Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Damian Łoziński
Nr albumu: 234352

Autonomiczny rozwój języka komunikacji przez zespół agentów

Praca magisterska
na kierunku MATEMATYKA

Praca wykonana pod kierunkiem
prof. dra hab. Andrzeja Skowrona
Instytut Matematyki

Czerwiec 2011

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data Podpis kierującego pracą

Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpis autora (autorów) pracy

Streszczenie

Celem niniejszej pracy jest zbudowanie modelu symulującego proces ewolucji języka, a ściślej modelu umożliwiającego analizę powstawania słowników oraz narzędzia umożliwiającego przeprowadzanie symulacji. Na model składa się zespół autonomicznych agentów, które w procesie ewolucyjnym budują słowniki umożliwiające komunikację. Procesy służące odkrywaniu właściwