstowych asocjacji złożonych. Każda kontekstowa asocjacja złożona posiada swój indeks, a jest nim położenie na liście, i ma w poprzedniku zbiór lewostronnych kontekstów, a w następniku zbiór prawostronnych kontekstów tego koloru.

asoc = [{((αR1, αG2, αB3), (R, G, B)), ((αR4, αG5, αB6), (R, G, B)), ... },
{((R, G, B), (βR1, βG2, βB3)), ((R, G, B), (βR4, βG5, βB6)), ... }]

gdzie:
(αR_i, αG_i, αB_i) - lewostronny kontekst koloru (R, G, B)
(βR_j, βG_j, βB_j) - prawostronny kontekst koloru (R, G, B)

Przykład 1:

asoc = [{((128, 128, 128), (128, 128, 128)), ((128, 255, 255), (128, 128, 128)), ((255, 0, 0), (128, 128, 128)), ((0, 0, 0), (128, 128, 128))}, {((128, 128, 128), (0, 0, 0)), ((128, 128, 128), (255, 255, 255)), ((128, 128, 128), (128, 128, 128)), ((128, 128, 128), (128, 255, 255)), ((128, 128, 128), (255, 0, 0))}]

gdzie:
((255, 0, 0), (128, 128, 128)) - produkcja kontekstowa lewostronna
((128, 128, 128), (0, 0, 0)) - produkcja kontekstowa prawostronna
(128, 128, 128) - kolor, którego dotyczą powyższe konteksty

Asocjację tą można przekształcić też do postaci:

asoc = (((128, 128, 128), (128, 255, 255), (255, 0, 0), (0, 0, 0)), (128, 128, 128), ((0, 0, 0), (255, 255, 255), (128, 128, 128), (128, 255, 255), (255, 0, 0)))

lub do listy odpowiadających kolorów:

asoc = 

Kwadrat posiadający białe krawędzie oznacza kolor, którego dotyczą konteksty. Kwadraty poprzedzające dotyczą kontekstów lewostronnych, a kwadraty następujące dotyczą kontekstów prawostronnych.

Analiza obrazu metodą konsolidacji kontekstowych asocjacji kolorów

Mamy dany następujący obraz graficzny:

Z pikseli kolejnych wierszy generujemy kontekstowe asocjacje jednostkowe, które konsolidujemy. Uzyskujemy listę 17 rozłącznych kontekstowych asocjacji złożonych dotyczące następujących kolorów:

[(76, 61, 175), (255, 255, 0), (128, 128, 0), (255, 255, 255), (0, 0, 0), (128, 255, 255), (255, 255, 128), (128, 0, 128), (128, 128, 64), (255, 0, 0), (128, 0, 0), (128, 0, 0), (255, 0, 255), (128, 128, 128), (192, 192, 192), (192, 192, 192), (128, 0, 0)]

lub w postaci graficznej:

Na liście uzyskanych asocjacji znajdują się następujące asocjacje złożone (podano: indeks asocjacji na liście, ilość kontekstów lewostronnych, ilość kontekstów prawostronnych, postać asocjacji) :

list_asoc[0] = 7, 6
[{((128, 128, 0), (76, 61, 175)), ((128, 128, 64), (76, 61, 175)), ((255, 255, 0), (76, 61, 175)), ((76, 61, 175), (76, 61, 175)), ((255, 0, 0), (76, 61, 175)), ((0, 0, 0), (76, 61, 175)), ((255, 0, 255), (76, 61, 175))},

{((76, 61, 175), (255, 0, 0)), ((76, 61, 175), (0, 0, 0)), ((76, 61, 175), (76, 61, 175)), ((76, 61, 175), (255, 0, 255)), ((76, 61, 175), (255, 255, 0)), ((76, 61, 175), (128, 128, 0))}]

Zestaw użytych pikseli:
(((128, 128, 0), (128, 128, 64), (255, 255, 0), (76, 61, 175), (255, 0, 0), (0, 0, 0), (255, 0, 255)), (76, 61, 175), ((255, 0, 0), (0, 0, 0), (76, 61, 175), (255, 0, 255), (255, 255, 0), (128, 128, 0)))

Zestaw użytych kolorów
Ilość kolorów: 14

Asocjacja ta rozpoznaje następujące elementy obrazu:

list_asoc[1] = 8 , 9
[{((0, 0, 0), (255, 255, 0)), ((128, 128, 0), (255, 255, 0)), ((255, 255, 255), (255, 255, 0)), ((128, 255, 255), (255, 255, 0)), ((128, 128, 64), (255, 255, 0)), ((255, 0, 0), (255, 255, 0)), ((255, 255, 0), (255, 255, 0)), ((76, 61, 175), (255, 255, 0))}, {((255, 255, 0), (128, 128, 64)), ((255, 255, 0), (128, 0, 0)), ((255, 255, 0), (255, 255, 255)), ((255, 255, 0), (0, 0, 0)), ((255, 255, 0), (128, 255, 255)), ((255, 255, 0), (255, 0, 0)), ((255, 255, 0), (76, 61, 175)), ((255, 255, 0), (255, 255, 0)), ((255, 255, 0), (128, 128, 0))}]

Zestaw użytych pikseli:
(((0, 0, 0), (128, 128, 0), (255, 255, 255), (128, 255, 255), (128, 128, 64), (255, 0, 0), (255, 255, 0), (76, 61, 175)), (255, 255, 0), ((128, 128, 64), (128, 0, 0), (255, 255, 255), (0, 0, 0), (128, 255, 255), (255, 0, 0), (76, 61, 175), (255, 255, 0), (128, 128, 0)))

Zestaw użytych kolorów:
Ilość kolorów: 18

Asocjacja ta rozpoznaje następujące elementy obrazu:

list_asoc[2] = 6 , 6
[{((255, 255, 0), (128, 128, 0)), ((255, 255, 255), (128, 128, 0)), ((128, 255, 255), (128, 128, 0)), ((0, 0, 0), (128, 128, 0)), ((128, 128, 0), (128, 128, 0)), ((76, 61, 175), (128, 128, 0))}, {((128, 128, 0), (76, 61, 175)), ((128, 128, 0), (0, 0, 0)), ((128, 128, 0), (128, 255, 255)), ((128, 128, 0), (255, 255, 255)), ((128, 128, 0), (255, 255, 0)), ((128, 128, 0), (128, 128, 0))}]

Zestaw użytych pikseli:
(((255, 255, 0), (255, 255, 255), (128, 255, 255), (0, 0, 0), (128, 128, 0), (76, 61, 175)), (128, 128, 0), ((76, 61, 175), (0, 0, 0), (128, 255, 255), (255, 255, 255), (255, 255, 0), (128, 128, 0)))

Zestaw użytych kolorów
Ilość kolorów: 13

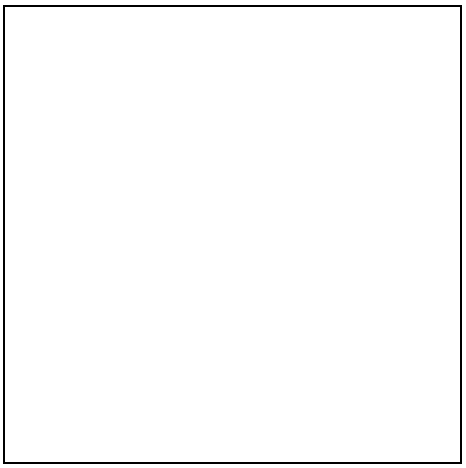
Asocjacja ta rozpoznaje następujące elementy obrazu:

list_asoc[3] = 12 , 9
[{((128, 128, 64), (255, 255, 255)), ((255, 255, 0), (255, 255, 255)), ((255, 0, 0), (255, 255, 255)), ((255, 0, 255), (255, 255, 255)), ((255, 255, 128), (255, 255, 255)), ((0, 0, 0), (255, 255, 255)), ((128, 0, 0), (255, 255, 255)), ((128, 128, 0), (255, 255, 255)), ((128, 128, 128), (255, 255, 255)), ((192, 192, 192), (255, 255, 255)), ((255, 255, 255), (255, 255, 255)), ((128, 0, 128), (255, 255, 255))}, {((255, 255, 255), (255, 255, 128)), ((255, 255, 255), (255, 0, 0)), ((255, 255, 255), (0, 0, 0)), ((255, 255, 255), (128, 0, 128)), ((255, 255, 255), (255, 255, 0)), ((255, 255, 255), (128, 128, 0)), ((255, 255, 255), (255, 0, 255)), ((255, 255, 255), (128, 128, 64)), ((255, 255, 255), (255, 255, 255))}]

Zestaw użytych pikseli:
(((128, 128, 64), (255, 255, 0), (255, 0, 0), (255, 0, 255), (255, 255, 128), (0, 0, 0), (128, 0, 0), (128, 128, 0), (128, 128, 128), (192, 192, 192), (255, 255, 255), (128, 0, 128)), (255, 255, 255), ((255, 255, 128), (255, 0, 0), (0, 0, 0), (128, 0, 128), (255, 255, 0), (128, 128, 0), (255, 0, 255), (128, 128, 64), (255, 255, 255)))

Zestaw użytych kolorów
Ilość kolorów: 22

Asocjacja ta rozpoznaje następujące elementy obrazu (ramka oznacza mało widoczne elementy):



list_asoc[4] = 12 , 13
[{((128, 128, 0), (0, 0, 0)), ((128, 128, 128), (0, 0, 0)), ((128, 0, 128), (0, 0, 0)), ((255, 255, 255), (0, 0, 0)), ((128, 255, 255), (0, 0, 0)), ((128, 128, 64), (0, 0, 0)), ((255, 255, 0), (0, 0, 0)), ((255, 0, 0), (0, 0, 0)), ((76, 61, 175), (0, 0, 0)), ((255, 255, 128), (0, 0, 0)), ((0, 0, 0), (0, 0, 0)), ((255, 0, 255), (0, 0, 0))}, {((0, 0, 0), (128, 0, 128)), ((0, 0, 0), (255, 0, 255)), ((0, 0, 0), (255, 255, 0)), ((0, 0, 0), (128, 128, 0)), ((0, 0, 0), (128, 128, 128)), ((0, 0, 0), (255, 0, 0)), ((0, 0, 0), (128, 128, 64)), ((0, 0, 0), (255, 255, 255)), ((0, 0, 0), (0, 0, 0)), ((0, 0, 0), (128, 255, 255)), ((0, 0, 0), (255, 255, 128)), ((0, 0, 0), (76, 61, 175)), ((0, 0, 0), (192, 192, 192))}]

Zestaw użytych pikseli:
(((128, 128, 0), (128, 128, 128), (128, 0, 128), (255, 255, 255), (128, 255, 255), (128, 128, 64), (255, 255, 0), (255, 0, 0), (76, 61, 175), (255, 255, 128), (0, 0, 0), (255, 0, 255)), (0, 0, 0), ((128, 0, 128), (255,

0, 255), (255, 255, 0), (128, 128, 0), (128, 128, 128), (255, 0, 0), (128, 128, 64), (255, 255, 255), (0, 0, 0), (128, 255, 255), (255, 255, 128), (76, 61, 175), (192, 192, 192)))

Zestaw użytych kolorów
Ilość kolorów: 26

Asocjacja ta rozpoznaje następujące elementy obrazu:

list_asoc[5] = 6 , 6
[{((128, 255, 255), (128, 255, 255)), ((128, 128, 128), (128, 255, 255)), ((255, 255, 0), (128, 255, 255)), ((255, 0, 0), (128, 255, 255)), ((0, 0, 0), (128, 255, 255)), ((128, 128, 0), (128, 255, 255))}, {((128, 255, 255), (128, 255, 255)), ((128, 255, 255), (255, 0, 0)), ((128, 255, 255), (0, 0, 0)), ((128, 255, 255), (255, 255, 0)), ((128, 255, 255), (128, 128, 0)), ((128, 255, 255), (128, 128, 128))}]

Zestaw użytych pikseli:
(((128, 255, 255), (128, 128, 128), (255, 255, 0), (255, 0, 0), (0, 0, 0), (128, 128, 0)), (128, 255, 255), ((128, 255, 255), (255, 0, 0), (0, 0, 0), (255, 255, 0), (128, 128, 0), (128, 128, 128)))

Zestaw użytych kolorów
Ilość kolorów: 13

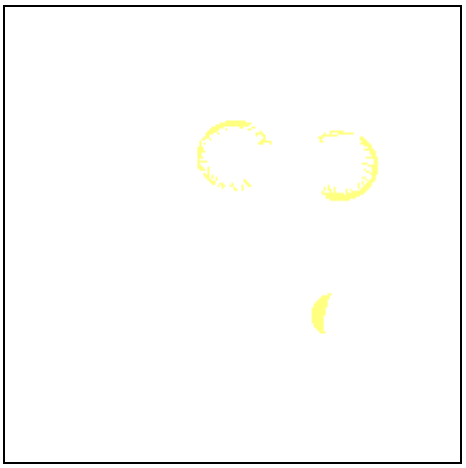
Asocjacja ta rozpoznaje następujące elementy obrazu:

list_asoc[6] = 5 , 5
[{((192, 192, 192), (255, 255, 128)), ((255, 255, 255), (255, 255, 128)), ((255, 0, 0), (255, 255, 128)), ((255, 255, 128), (255, 255, 128)), ((0, 0, 0), (255, 255, 128))}, {((255, 255, 128), (255, 255, 255)), ((255, 255, 128), (255, 255, 128)), ((255, 255, 128), (0, 0, 0)), ((255, 255, 128), (255, 0, 0)), ((255, 255, 128), (192, 192, 192))}]

Zestaw użytych pikseli:
(((192, 192, 192), (255, 255, 255), (255, 0, 0), (255, 255, 128), (0, 0, 0)), (255, 255, 128), ((255, 255, 255), (255, 255, 128), (0, 0, 0), (255, 0, 0), (192, 192, 192)))

Zestaw użytych kolorów
Ilość kolorów: 11

Asocjacja ta rozpoznaje następujące elementy obrazu (ramka oznacza mało widoczne elementy):



list_asoc[7] = 4 , 4
[{((255, 255, 255), (128, 0, 128)), ((128, 128, 64), (128, 0, 128)), ((0, 0, 0), (128, 0, 128)), ((128, 0, 128), (128, 0, 128))}, {((128, 0, 128), (128, 128, 64)), ((128, 0, 128), (0, 0, 0)), ((128, 0, 128), (255, 255, 255)), ((128, 0, 128), (128, 0, 128))}]

Zestaw użytych pikseli:
(((255, 255, 255), (128, 128, 64), (0, 0, 0), (128, 0, 128)), (128, 0, 128), ((128, 128, 64), (0, 0, 0), (255, 255, 255), (128, 0, 128)))

Zestaw użytych kolorów
Ilość kolorów: 9

Asocjacja ta rozpoznaje następujące elementy obrazu:

list_asoc[8] = 7 , 8
[{((255, 255, 255), (128, 128, 64)), ((255, 255, 0), (128, 128, 64)), ((255, 0, 255), (128, 128, 64)), ((0, 0, 0), (128, 128, 64)), ((128, 0, 0), (128, 128, 64)), ((128, 128, 64), (128, 128, 64)), ((128, 0, 128), (128, 128, 64))}, {((128, 128, 64), (255, 255, 255)), ((128, 128, 64), (0, 0, 0)), ((128, 128, 64), (128, 0, 128)), ((128, 128, 64), (76, 61, 175)), ((128, 128, 64), (255, 255, 0)), ((128, 128, 64), (255, 0, 255)), ((128, 128, 64), (128, 128, 64)), ((128, 128, 64), (128, 0, 0))}]

Zestaw użytych pikseli:
(((255, 255, 255), (255, 255, 0), (255, 0, 255), (0, 0, 0), (128, 0, 0), (128, 128, 64), (128, 0, 128)), (128, 128, 64), ((255, 255, 255), (0, 0, 0), (128, 0, 128), (76, 61, 175), (255, 255, 0), (255, 0, 255), (128, 128, 64), (128, 0, 0)))

Zestaw użytych kolorów
Ilość kolorów: 16

Asocjacja ta rozpoznaje następujące elementy obrazu:

list_asoc[9] = 8 , 9
[{((255, 255, 255), (255, 0, 0)), ((128, 255, 255), (255, 0, 0)), ((255, 255, 0), (255, 0, 0)), ((255, 0, 0), (255, 0, 0)), ((76, 61, 175), (255, 0, 0)), ((255, 255, 128), (255, 0, 0)), ((0, 0, 0), (255, 0, 0)), ((128, 128, 128), (255, 0, 0))}, {((255, 0, 0), (0, 0, 0)), ((255, 0, 0), (128, 255, 255)), ((255, 0, 0), (76, 61, 175)), ((255, 0, 0), (128, 128, 128)), ((255, 0, 0), (255, 255, 255)), ((255, 0, 0), (255, 255, 128)), ((255, 0, 0), (255, 0, 0)), ((255, 0, 0), (192, 192, 192)), ((255, 0, 0), (255, 255, 0))}]

Zestaw użytych pikseli:
(((255, 255, 255), (128, 255, 255), (255, 255, 0), (255, 0, 0), (76, 61, 175), (255, 255, 128), (0, 0, 0), (128, 128, 128)), (255, 0, 0), ((0, 0, 0), (128, 255, 255), (76, 61, 175), (128, 128, 128), (255, 255, 255), (255, 255, 128), (255, 0, 0), (192, 192, 192), (255, 255, 0)))

Zestaw użytych kolorów
Ilość kolorów: 18

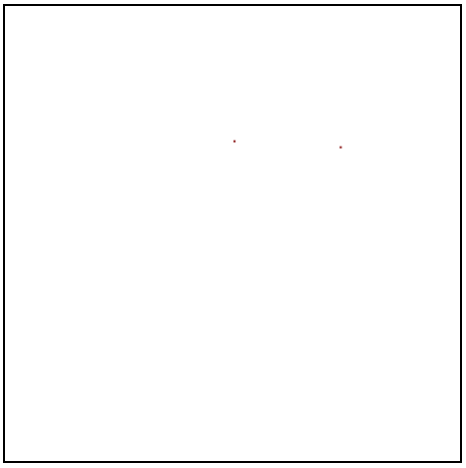
Asocjacja ta rozpoznaje następujące elementy obrazu:

list_asoc[10] = 1 , 1
[{((128, 128, 64), (128, 0, 0))}, {((128, 0, 0), (128, 0, 0))}]

Zestaw użytych pikseli:
(((128, 128, 64),), (128, 0, 0), ((128, 0, 0),))

Zestaw użytych kolorów
Ilość kolorów: 3

Asocjacja ta rozpoznaje następujące elementy obrazu (ramka oznacza mało widoczne elementy):

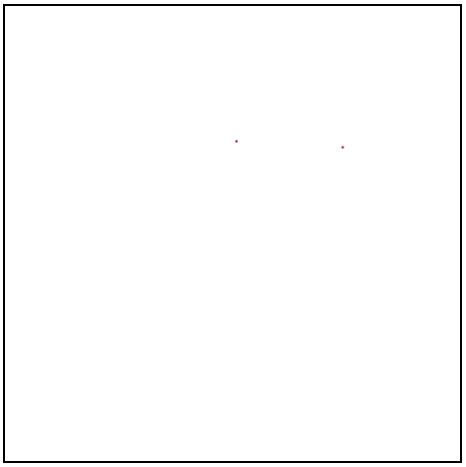


list_asoc[11] = 1 , 1
[{((128, 0, 0), (128, 0, 0))}, {((128, 0, 0), (128, 128, 64))}]

Zestaw użytych pikseli:
(((128, 0, 0),), (128, 0, 0), ((128, 128, 64),))

Zestaw użytych kolorów
Ilość kolorów: 3

Asocjacja ta rozpoznaje następujące elementy obrazu (ramka oznacza mało widoczne elementy):



list_asoc[12] = 5 , 5
[{((0, 0, 0), (255, 0, 255)), ((255, 0, 255), (255, 0, 255)), ((255, 255, 255), (255, 0, 255)), ((128, 128, 64), (255, 0, 255)), ((76, 61, 175), (255, 0, 255))}, {((255, 0, 255), (255, 0, 255)), ((255, 0, 255), (128, 128, 64)), ((255, 0, 255), (255, 255, 255)), ((255, 0, 255), (0, 0, 0)), ((255, 0, 255), (76, 61, 175))}]

Zestaw użytych pikseli:
(((0, 0, 0), (255, 0, 255), (255, 255, 255), (128, 128, 64), (76, 61, 175)), (255, 0, 255), ((255, 0, 255), (128, 128, 64), (255, 255, 255), (0, 0, 0), (76, 61, 175)))

Zestaw użytych kolorów
Ilość kolorów: 11

Asocjacja ta rozpoznaje następujące elementy obrazu:

list_asoc[13] = 4 , 5
[{((128, 128, 128), (128, 128, 128)), ((128, 255, 255), (128, 128, 128)), ((255, 0, 0), (128, 128, 128)), ((0, 0, 0), (128, 128, 128))}, {((128, 128, 128), (0, 0, 0)), ((128, 128, 128), (255, 255, 255)), ((128, 128, 128), (128, 128, 128)), ((128, 128, 128), (128, 255, 255)), ((128, 128, 128), (255, 0, 0))}]

Zestaw użytych pikseli:
(((128, 128, 128), (128, 255, 255), (255, 0, 0), (0, 0, 0)), (128, 128, 128), ((0, 0, 0), (255, 255, 255), (128, 128, 128), (128, 255, 255), (255, 0, 0)))

Zestaw użytych kolorów
Ilość kolorów: 10

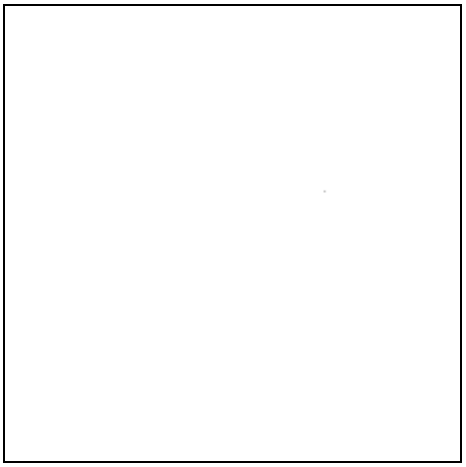
Asocjacja ta rozpoznaje następujące elementy obrazu:

list_asoc[14] = 1 , 1
[{((255, 255, 128), (192, 192, 192))}, {((192, 192, 192), (255, 255, 255))}]

Zestaw użytych pikseli:
(((255, 255, 128),), (192, 192, 192), ((255, 255, 255),))

Zestaw użytych kolorów
Ilość kolorów: 3

Asocjacja ta rozpoznaje następujące elementy obrazu (ramka oznacza mało widoczne elementy):

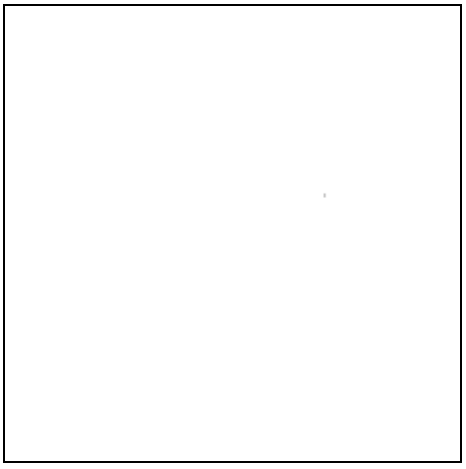


list_asoc[15] = 2 , 1
[{((255, 0, 0), (192, 192, 192)), ((0, 0, 0), (192, 192, 192))}, {((192, 192, 192), (255, 255, 128))}]

Zestaw użytych pikseli:
(((255, 0, 0), (0, 0, 0)), (192, 192, 192), ((255, 255, 128),))

Zestaw użytych kolorów
Ilość kolorów: 4

Asocjacja ta rozpoznaje następujące elementy obrazu (ramka oznacza mało widoczne elementy):



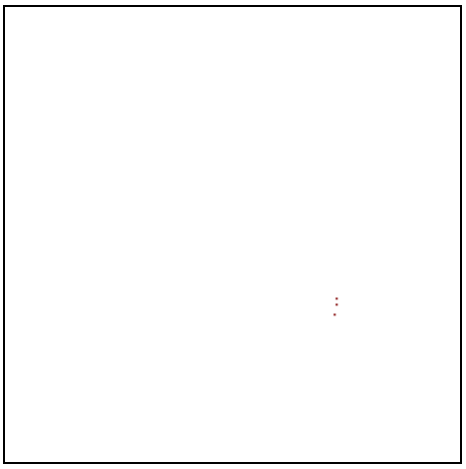
list_asoc[16] = 1 , 1
[{((255, 255, 0), (128, 0, 0))}, {((128, 0, 0), (255, 255, 255))}]

Zestaw użytych pikseli:

(((255, 255, 0),), (128, 0, 0), ((255, 255, 255),))

Zestaw użytych kolorów
Ilość kolorów: 3

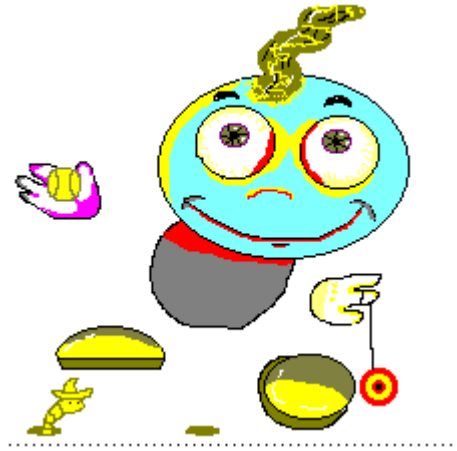
Asocjacja ta rozpoznaje następujące elementy obrazu (ramka oznacza mało widoczne elementy):



Asocjacje o indeksach: 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16 rozpoznają obraz:

Asocjacje o indeksach 0,1,2,3,4,5,6,7,8,9,12,13,15) rozpoznają obraz:

Asocjacje o indeksach: 1,2,3,4,5,6,7,8,9,12,13,15 rozpoznają następujące elementy obrazu:



Asocjacje o indeksach: 1,2,3,4,5,6,8,9,12,13,15 rozpoznają następujące elementy obrazu:

Asocjacje o indeksach: 1,2,3,4,5,6,8,9,13,15 rozpoznają następujące elementy obrazu:

Asocjacje o indeksach: 1,2,3,4,6,8,9,13,15 rozpoznają następujące elementy obrazu:

Asocjacje o indeksach: 2,3,4,6,8,9,13,15 rozpoznają następujące elementy obrazu:

Asocjacje o indeksach: 2,3,4,8,9,13,15 rozpoznają następujące elementy obrazu:

Asocjacje o indeksach: 2,3,4,9,13,15 rozpoznają następujące elementy obrazu:

Asocjacje o indeksach: 2,3,4,9,15 rozpoznają następujące elementy obrazu:

Asocjacje o indeksach: 2,3,4,15 rozpoznają następujące elementy obrazu:

Asocjacje o indeksach: 2,4,15 rozpoznają następujące elementy obrazu:

Asocjacje o indeksach: 4, 15 rozpoznają następujące elementy obrazu:

Generacja kontekstowej asocjacji jednostkowej

Generacja kontekstowej asocjacji jednostkowej polega na wykonywaniu następującego kodu programu:

```
for y in range(0, height-1):
        for x in range(0, width-1):

                if x == 0:
                        c = False

                e[1] = (pixels[x,y], pixels[x+1,y])

                if c:
                        list_asoc = consolid(list_asoc, e)
                e[0] = e[1]
                c = True
```

Kod generuje strukturę:

```
e = [(pixels[x,y], pixels[x+1,y]), (pixels[x+1,y], pixels[x+2,y])]
```

gdzie:

```
e - [(lewostronny kontekst, piksel), (piksel, prawostronny kontekst)]
e[0] = (pixels[x,y], pixels[x+1,y])
e[1] = (pixels[x+1,y], pixels[x+2,y])
```

Po zakończeniu ostatniego wiersza i rozpoczęciu nowego, aby się nie zdarzyło, że zostanie wygenerowana błędna struktura, taka jak:

```
e = [(pixels[x,y], pixels[x+1,y]), (pixels[0,y], pixels[1,y])]
```

do eliminacji tego błędu służy instrukcja:

```
if x == 0: c = False
```

która oznacza, że na początku każdego nowego wiersza nieukończoną asocjację jednostkową z ostatniego wiersza (kiedy: `c = True`) należy odrzucić i rozpocząć tworzenie od nowa nowej asocjacji jednostkowej.

Konsolidacja asocjacji jednostkowych

Konsolidację asocjacji jednostkowych przeprowadzamy łącząc kolejną asocjację jednostkową z listą skonsolidowanych i rozłącznych asocjacji złożonych.

W celu zoptymalizowania i uproszczenia mechanizmu konsolidacji przeanalizowano dotychczas stosowany algorytm i mechanizm konsolidacji.

Zauważono, że:

Przy konsolidacji listy rozłącznych asocjacji z asocjacją jednostkową zwrócono uwagę na to, że asocjacja jednostkowa zawierająca jeden element w poprzedniku, którego wartość może się powtórzyć w poprzedniku co najwyżej jednej asocjacji na liście rozłącznych asocjacji, oraz zawierająca jeden element w następniku, którego wartość może się powtórzyć w następniku co najwyżej jednej asocjacji na liście rozłącznych asocjacji.

Z tego względu należy rozpatrzyć następujące przypadki:

- asocjacja jednostkowa jest rozłączna z asocjacjami listy - wówczas należy ją dodać do listy asocjacji,
- asocjacja jednostkowa jest nierozłączna jednostronnie (po stronie poprzednika lub następnika) lub dwustronnie z jedną z asocjacji listy - wówczas należy ją połączyć z tą asocjacją,
- asocjacja jednostkowa jest nierozłączna dwustronnie z jedną z asocjacji listy - wówczas oznacza to, że połączenie już nastąpiło i nie należy podejmować żadnej akcji,
- asocjacja jednostkowa jest nierozłączna dwustronnie z dwoma różnymi asocjacjami listy - wówczas należy te dwie asocjacje i asocjację jednostkową połączyć w jedną asocjację,

Ogólnie dwie asocjacje a i b są nierozłączne jeżeli spełnione jest wyrażenie:

```
if a[0] & b[0] != set() or a[1] & b[1] != set()
```

Połączenie listy asocjacji rozłącznych list_asoc z asocjacją jednostkową e przebiega w taki sposób, aby po połączeniu asocjacje na liście były nadal rozłączne. Przebiega to według następującej reguły:

```
def consolid(list_asoc, e):
    asoc_p, asoc_n = [], []
    for asoc in list_asoc:

        if e[0] in asoc[0]:
            asoc_p = asoc

        if e[1] in asoc[1]:
            asoc_n = asoc

        if asoc_p != [] and asoc_n != []:
            if asoc_p != asoc_n:

                asoc_n[0] |= asoc_p[0]
```

```
                asoc_n[1] |= asoc_p[1]
                list_asoc.remove(asoc_p)

            break
    else:
        if asoc_p == [] and asoc_n == []:
            list_asoc.append([{e[0]},{e[1]}])
        elif asoc_p != []:
            asoc_p[1].add(e[1])
        elif asoc_n != []:
            asoc_n[0].add(e[0])

    return list_asoc
```

Analizując powyższą regułę zauważamy, że przeglądanie kolejnych asocjacji asoc z listy list_asoc przebiega w pętli: for asoc in list_asoc. Podczas przeglądania tych asocjacji sprawdzanie jest, czy poprzednik asocjacji ma produkcję wspólną z poprzednikiem asocjacji jednostkowej:

```
if e[0] in asoc[0]:
        asoc_p = asoc
```

oraz czy następnik tej asocjacji ma produkcję wspólną z następnikiem asocjacji jednostkowej:

```
if e[1] in asoc[1]:
        asoc_n = asoc
```

W momencie zaistnienia powyższych sytuacji zmienne asoc_p i asoc_n zostają ustawione na te asocjacje w przeciwnym wypadku mają wartość []. Należy pamiętać, że dla listy asocjacji rozłącznych, może wystąpić najwyżej jedna produkcja wspólna po stronie poprzednika i jedna produkcja wspólna po stronie następnika. W chwili, kiedy zmienne asoc_p i asoc_n wskazują asocjacje, w których wystąpiła produkcja wspólna dla poprzednika i następnika dalsze przeglądanie asocjacji zostaje wstrzymane break. Dalsze szukanie jest zbędne ponieważ tylko jedna asocjacja może być nierozłączna po stronie poprzednika i tylko jedna asocjacja może być nierozłączna po stronie następnika z asocjacją jednostkową. Jeżeli asocjacje są różne „if asoc_p != asoc_n" następuje połączenie tych asocjacji w jedną asocjację i usunięcie z listy zbędnej asocjacji:

```
asoc_n[0] |= asoc_p[0]
asoc_n[1] |= asoc_p[1]
list_asoc.remove(asoc_p)
```

Natomiast, kiedy zmienne wskazują tą samą asocjację „asoc_p == asoc_n", wskazując nierozłączność dwustronną, to znaczy, że asocjacja z listy zawiera już elementy asocjacji jednostkowej i nie należy podejmować żadnej akcji. Należy pamiętać, że asocjacja wskazana przez asoc_p zawiera już produkcję z poprzednika asocjacji jednostkowej e, a asocjacja wskazana przez asoc_n zawiera już produkcję z następnika asocjacji jednostkowej e. Można też zastosować alternatywny sposób połączenia dwóch asocjacji:

```
asoc_p[0] |= asoc_n[0]
asoc_p[1] |= asoc_n[1]
list_asoc.remove(asoc_n)
```

który zmienia jedynie położenie asocjacji na liście nie wpływając na ich postać. Jeżeli instrukcja break nie wystąpi mamy możliwość obsłużenia bloku else:

```
else:
    if asoc_p == [] and asoc_n == []:
        list_asoc.append([{e[0]},{e[1]}])
    elif asoc_p != []:
        asoc_p[1].add(e[1])
    elif asoc_n != []:
        asoc_n[0].add(e[0])
```

Są tu przypadki, kiedy wszystkie asocjacje na liście są rozłączne dwustronnie z asocjacją jednostkową - wtedy asocjacja jednostkowa zostaje dodana do listy asocjacji oraz kiedy tylko asocjacje po stronie następnika lub tylko asocjacje po stronie poprzednika są rozłączne z asocjacją jednostkową (rozłączność jednostronna) - wówczas asocjacja jednostkowa zostaje połączona z tą asocjacją z listy, która jest nierozłączna z asocjacją jednostkową, uzupełniając produkcję z asocjacji jednostkowej po stronie rozłączności.

Należy zauważyć, że w poprzednich pracach, w treści funkcji asocjacji consolid(list_asoc, e) była używana mniej efektywna i mniej zrozumiała definicja konsolidacji. W tej pracy opracowano bardziej czytelną, efektywną i zrozumiałą definicję dającą możliwość dalszej jej modyfikacji na przyszłe zastosowania.

Dodatkowe zalety Python-a

Podczas projektowania zauważono, że w interpreterze Pythona można zatrzymywać bieg kodu przy pomocy klawiszy Crtl-C, a także zmieniać działanie wybranych funkcji poprzez wprowadzanie na bieżąco zmienionej ich treści i sprawdzać działanie programu lub jego części z tak dokonanymi zmianami bez opuszczania interpretera, przy zachowaniu bieżących ustawionych przez program wartości zmiennych globalnych.

Kod źródłowy:

Do badań kontekstowych asocjacji regularnych użyto następującego kodu źródłowego:

```
# Kontekstowe asocjacje regularne (Pyton 3.9.7)
# Waldemar Wietrzykowski, 27.07.2022
# ver 2

def consolid(list_asoc, e):
   asoc_p, asoc_n = [], []
   for asoc in list_asoc:

      if e[0] in asoc[0]:
         asoc_p = asoc
```

```
        if e[1] in asoc[1]:
            asoc_n = asoc

        if asoc_p != [] and asoc_n != []:
            if asoc_p != asoc_n:
                asoc_n[0] |= asoc_p[0]
                asoc_n[1] |= asoc_p[1]
                list_asoc.remove(asoc_p)

                """
                # alternatywnie
                #asoc_p[0] |= asoc_n[0]
                #asoc_p[1] |= asoc_n[1]
                #list_asoc.remove(asoc_n)
                """
            break
    else:
        if asoc_p == [] and asoc_n == []:
            list_asoc.append([{e[0]},{e[1]}])
        elif asoc_p != []:
            asoc_p[1].add(e[1])
        elif asoc_n != []:
            asoc_n[0].add(e[0])

    return list_asoc

def pixels_asoc(ind, list_asoc):
    pixels_p, pixel, pixels_n = (), (), ()

    for prod in list_asoc[ind][0]:
        pixels_p += prod[0],

    for prod in list_asoc[ind][0]:
        pixel = prod[1]
        break

    for prod in list_asoc[ind][1]:
        pixels_n += prod[1],

    return pixels_p, pixel, pixels_n

def pixel_asoc(x, pix, dl_color):
    ind = x // dl_color
    r = x % dl_color
    bialy = (255,255,255)
    if ind < len(pix[0]):
        return pix[0][ind]
    elif ind == len(pix[0]):
        if r == 0:
            return bialy
        else:
            return pix[1]
    elif ind > len(pix[0]) and ind < len(pix[0]) + len(pix[2]) + 1:
        if r == 0 and ind == len(pix[0]) + 1:
            return bialy
        else:
            return pix[2][ind - len(pix[0]) - 1]
```

```
    else:
        return bialy

def color_asoc(x, colors, dl_color):
    ind = x // dl_color
    bialy = (255,255,255)
    if ind < len(colors):
        return colors[ind]
    else:
        return bialy

def find_asoc(list_asoc, ind, e):
    if e[0] in list_asoc[ind][0] and e[1] in list_asoc[ind][1]:
        return True
    else:
        return False

def link_asoc(list_asoc, inds, e):
    for ind in inds:
        if find_asoc(list_asoc, ind, e):
            return True
    else:
        return False

def test(list_asoc):
    dl = len(list_asoc)
    for i in range(0,dl):
        for j in range(i+1,dl):
            if list_asoc[i][0] & list_asoc[j][0] != set():
                return False
            if list_asoc[i][1] & list_asoc[j][1] != set():
                return False
    return True

def not_context_asoc(e):
    if e[0][1] == e[1][0]:
        return False
    else:
        return True

def get_colors(list_asoc):
    colors = []
    for asoc in list_asoc:
        for prod in asoc[0]:
          colors += prod[1],
          break
    return colors

#################### koniec funkcji consolid #############

def show_asoc(list_asoc, opcja = 0):
    print('len(list_asoc) = ',len(list_asoc))
    print()
    i = 0
    for asoc in list_asoc:
        print(f'len(list_asoc[{i}][0],[1]) = ',len(list_asoc[i][0]),',
',len(list_asoc[i][1]))
```

```
       if opcja == 0:
          print(list_asoc[i])
          print()
       i += 1

def concolid_asoc(list_asoc, opcja = 0):
    img = Image.open(path+file).convert('RGB')
    width, height = img.size
    pixels = img.load()
    e = [0, 0]

    if opcja == 0:
       c = False
       for y in range(0, height-1):

          for x in range(0, width-1):

             if x == 0:
                c = False

             e[1] = (pixels[x,y], pixels[x+1,y])

             if c:
                if not_context_asoc(e):
                   print('Error! Not_context_asoc')
                   print('x,y = ',x,y)
                   print('e =',e)

                list_asoc = consolid(list_asoc, e)
             e[0] = e[1]
             c = True

    elif opcja == 1:
       c = False
       for x in range(0, width-1):
          for y in range(0, height-1):

             if y == 0:
                c = False

             e[1] = (pixels[x,y], pixels[x+1,y])

             if c:
                if not_context_asoc(e):
                   print('Error! Not_context_asoc')
                   print('x,y = ',x,y)
                   print('e =',e)

                list_asoc = consolid(list_asoc, e)
             e[0] = e[1]
             c = True

    return list_asoc

def image_bmp_asoc(inds, list_asoc):
    img = Image.open(path+file).convert('RGB')
    width, height = img.size
```

```
    print('width,height: ',width, height);
    pixels = img.load()
    e = [0, 0]
    white = (255,255,255)

    c = False
    for y in range(0, height-1): # range(początek, stop, krok)

        for x in range(0, width-1):
            e[1] = (pixels[x,y], pixels[x+1,y])

            if c:
                # if find_asoc(list_asoc, ind, e):

                if link_asoc(list_asoc, inds, e):
                    pass
                else:
                    pixels[x,y] = white
            else:
                pixels[0,0] = white

            e[0] = e[1]
            c = True
        else:
            pixels[x+1,y] = white
    else:
        for x in range(0, width):
            pixels[x,y+1] = white

    img.show()
    img.save(path+'/_hopek.bmp')

    return

def image_bmp_pixel(ind, list_asoc):
    img = Image.open(path+file).convert('RGB')
    width, height = img.size
    pixels = img.load()
    pix = pixels_asoc(ind, list_asoc)
    count_color = len(pix[0])+len(pix[2])+1
    dl_color = width // count_color
    height_color = 25
    print(pix)
    print('Ilość kolorów: ',count_color)

    for y in range(0, height-1 ):
        # for x in range(0, width-1):
        for x in range(0, width-1):

            pixels[x,y] = pixel_asoc(x, pix, dl_color)

    img2 = img.crop((0, 0, dl_color * count_color, height_color))

    img2.show()
    img2.save(path+'/_color.bmp')

    return
```

```
def color_bmp(list_asoc):
    img = Image.open(path+file).convert('RGB')
    width, height = img.size
    pixels = img.load()
    colors = get_colors(list_asoc)
    count_color = len(colors)
    print(colors)
    dl_color = width // count_color
    height_color = 25
    print('Ilość kolorów podstawowych: ',count_color)

    for y in range(0, height-1 ):

        for x in range(0, width-1):

            pixels[x,y] = color_asoc(x, colors, dl_color)

    img2 = img.crop((0, 0, dl_color * count_color, height_color))

    img2.show()
    img2.save(path+'/_color.bmp')

    return


path = 'C://Dane/DIL/SCIENCE/Maszyna Wietrzykowskiego/PROJEKT świadomej
maszyny/ASOCJACJE REGULARNE KONTEKSTOWE/PROGRAM'
file = '/hopek.bmp'

from PIL import Image

print('Konsolidacja gradientów w pliku: ',file[1:])
list_asoc = []
list_asoc = concolid_asoc(list_asoc, 0)
show_asoc(list_asoc, 0)
print('Asocjacje ortogonalne: ',test(list_asoc))
color_bmp(list_asoc)
image_bmp_asoc((0,), list_asoc)
image_bmp_pixel(0, list_asoc)

#############################################################
#                           dodatek                         #
#############################################################
```

Dodatek 1

Waldemar Wietrzykowski

Instytut Inteligencji Stosowanej, 8 sierpień 2022

Wstęp

W oparciu o pracę „Kontekstowe asocjacje regularne” przeprowadzono analizę znaczenia kontekstowych asocjacji regularnych.

Rozmiar kontekstowych asocjacji regularnych

Asocjacje kontekstowe utworzone na podstawie obrazu o rozmiarze 51984 pikseli składają się z 17 asocjacji złożonych i zajmują w sumie zaledwie 180 pikseli w parach po: (7, 6), (8, 9), (6, 6), (12, 9), (12, 13), (6, 6), (5, 5), (4, 4), (7, 8), (8, 9), (1, 1), (1, 1), (5, 5), (4, 5), (1, 1), (2, 1), (1, 1) pikseli dla poprzedników i następników dla każdej asocjacji, w sumie 90 pikseli dla poprzedników i 90 pikseli dla następników wszystkich asocjacji. Na asocjacje potrzeba zatem 0,35 % wszystkich pikseli jakie zajmuje obraz pomimo tego, że asocjacje odnoszą się do całej zawartości obrazu.

Co zawierają kontekstowe asocjacje

Jak wynika z definicji kontekstowych asocjacji zawierają one informacje o regularności kontekstów, w jakich występują piksele w obrazie.

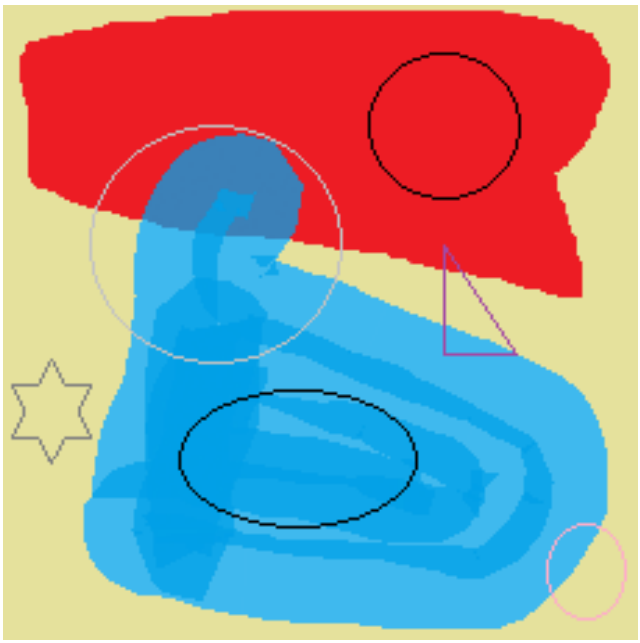
Konsolidacja kontekstowych asocjacji

Jak wynika z prawa kontekstowych asocjacji regularnych, podanego w pracy „Kontekstowe asocjacje regularne”:

- wystąpienie określonego koloru (piksela) z lewostronnym i prawostronnym kontekstem (pikselem) powoduje, że staje się on zaczynem konsolidowania z utworzonej wówczas asocjacji złożonej dotyczący tego koloru,
- nowy kontekst lewostronny zostaje dodany do asocjacji złożonej, kiedy współistniejący prawy kontekst koloru istnieje już zapisany w asocjacji złożonej,
- nowy kontekst prawostronny zostaje dodany do asocjacji złożonej, kiedy współistniejący lewy kontekst koloru istnieje już zapisany w asocjacji złożonej,
- jeżeli współistniejące konteksty koloru są już zapisane w asocjacji złożonej potwierdza to tylko, że asocjacja złożona taki przypadek już odnotowała,
- jeżeli kontekst lewy współistniejącego koloru należy do jednej asocjacji złożonej a kontekst prawy należy do innej asocjacji złożonej to dwie asocjacji złożone zostają połączone.

Powyższe zasady powodują, że mogą istnieć dwie lub więcej asocjacji złożonych tego samego koloru jeżeli zbiory odnotowanych w nich konteksty prawe i lewe względem tych asocjacji są rozłączne.

Przykład:



Ilość asocjacji: 30
Ilość produkcji: w poprzednikach: 156 w następnikach: 156 w sumie: 312

Zestaw pikseli podstawowych:
[(229, 225, 156), (237, 28, 36), (240, 69, 76), (240, 69, 76), (0, 0, 0), (195, 195, 195), (59, 129, 184), (59, 128, 183), (15, 153, 219), (4, 157, 228), (63, 185, 238), (63, 182, 235), (64, 185, 238), (16, 167, 233), (63, 183, 236), (163, 73, 164), (4, 161, 231), (5, 162, 231), (2, 162, 230), (1, 160, 231), (1, 160, 230), (65, 186, 238), (2, 161, 230), (1, 161, 230), (127, 127, 127), (5, 163, 230), (17, 167, 233), (255, 174, 201), (2, 162, 230), (65, 186, 238)]

Zestaw użytych kolorów podstawowych:
Ilość kolorów podstawowych: 30

Lista asocjacji:

list_asoc[0] = 9 , 10
[{((63, 185, 238), (229, 225, 156)), ((163, 73, 164), (229, 225, 156)), ((127, 127, 127), (229, 225, 156)), ((195, 195, 195), (229, 225, 156)), ((229, 225, 156), (229, 225, 156)), ((240, 69, 76), (229, 225, 156)), ((255, 174, 201), (229, 225, 156)), ((64, 185, 238), (229, 225, 156)), ((237, 28, 36), (229, 225, 156))}, {((229, 225, 156), (195, 195, 195)), ((229, 225, 156), (237, 28, 36)), ((229, 225, 156), (240, 69, 76)), ((229, 225, 156), (229, 225, 156)), ((229, 225, 156), (64, 185, 238)), ((229, 225, 156), (127, 127, 127)), ((229, 225, 156), (163, 73, 164)), ((229, 225, 156), (63, 185, 238)), ((229, 225, 156), (255, 174, 201)), ((229, 225, 156), (16, 167, 233))}]

list_asoc[1] = 8 , 8

[{((229, 225, 156), (237, 28, 36)), ((240, 69, 76), (237, 28, 36)), ((59, 128, 183), (237, 28, 36)), ((0, 0, 0), (237, 28, 36)), ((237, 28, 36), (237, 28, 36)), ((59, 129, 184), (237, 28, 36)), ((163, 73, 164), (237, 28, 36)), ((195, 195, 195), (237, 28, 36))}, {((237, 28, 36), (163, 73, 164)), ((237, 28, 36), (0, 0, 0)), ((237, 28, 36), (59, 128, 183)), ((237, 28, 36), (195, 195, 195)), ((237, 28, 36), (237, 28, 36)), ((237, 28, 36), (240, 69, 76)), ((237, 28, 36), (59, 129, 184)), ((237, 28, 36), (229, 225, 156))}]

list_asoc[2] = 1 , 1
[{((237, 28, 36), (240, 69, 76))}, {((240, 69, 76), (229, 225, 156))}]

list_asoc[3] = 1 , 1
[{((229, 225, 156), (240, 69, 76))}, {((240, 69, 76), (237, 28, 36))}]

list_asoc[4] = 10 , 9
[{((237, 28, 36), (0, 0, 0)), ((4, 161, 231), (0, 0, 0)), ((2, 162, 230), (0, 0, 0)), ((2, 161, 230), (0, 0, 0)), ((5, 163, 230), (0, 0, 0)), ((1, 160, 231), (0, 0, 0)), ((63, 185, 238), (0, 0, 0)), ((0, 0, 0), (0, 0, 0)), ((1, 160, 230), (0, 0, 0)), ((16, 167, 233), (0, 0, 0))}, {((0, 0, 0), (1, 160, 231)), ((0, 0, 0), (2, 162, 230)), ((0, 0, 0), (5, 163, 230)), ((0, 0, 0), (1, 160, 230)), ((0, 0, 0), (63, 185, 238)), ((0, 0, 0), (0, 0, 0)), ((0, 0, 0), (237, 28, 36)), ((0, 0, 0), (16, 167, 233)), ((0, 0, 0), (4, 161, 231))}]

list_asoc[5] = 8 , 10
[{((195, 195, 195), (195, 195, 195)), ((229, 225, 156), (195, 195, 195)), ((1, 160, 231), (195, 195, 195)), ((237, 28, 36), (195, 195, 195)), ((4, 161, 231), (195, 195, 195)), ((63, 185, 238), (195, 195, 195)), ((59, 129, 184), (195, 195, 195)), ((16, 167, 233), (195, 195, 195))}, {((195, 195, 195), (195, 195, 195)), ((195, 195, 195), (1, 160, 231)), ((195, 195, 195), (229, 225, 156)), ((195, 195, 195), (5, 163, 230)), ((195, 195, 195), (64, 185, 238)), ((195, 195, 195), (1, 160, 230)), ((195, 195, 195), (63, 185, 238)), ((195, 195, 195), (237, 28, 36)), ((195, 195, 195), (16, 167, 233)), ((195, 195, 195), (4, 161, 231))}]

list_asoc[6] = 7 , 5
[{((59, 129, 184), (59, 129, 184)), ((63, 182, 235), (59, 129, 184)), ((59, 128, 183), (59, 129, 184)), ((15, 153, 219), (59, 129, 184)), ((63, 183, 236), (59, 129, 184)), ((237, 28, 36), (59, 129, 184)), ((63, 185, 238), (59, 129, 184))}, {((59, 129, 184), (59, 129, 184)), ((59, 129, 184), (15, 153, 219)), ((59, 129, 184), (59, 128, 183)), ((59, 129, 184), (237, 28, 36)), ((59, 129, 184), (195, 195, 195))}]

list_asoc[7] = 4 , 4
[{((59, 129, 184), (59, 128, 183)), ((59, 128, 183), (59, 128, 183)), ((15, 153, 219), (59, 128, 183)), ((237, 28, 36), (59, 128, 183))}, {((59, 128, 183), (59, 129, 184)), ((59, 128, 183), (59, 128, 183)), ((59, 128, 183), (237, 28, 36)), ((59, 128, 183), (15, 153, 219))}]

list_asoc[8] = 6 , 4
[{((15, 153, 219), (15, 153, 219)), ((4, 157, 228), (15, 153, 219)), ((63, 185, 238), (15, 153, 219)), ((59, 129, 184), (15, 153, 219)), ((16, 167, 233), (15, 153, 219)), ((59, 128, 183), (15, 153, 219))}, {((15, 153, 219), (15, 153, 219)), ((15, 153, 219), (59, 129, 184)), ((15, 153, 219), (59, 128, 183)), ((15, 153, 219), (4, 157, 228))}]

list_asoc[9] = 2 , 2
[{((4, 157, 228), (4, 157, 228)), ((15, 153, 219), (4, 157, 228))}, {((4, 157, 228), (15, 153, 219)), ((4, 157, 228), (4, 157, 228))}]

list_asoc[10] = 12 , 15
[{((64, 185, 238), (63, 185, 238)), ((17, 167, 233), (63, 185, 238)), ((255, 174, 201), (63, 185, 238)), ((4, 161, 231), (63, 185, 238)), ((63, 185, 238), (63, 185, 238)), ((163, 73, 164), (63, 185, 238)), ((5, 163, 230), (63, 185, 238)), ((0, 0, 0), (63, 185, 238)), ((16, 167, 233), (63, 185, 238)), ((65, 186, 238), (63, 185, 238)), ((195, 195, 195), (63, 185, 238)), ((229, 225, 156), (63, 185, 238))}, {((63, 185, 238), (229, 225, 156)), ((63, 185, 238), (63, 183, 236)), ((63, 185, 238), (65, 186, 238)), ((63, 185, 238), (255, 174, 201)), ((63, 185, 238), (64, 185, 238)), ((63, 185, 238), (15, 153, 219)), ((63, 185, 238), (163, 73, 164)), ((63, 185, 238), (63, 182, 235)), ((63, 185, 238), (63, 185, 238)), ((63, 185, 238), (16, 167, 233)), ((63, 185, 238), (0, 0, 0)), ((63, 185, 238), (4, 161, 231)), ((63, 185, 238), (195, 195, 195)), ((63, 185, 238), (17, 167, 233)), ((63, 185, 238), (59, 129, 184))}]

list_asoc[11] = 1 , 1
[{((63, 185, 238), (63, 182, 235))}, {((63, 182, 235), (59, 129, 184))}]

list_asoc[12] = 7 , 6
[{((64, 185, 238), (64, 185, 238)), ((255, 174, 201), (64, 185, 238)), ((63, 185, 238), (64, 185, 238)), ((16, 167, 233), (64, 185, 238)), ((65, 186, 238), (64, 185, 238)), ((195, 195, 195), (64, 185, 238)), ((229, 225, 156), (64, 185, 238))}, {((64, 185, 238), (64, 185, 238)), ((64, 185, 238), (63, 185, 238)), ((64, 185, 238), (255, 174, 201)), ((64, 185, 238), (16, 167, 233)), ((64, 185, 238), (229, 225, 156)), ((64, 185, 238), (65, 186, 238))}]

list_asoc[13] = 11 , 12
[{((0, 0, 0), (16, 167, 233)), ((17, 167, 233), (16, 167, 233)), ((4, 161, 231), (16, 167, 233)), ((2, 162, 230), (16, 167, 233)), ((64, 185, 238), (16, 167, 233)), ((63, 185, 238), (16, 167, 233)), ((1, 160, 231), (16, 167, 233)), ((5, 163, 230), (16, 167, 233)), ((16, 167, 233), (16, 167, 233)), ((195, 195, 195), (16, 167, 233)), ((229, 225, 156), (16, 167, 233))}, {((16, 167, 233), (17, 167, 233)), ((16, 167, 233), (0, 0, 0)), ((16, 167, 233), (4, 161, 231)), ((16, 167, 233), (1, 160, 231)), ((16, 167, 233), (2, 162, 230)), ((16, 167, 233), (2, 161, 230)), ((16, 167, 233), (5, 163, 230)), ((16, 167, 233), (64, 185, 238)), ((16, 167, 233), (15, 153, 219)), ((16, 167, 233), (63, 185, 238)), ((16, 167, 233), (16, 167, 233)), ((16, 167, 233), (195, 195, 195))}]

list_asoc[14] = 1 , 1
[{((63, 185, 238), (63, 183, 236))}, {((63, 183, 236), (59, 129, 184))}]

list_asoc[15] = 4 , 4
[{((229, 225, 156), (163, 73, 164)), ((63, 185, 238), (163, 73, 164)), ((237, 28, 36), (163, 73, 164)), ((163, 73, 164), (163, 73, 164))}, {((163, 73, 164), (229, 225, 156)), ((163, 73, 164), (237, 28, 36)), ((163, 73, 164), (63, 185, 238)), ((163, 73, 164), (163, 73, 164))}]

list_asoc[16] = 10 , 11
[{((5, 162, 231), (4, 161, 231)), ((16, 167, 233), (4, 161, 231)), ((195, 195, 195), (4, 161, 231)), ((0, 0, 0), (4, 161, 231)), ((2, 162, 230), (4, 161, 231)), ((1, 160, 231), (4, 161, 231)), ((63, 185, 238), (4, 161, 231)), ((4, 161, 231), (4, 161, 231)), ((2, 161, 230), (4, 161, 231)), ((5, 163, 230), (4, 161, 231))}, {((4, 161, 231), (2, 161, 230)), ((4, 161, 231), (5, 163, 230)), ((4, 161, 231), (5, 162, 231)), ((4, 161, 231), (63, 185, 238)), ((4, 161, 231), (16, 167, 233)), ((4, 161, 231), (0, 0, 0)), ((4, 161, 231), (1, 161, 230)),

((4, 161, 231), (195, 195, 195)), ((4, 161, 231), (4, 161, 231)), ((4, 161, 231), (1, 160, 231)), ((4, 161, 231), (2, 162, 230))}]

list_asoc[17] = 1 , 1
[{((4, 161, 231), (5, 162, 231))}, {((5, 162, 231), (4, 161, 231))}]

list_asoc[18] = 8 , 7
[{((2, 161, 230), (2, 162, 230)), ((1, 160, 231), (2, 162, 230)), ((0, 0, 0), (2, 162, 230)), ((5, 163, 230), (2, 162, 230)), ((1, 160, 230), (2, 162, 230)), ((16, 167, 233), (2, 162, 230)), ((2, 162, 230), (2, 162, 230)), ((4, 161, 231), (2, 162, 230))}, {((2, 162, 230), (1, 161, 230)), ((2, 162, 230), (0, 0, 0)), ((2, 162, 230), (4, 161, 231)), ((2, 162, 230), (16, 167, 233)), ((2, 162, 230), (1, 160, 231)), ((2, 162, 230), (2, 162, 230)), ((2, 162, 230), (2, 161, 230))}]

list_asoc[19] = 10 , 10
[{((2, 161, 230), (1, 160, 231)), ((5, 163, 230), (1, 160, 231)), ((195, 195, 195), (1, 160, 231)), ((1, 160, 230), (1, 160, 231)), ((0, 0, 0), (1, 160, 231)), ((16, 167, 233), (1, 160, 231)), ((1, 161, 230), (1, 160, 231)), ((2, 162, 230), (1, 160, 231)), ((1, 160, 231), (1, 160, 231)), ((4, 161, 231), (1, 160, 231))}, {((1, 160, 231), (2, 161, 230)), ((1, 160, 231), (2, 162, 230)), ((1, 160, 231), (5, 163, 230)), ((1, 160, 231), (1, 160, 230)), ((1, 160, 231), (1, 161, 230)), ((1, 160, 231), (16, 167, 233)), ((1, 160, 231), (4, 161, 231)), ((1, 160, 231), (0, 0, 0)), ((1, 160, 231), (195, 195, 195)), ((1, 160, 231), (1, 160, 231))}]

list_asoc[20] = 5 , 5
[{((1, 160, 231), (1, 160, 230)), ((0, 0, 0), (1, 160, 230)), ((1, 160, 230), (1, 160, 230)), ((195, 195, 195), (1, 160, 230)), ((1, 161, 230), (1, 160, 230))}, {((1, 160, 230), (1, 160, 231)), ((1, 160, 230), (2, 162, 230)), ((1, 160, 230), (0, 0, 0)), ((1, 160, 230), (1, 160, 230)), ((1, 160, 230), (1, 161, 230))}]

list_asoc[21] = 1 , 1
[{((64, 185, 238), (65, 186, 238))}, {((65, 186, 238), (63, 185, 238))}]

list_asoc[22] = 5 , 6
[{((1, 160, 231), (2, 161, 230)), ((2, 161, 230), (2, 161, 230)), ((16, 167, 233), (2, 161, 230)), ((4, 161, 231), (2, 161, 230)), ((2, 162, 230), (2, 161, 230))}, {((2, 161, 230), (1, 160, 231)), ((2, 161, 230), (2, 162, 230)), ((2, 161, 230), (2, 161, 230)), ((2, 161, 230), (0, 0, 0)), ((2, 161, 230), (1, 161, 230)), ((2, 161, 230), (4, 161, 231))}]

list_asoc[23] = 6 , 5
[{((2, 162, 230), (1, 161, 230)), ((1, 160, 231), (1, 161, 230)), ((4, 161, 231), (1, 161, 230)), ((2, 161, 230), (1, 161, 230)), ((1, 160, 230), (1, 161, 230)), ((1, 161, 230), (1, 161, 230))}, {((1, 161, 230), (1, 160, 231)), ((1, 161, 230), (2, 162, 230)), ((1, 161, 230), (5, 163, 230)), ((1, 161, 230), (1, 160, 230)), ((1, 161, 230), (1, 161, 230))}]

list_asoc[24] = 2 , 2
[{((229, 225, 156), (127, 127, 127)), ((127, 127, 127), (127, 127, 127))}, {((127, 127, 127), (229, 225, 156)), ((127, 127, 127), (127, 127, 127))}]

list_asoc[25] = 8 , 7

[{((0, 0, 0), (5, 163, 230)), ((16, 167, 233), (5, 163, 230)), ((2, 162, 230), (5, 163, 230)), ((1, 160, 231), (5, 163, 230)), ((5, 163, 230), (5, 163, 230)), ((1, 161, 230), (5, 163, 230)), ((4, 161, 231), (5, 163, 230)), ((195, 195, 195), (5, 163, 230))}, {((5, 163, 230), (0, 0, 0)), ((5, 163, 230), (63, 185, 238)), ((5, 163, 230), (1, 160, 231)), ((5, 163, 230), (2, 162, 230)), ((5, 163, 230), (16, 167, 233)), ((5, 163, 230), (5, 163, 230)), ((5, 163, 230), (4, 161, 231))}]

list_asoc[26] = 2 , 2
[{((16, 167, 233), (17, 167, 233)), ((63, 185, 238), (17, 167, 233))}, {((17, 167, 233), (63, 185, 238)), ((17, 167, 233), (16, 167, 233))}]

list_asoc[27] = 4 , 4
[{((64, 185, 238), (255, 174, 201)), ((255, 174, 201), (255, 174, 201)), ((63, 185, 238), (255, 174, 201)), ((229, 225, 156), (255, 174, 201))}, {((255, 174, 201), (63, 185, 238)), ((255, 174, 201), (255, 174, 201)), ((255, 174, 201), (64, 185, 238)), ((255, 174, 201), (229, 225, 156))}]

list_asoc[28] = 1 , 1
[{((1, 161, 230), (2, 162, 230))}, {((2, 162, 230), (5, 163, 230))}]

list_asoc[29] = 1 , 1
[{((63, 185, 238), (65, 186, 238))}, {((65, 186, 238), (64, 185, 238))}]

Szczegóły asocjacji:

list_asoc[0] = 9 , 10
[{((63, 185, 238), (229, 225, 156)), ((163, 73, 164), (229, 225, 156)), ((127, 127, 127), (229, 225, 156)), ((195, 195, 195), (229, 225, 156)), ((229, 225, 156), (229, 225, 156)), ((240, 69, 76), (229, 225, 156)), ((255, 174, 201), (229, 225, 156)), ((64, 185, 238), (229, 225, 156)), ((237, 28, 36), (229, 225, 156))}, {((229, 225, 156), (195, 195, 195)), ((229, 225, 156), (237, 28, 36)), ((229, 225, 156), (240, 69, 76)), ((229, 225, 156), (229, 225, 156)), ((229, 225, 156), (64, 185, 238)), ((229, 225, 156), (127, 127, 127)), ((229, 225, 156), (163, 73, 164)), ((229, 225, 156), (63, 185, 238)), ((229, 225, 156), (255, 174, 201)), ((229, 225, 156), (16, 167, 233))}]

Zestaw użytych pikseli:
(((63, 185, 238), (163, 73, 164), (127, 127, 127), (195, 195, 195), (229, 225, 156), (240, 69, 76), (255, 174, 201), (64, 185, 238), (237, 28, 36)), (229, 225, 156), ((195, 195, 195), (237, 28, 36), (240, 69, 76), (229, 225, 156), (64, 185, 238), (127, 127, 127), (163, 73, 164), (63, 185, 238), (255, 174, 201), (16, 167, 233)))

Zestaw użytych kolorów
Ilość kolorów: 20

Ilość zgodności: 14657

list_asoc[1] = 8 , 8
[{((229, 225, 156), (237, 28, 36)), ((240, 69, 76), (237, 28, 36)), ((59, 128, 183), (237, 28, 36)), ((0, 0, 0), (237, 28, 36)), ((237, 28, 36), (237, 28, 36)), ((59, 129, 184), (237, 28, 36)), ((163, 73, 164), (237, 28, 36)), ((195, 195, 195), (237, 28, 36))}, {((237, 28, 36), (163, 73, 164)), ((237, 28, 36), (0, 0, 0)), ((237, 28, 36), (59, 128, 183)), ((237, 28, 36), (195, 195, 195)), ((237, 28, 36), (237, 28, 36)), ((237, 28, 36), (240, 69, 76)), ((237, 28, 36), (59, 129, 184)), ((237, 28, 36), (229, 225, 156))}]

Zestaw użytych pikseli:
(((229, 225, 156), (240, 69, 76), (59, 128, 183), (0, 0, 0), (237, 28, 36), (59, 129, 184), (163, 73, 164), (195, 195, 195)), (237, 28, 36), ((163, 73, 164), (0, 0, 0), (59, 128, 183), (195, 195, 195), (237, 28, 36), (240, 69, 76), (59, 129, 184), (229, 225, 156)))

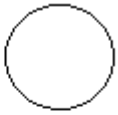
Zestaw użytych kolorów
Ilość kolorów: 17

Ilość zgodności: 14561

list_asoc[4] = 10 , 9
[{((237, 28, 36), (0, 0, 0)), ((4, 161, 231), (0, 0, 0)), ((2, 162, 230), (0, 0, 0)), ((2, 161, 230), (0, 0, 0)), ((5, 163, 230), (0, 0, 0)), ((1, 160, 231), (0, 0, 0)), ((63, 185, 238), (0, 0, 0)), ((0, 0, 0), (0, 0, 0)), ((1, 160, 230), (0, 0, 0)), ((16, 167, 233), (0, 0, 0))}, {((0, 0, 0), (1, 160, 231)), ((0, 0, 0), (2, 162, 230)), ((0, 0, 0), (5, 163, 230)), ((0, 0, 0), (1, 160, 230)), ((0, 0, 0), (63, 185, 238)), ((0, 0, 0), (0, 0, 0)), ((0, 0, 0), (237, 28, 36)), ((0, 0, 0), (16, 167, 233)), ((0, 0, 0), (4, 161, 231))}]

Zestaw użytych pikseli:
(((237, 28, 36), (4, 161, 231), (2, 162, 230), (2, 161, 230), (5, 163, 230), (1, 160, 231), (63, 185, 238), (0, 0, 0), (1, 160, 230), (16, 167, 233)), (0, 0, 0), ((1, 160, 231), (2, 162, 230), (5, 163, 230), (1, 160, 230), (63, 185, 238), (0, 0, 0), (237, 28, 36), (16, 167, 233), (4, 161, 231)))

Zestaw użytych kolorów
Ilość kolorów: 20

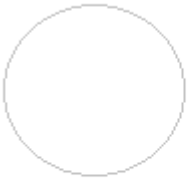


Ilość zgodności: 344

list_asoc[5] = 8 , 10
[{((195, 195, 195), (195, 195, 195)), ((229, 225, 156), (195, 195, 195)), ((1, 160, 231), (195, 195, 195)), ((237, 28, 36), (195, 195, 195)), ((4, 161, 231), (195, 195, 195)), ((63, 185, 238), (195, 195, 195)), ((59, 129, 184), (195, 195, 195)), ((16, 167, 233), (195, 195, 195))}, {((195, 195, 195), (195, 195, 195)), ((195, 195, 195), (1, 160, 231)), ((195, 195, 195), (229, 225, 156)), ((195, 195, 195), (5, 163, 230)), ((195, 195, 195), (64, 185, 238)), ((195, 195, 195), (1, 160, 230)), ((195, 195, 195), (63, 185, 238)), ((195, 195, 195), (237, 28, 36)), ((195, 195, 195), (16, 167, 233)), ((195, 195, 195), (4, 161, 231))}]

Zestaw użytych pikseli:
(((195, 195, 195), (229, 225, 156), (1, 160, 231), (237, 28, 36), (4, 161, 231), (63, 185, 238), (59, 129, 184), (16, 167, 233)), (195, 195, 195), ((195, 195, 195), (1, 160, 231), (229, 225, 156), (5, 163, 230), (64, 185, 238), (1, 160, 230), (63, 185, 238), (237, 28, 36), (16, 167, 233), (4, 161, 231)))

Zestaw użytych kolorów
Ilość kolorów: 19



Ilość zgodności: 246

list_asoc[6] = 7 , 5
[{((59, 129, 184), (59, 129, 184)), ((63, 182, 235), (59, 129, 184)), ((59, 128, 183), (59, 129, 184)), ((15, 153, 219), (59, 129, 184)), ((63, 183, 236), (59, 129, 184)), ((237, 28, 36), (59, 129, 184)), ((63, 185, 238), (59, 129, 184))}, {((59, 129, 184), (59, 129, 184)), ((59, 129, 184), (15, 153, 219)), ((59, 129, 184), (59, 128, 183)), ((59, 129, 184), (237, 28, 36)), ((59, 129, 184), (195, 195, 195))}]

Zestaw użytych pikseli:
(((59, 129, 184), (63, 182, 235), (59, 128, 183), (15, 153, 219), (63, 183, 236), (237, 28, 36), (63, 185, 238)), (59, 129, 184), ((59, 129, 184), (15, 153, 219), (59, 128, 183), (237, 28, 36), (195, 195, 195)))

Zestaw użytych kolorów
Ilość kolorów: 13

Ilość zgodności: 1363

list_asoc[7] = 4 , 4
[{((59, 129, 184), (59, 128, 183)), ((59, 128, 183), (59, 128, 183)), ((15, 153, 219), (59, 128, 183)), ((237, 28, 36), (59, 128, 183))}, {((59, 128, 183), (59, 129, 184)), ((59, 128, 183), (59, 128, 183)), ((59, 128, 183), (237, 28, 36)), ((59, 128, 183), (15, 153, 219))}]

Zestaw użytych pikseli:
(((59, 129, 184), (59, 128, 183), (15, 153, 219), (237, 28, 36)), (59, 128, 183), ((59, 129, 184), (59, 128, 183), (237, 28, 36), (15, 153, 219)))

Ilość użytych kolorów
Ilość kolorów: 9

Ilość zgodności: 64

list_asoc[8] = 6 , 4
[{((15, 153, 219), (15, 153, 219)), ((4, 157, 228), (15, 153, 219)), ((63, 185, 238), (15, 153, 219)), ((59, 129, 184), (15, 153, 219)), ((16, 167, 233), (15, 153, 219)), ((59, 128, 183), (15, 153, 219))}, {((15, 153, 219), (15, 153, 219)), ((15, 153, 219), (59, 129, 184)), ((15, 153, 219), (59, 128, 183)), ((15, 153, 219), (4, 157, 228))}]

Zestaw użytych pikseli:
(((15, 153, 219), (4, 157, 228), (63, 185, 238), (59, 129, 184), (16, 167, 233), (59, 128, 183)), (15, 153, 219), ((15, 153, 219), (59, 129, 184), (59, 128, 183), (4, 157, 228)))

Ilość użytych kolorów
Ilość kolorów: 11

Ilość zgodności: 186

list_asoc[9] = 2 , 2
[{((4, 157, 228), (4, 157, 228)), ((15, 153, 219), (4, 157, 228))}, {((4, 157, 228), (15, 153, 219)), ((4, 157, 228), (4, 157, 228))}]

Zestaw użytych pikseli:
(((4, 157, 228), (15, 153, 219)), (4, 157, 228), ((15, 153, 219), (4, 157, 228)))

Ilość użytych kolorów
Ilość kolorów: 5

Ilość zgodności: 12

list_asoc[10] = 12 , 15
[{((64, 185, 238), (63, 185, 238)), ((17, 167, 233), (63, 185, 238)), ((255, 174, 201), (63, 185, 238)), ((4, 161, 231), (63, 185, 238)), ((63, 185, 238), (63, 185, 238)), ((163, 73, 164), (63, 185, 238)), ((5, 163, 230), (63, 185, 238)), ((0, 0, 0), (63, 185, 238)), ((16, 167, 233), (63, 185, 238)), ((65, 186, 238), (63, 185, 238)), ((195, 195, 195), (63, 185, 238)), ((229, 225, 156), (63, 185, 238))}, {((63, 185, 238), (229, 225, 156)), ((63, 185, 238), (63, 183, 236)), ((63, 185, 238), (65, 186, 238)), ((63, 185, 238), (255, 174,

201)), ((63, 185, 238), (64, 185, 238)), ((63, 185, 238), (15, 153, 219)), ((63, 185, 238), (163, 73, 164)), ((63, 185, 238), (63, 182, 235)), ((63, 185, 238), (63, 185, 238)), ((63, 185, 238), (16, 167, 233)), ((63, 185, 238), (0, 0, 0)), ((63, 185, 238), (4, 161, 231)), ((63, 185, 238), (195, 195, 195)), ((63, 185, 238), (17, 167, 233)), ((63, 185, 238), (59, 129, 184))}]

Zestaw użytych pikseli:
(((64, 185, 238), (17, 167, 233), (255, 174, 201), (4, 161, 231), (63, 185, 238), (163, 73, 164), (5, 163, 230), (0, 0, 0), (16, 167, 233), (65, 186, 238), (195, 195, 195), (229, 225, 156)), (63, 185, 238), ((229, 225, 156), (63, 183, 236), (65, 186, 238), (255, 174, 201), (64, 185, 238), (15, 153, 219), (163, 73, 164), (63, 182, 235), (63, 185, 238), (16, 167, 233), (0, 0, 0), (4, 161, 231), (195, 195, 195), (17, 167, 233), (59, 129, 184)))

Ilość użytych kolorów
Ilość kolorów: 28

Ilość zgodności: 9726

list_asoc[12] = 7 , 6
[{((64, 185, 238), (64, 185, 238)), ((255, 174, 201), (64, 185, 238)), ((63, 185, 238), (64, 185, 238)), ((16, 167, 233), (64, 185, 238)), ((65, 186, 238), (64, 185, 238)), ((195, 195, 195), (64, 185, 238)), ((229, 225, 156), (64, 185, 238))}, {((64, 185, 238), (64, 185, 238)), ((64, 185, 238), (63, 185, 238)), ((64, 185, 238), (255, 174, 201)), ((64, 185, 238), (16, 167, 233)), ((64, 185, 238), (229, 225, 156)), ((64, 185, 238), (65, 186, 238))}]

Zestaw użytych pikseli:
(((64, 185, 238), (255, 174, 201), (63, 185, 238), (16, 167, 233), (65, 186, 238), (195, 195, 195), (229, 225, 156)), (64, 185, 238), ((64, 185, 238), (63, 185, 238), (255, 174, 201), (16, 167, 233), (229, 225, 156), (65, 186, 238)))

Ilość użytych kolorów
Ilość kolorów: 14

Ilość zgodności: 532

list_asoc[13] = 11 , 12
[{((0, 0, 0), (16, 167, 233)), ((17, 167, 233), (16, 167, 233)), ((4, 161, 231), (16, 167, 233)), ((2, 162, 230), (16, 167, 233)), ((64, 185, 238), (16, 167, 233)), ((63, 185, 238), (16, 167, 233)), ((1, 160, 231), (16, 167, 233)), ((5, 163, 230), (16, 167, 233)), ((16, 167, 233), (16, 167, 233)), ((195, 195, 195), (16, 167, 233)), ((229, 225, 156), (16, 167, 233))}, {((16, 167, 233), (17, 167, 233)), ((16, 167, 233), (0, 0, 0)), ((16, 167, 233), (4, 161, 231)), ((16, 167, 233), (1, 160, 231)), ((16, 167, 233), (2, 162, 230)), ((16, 167, 233), (2, 161, 230)), ((16, 167, 233), (5, 163, 230)), ((16, 167, 233), (64, 185, 238)), ((16, 167, 233), (15, 153, 219)), ((16, 167, 233), (63, 185, 238)), ((16, 167, 233), (16, 167, 233)), ((16, 167, 233), (195, 195, 195))}]

Zestaw użytych pikseli:
(((0, 0, 0), (17, 167, 233), (4, 161, 231), (2, 162, 230), (64, 185, 238), (63, 185, 238), (1, 160, 231), (5, 163, 230), (16, 167, 233), (195, 195, 195), (229, 225, 156)), (16, 167, 233), ((17, 167, 233), (0, 0, 0), (4, 161, 231), (1, 160, 231), (2, 162, 230), (2, 161, 230), (5, 163, 230), (64, 185, 238), (15, 153, 219), (63, 185, 238), (16, 167, 233), (195, 195, 195)))

Ilość użytych kolorów
Ilość kolorów: 24

Ilość zgodności: 6230

list_asoc[15] = 4 , 4
[{((229, 225, 156), (163, 73, 164)), ((63, 185, 238), (163, 73, 164)), ((237, 28, 36), (163, 73, 164)), ((163, 73, 164), (163, 73, 164))}, {((163, 73, 164), (229, 225, 156)), ((163, 73, 164), (237, 28, 36)), ((163, 73, 164), (63, 185, 238)), ((163, 73, 164), (163, 73, 164))}]

Zestaw użytych pikseli:
(((229, 225, 156), (63, 185, 238), (237, 28, 36), (163, 73, 164)), (163, 73, 164), ((229, 225, 156), (237, 28, 36), (63, 185, 238), (163, 73, 164)))

Ilość użytych kolorów
Ilość kolorów: 9

Ilość zgodności: 104

list_asoc[16] = 10 , 11
[{((5, 162, 231), (4, 161, 231)), ((16, 167, 233), (4, 161, 231)), ((195, 195, 195), (4, 161, 231)), ((0, 0, 0), (4, 161, 231)), ((2, 162, 230), (4, 161, 231)), ((1, 160, 231), (4, 161, 231)), ((63, 185, 238), (4, 161, 231)), ((4, 161, 231), (4, 161, 231)), ((2, 161, 230), (4, 161, 231)), ((5, 163, 230), (4, 161, 231))}, {((4,

161, 231), (2, 161, 230)), ((4, 161, 231), (5, 163, 230)), ((4, 161, 231), (5, 162, 231)), ((4, 161, 231), (63, 185, 238)), ((4, 161, 231), (16, 167, 233)), ((4, 161, 231), (0, 0, 0)), ((4, 161, 231), (1, 161, 230)), ((4, 161, 231), (195, 195, 195)), ((4, 161, 231), (4, 161, 231)), ((4, 161, 231), (1, 160, 231)), ((4, 161, 231), (2, 162, 230))}]

Zestaw użytych pikseli:
(((5, 162, 231), (16, 167, 233), (195, 195, 195), (0, 0, 0), (2, 162, 230), (1, 160, 231), (63, 185, 238), (4, 161, 231), (2, 161, 230), (5, 163, 230)), (4, 161, 231), ((2, 161, 230), (5, 163, 230), (5, 162, 231), (63, 185, 238), (16, 167, 233), (0, 0, 0), (1, 161, 230), (195, 195, 195), (4, 161, 231), (1, 160, 231), (2, 162, 230)))

Ilość użytych kolorów
Ilość kolorów: 22

Ilość zgodności: 1316

list_asoc[18] = 8 , 7
[{((2, 161, 230), (2, 162, 230)), ((1, 160, 231), (2, 162, 230)), ((0, 0, 0), (2, 162, 230)), ((5, 163, 230), (2, 162, 230)), ((1, 160, 230), (2, 162, 230)), ((16, 167, 233), (2, 162, 230)), ((2, 162, 230), (2, 162, 230)), ((4, 161, 231), (2, 162, 230))}, {((2, 162, 230), (1, 161, 230)), ((2, 162, 230), (0, 0, 0)), ((2, 162, 230), (4, 161, 231)), ((2, 162, 230), (16, 167, 233)), ((2, 162, 230), (1, 160, 231)), ((2, 162, 230), (2, 162, 230)), ((2, 162, 230), (2, 161, 230))}]

Zestaw użytych pikseli:
(((2, 161, 230), (1, 160, 231), (0, 0, 0), (5, 163, 230), (1, 160, 230), (16, 167, 233), (2, 162, 230), (4, 161, 231)), (2, 162, 230), ((1, 161, 230), (0, 0, 0), (4, 161, 231), (16, 167, 233), (1, 160, 231), (2, 162, 230), (2, 161, 230)))

Ilość użytych kolorów
Ilość kolorów: 16

Ilość zgodności: 116

list_asoc[19] = 10 , 10
[{((2, 161, 230), (1, 160, 231)), ((5, 163, 230), (1, 160, 231)), ((195, 195, 195), (1, 160, 231)), ((1, 160, 230), (1, 160, 231)), ((0, 0, 0), (1, 160, 231)), ((16, 167, 233), (1, 160, 231)), ((1, 161, 230), (1, 160, 231)), ((2, 162, 230), (1, 160, 231)), ((1, 160, 231), (1, 160, 231)), ((4, 161, 231), (1, 160, 231))}, {((1, 160, 231), (2, 161, 230)), ((1, 160, 231), (2, 162, 230)), ((1, 160, 231), (5, 163, 230)), ((1, 160, 231), (1, 160, 230)), ((1, 160, 231), (1, 161, 230)), ((1, 160, 231), (16, 167, 233)), ((1, 160, 231), (4, 161, 231)), ((1, 160, 231), (0, 0, 0)), ((1, 160, 231), (195, 195, 195)), ((1, 160, 231), (1, 160, 231))}]

Zestaw użytych pikseli:
(((2, 161, 230), (5, 163, 230), (195, 195, 195), (1, 160, 230), (0, 0, 0), (16, 167, 233), (1, 161, 230), (2, 162, 230), (1, 160, 231), (4, 161, 231)), (1, 160, 231), ((2, 161, 230), (2, 162, 230), (5, 163, 230), (1, 160, 230), (1, 161, 230), (16, 167, 233), (4, 161, 231), (0, 0, 0), (195, 195, 195), (1, 160, 231)))

Ilość użytych kolorów
Ilość kolorów: 21

Ilość zgodności: 1561

list_asoc[20] = 5 , 5
[{((1, 160, 231), (1, 160, 230)), ((0, 0, 0), (1, 160, 230)), ((1, 160, 230), (1, 160, 230)), ((195, 195, 195), (1, 160, 230)), ((1, 161, 230), (1, 160, 230))}, {((1, 160, 230), (1, 160, 231)), ((1, 160, 230), (2, 162, 230)), ((1, 160, 230), (0, 0, 0)), ((1, 160, 230), (1, 160, 230)), ((1, 160, 230), (1, 161, 230))}]

Zestaw użytych pikseli:
(((1, 160, 231), (0, 0, 0), (1, 160, 230), (195, 195, 195), (1, 161, 230)), (1, 160, 230), ((1, 160, 231), (2, 162, 230), (0, 0, 0), (1, 160, 230), (1, 161, 230)))

Ilość użytych kolorów
Ilość kolorów: 11

Ilość zgodności: 82

list_asoc[22] = 5 , 6
[{((1, 160, 231), (2, 161, 230)), ((2, 161, 230), (2, 161, 230)), ((16, 167, 233), (2, 161, 230)), ((4, 161, 231), (2, 161, 230)), ((2, 162, 230), (2, 161, 230))}, {((2, 161, 230), (1, 160, 231)), ((2, 161, 230), (2, 162, 230)), ((2, 161, 230), (2, 161, 230)), ((2, 161, 230), (0, 0, 0)), ((2, 161, 230), (1, 161, 230)), ((2, 161, 230), (4, 161, 231))}]

Zestaw użytych pikseli:
(((1, 160, 231), (2, 161, 230), (16, 167, 233), (4, 161, 231), (2, 162, 230)), (2, 161, 230), ((1, 160, 231), (2, 162, 230), (2, 161, 230), (0, 0, 0), (1, 161, 230), (4, 161, 231)))

Ilość użytych kolorów
Ilość kolorów: 12

Ilość zgodności: 43

list_asoc[23] = 6 , 5
[{((2, 162, 230), (1, 161, 230)), ((1, 160, 231), (1, 161, 230)), ((4, 161, 231), (1, 161, 230)), ((2, 161, 230), (1, 161, 230)), ((1, 160, 230), (1, 161, 230)), ((1, 161, 230), (1, 161, 230))}, {((1, 161, 230), (1, 160, 231)), ((1, 161, 230), (2, 162, 230)), ((1, 161, 230), (5, 163, 230)), ((1, 161, 230), (1, 160, 230)), ((1, 161, 230), (1, 161, 230))}]

Zestaw użytych pikseli:
(((2, 162, 230), (1, 160, 231), (4, 161, 231), (2, 161, 230), (1, 160, 230), (1, 161, 230)), (1, 161, 230), ((1, 160, 231), (2, 162, 230), (5, 163, 230), (1, 160, 230), (1, 161, 230)))

Ilość użytych kolorów
Ilość kolorów: 12

Ilość zgodności: 54

list_asoc[24] = 2 , 2
[{((229, 225, 156), (127, 127, 127)), ((127, 127, 127), (127, 127, 127))}, {((127, 127, 127), (229, 225, 156)), ((127, 127, 127), (127, 127, 127))}]

(((229, 225, 156), (127, 127, 127)), (127, 127, 127), ((229, 225, 156), (127, 127, 127)))

Ilość użytych kolorów
Ilość kolorów: 5

Ilość zgodności: 110

list_asoc[25] = 8 , 7
[{((0, 0, 0), (5, 163, 230)), ((16, 167, 233), (5, 163, 230)), ((2, 162, 230), (5, 163, 230)), ((1, 160, 231), (5, 163, 230)), ((5, 163, 230), (5, 163, 230)), ((1, 161, 230), (5, 163, 230)), ((4, 161, 231), (5, 163, 230)), ((195, 195, 195), (5, 163, 230))}, {((5, 163, 230), (0, 0, 0)), ((5, 163, 230), (63, 185, 238)), ((5, 163, 230), (1, 160, 231)), ((5, 163, 230), (2, 162, 230)), ((5, 163, 230), (16, 167, 233)), ((5, 163, 230), (5, 163, 230)), ((5, 163, 230), (4, 161, 231))}]

Zestaw użytych pikseli:
(((0, 0, 0), (16, 167, 233), (2, 162, 230), (1, 160, 231), (5, 163, 230), (1, 161, 230), (4, 161, 231), (195, 195, 195)), (5, 163, 230), ((0, 0, 0), (63, 185, 238), (1, 160, 231), (2, 162, 230), (16, 167, 233), (5, 163, 230), (4, 161, 231)))

Ilość użytych kolorów
Ilość kolorów: 16

Ilość zgodności: 94

list_asoc[27] = 4 , 4
[{((64, 185, 238), (255, 174, 201)), ((255, 174, 201), (255, 174, 201)), ((63, 185, 238), (255, 174, 201)), ((229, 225, 156), (255, 174, 201))}, {((255, 174, 201), (63, 185, 238)), ((255, 174, 201), (255, 174, 201)), ((255, 174, 201), (64, 185, 238)), ((255, 174, 201), (229, 225, 156))}]

Zestaw użytych pikseli:
(((64, 185, 238), (255, 174, 201), (63, 185, 238), (229, 225, 156)), (255, 174, 201), ((63, 185, 238), (255, 174, 201), (64, 185, 238), (229, 225, 156)))

Ilość użytych kolorów
Ilość kolorów: 9

Ilość zgodności: 88

Dodatkowe zalety Python-a

W pythonie w trybie konwersacji wprowadzamy definicje funkcji i ich treści a następnie wywołujemy funkcje, z zależności od potrzeby. Nie opuszczając interpretera możemy przy tym wykonywać działania arytmetyczne i otrzymywać wyniki tych działań, sprawdzać działanie języka python i poprawność interesującego zapisu, podmieniać treści definicji funkcji pod warunkiem tych samych deklaracji funkcji, zwracać swobodnie wiele wartości, zaoszczędzamy wiele czasu w stosunku do języków kompilowanych. Sprawia to, że python nadaje się do prac naukowych, gdzie tworzymy i sprawdzamy poprawność różnych teorii.

Bibliografia

1. Waldemar Wietrzykowski, *Rastrowe asocjacje regularne*, IIS 2022
2. Waldemar Wietrzykowski, *Rozpoznanie w asocjacjach ponadregularnych*, IIS 2021
3. Waldemar Wietrzykowski, *Asocjacje ponadregularne*, IIS 2021
4. Waldemar Wietrzykowski, *Implementacja asocjacyjnej maszyny Turinga*, IIS 2021
5. Waldemar Wietrzykowski, *Maszyna asocjacyjna*, IIS 2021
6. Waldemar Wietrzykowski, *Asocjacje regularne*, IIS 2021
7. Waldemar Wietrzykowski, *Subiektywne badanie pamięci*, IIS 2020
8. Waldemar Wietrzykowski, *Konsolidacja elementarna*, IIS 2019
9. Waldemar Wietrzykowski, *Naturalny System Operacyjny*, IIS 2019
10. Waldemar Wietrzykowski, *Dwie maszyny*, IIS 2018
11. Waldemar Wietrzykowski, *Uczenie języka naturalnego*, IIS 2018
12. Waldemar Wietrzykowski, *Implementacja rachunku sekwentowego*, DIL 2018
13. Waldemar Wietrzykowski, *Wystarczalność rachunku sekwentowego. Rachunek tautologiczny*, DIL 2018
14. Waldemar Wietrzykowski, *Teoria wiedzy*, DIL 2018
15. Waldemar Wietrzykowski, *Teoria wiedzy teoretycznej*, DIL 2018
16. Waldemar Wietrzykowski, *Teoria wiedzy praktycznej*, DIL 2018
17. Waldemar Wietrzykowski, *Teoria świadomej maszyny*, DIL 2018
18. Waldemar Wietrzykowski, *Założenia wstępne do projektu świadomej maszyny*, DIL 2017
19. Waldemar Wietrzykowski, *Świadoma maszyna*, DIL 2017
20. Waldemar Wietrzykowski, *Świadoma reakcja maszyn*, DIL 2017
21. Waldemar Wietrzykowski, *Samoucząca się maszyna motoryczna*, DIL 2017
22. Waldemar Wietrzykowski, *Strumień świadomości*, DIL 2017
23. Waldemar Wietrzykowski, S*calanie interpretacji maszyn*, DIL 2017
24. Waldemar Wietrzykowski, *Obraz rzeczywistości z poziomu maszyny*, DIL 2017
25. Waldemar Wietrzykowski, *Świadomość wielu maszyn*, DIL 2017
26. Waldemar Wietrzykowski, *Wiele maszyn*, DIL 2017
27. Waldemar Wietrzykowski, *Znaczenie interpretacji maszyn*, DIL 2017
28. Waldemar Wietrzykowski, *Interpretacje maszyn*, DIL 2016
29. Waldemar Wietrzykowski, *Jednofunkcyjne maszyny*, DIL 2016
30. Waldemar Wietrzykowski, *Samoucząca się maszyna*, DIL 2016
31. Waldemar Wietrzykowski, *Procesor samouczący się*, DIL 2016
32. Waldemar Wietrzykowski, *Processor self-learning*, DIL 2016
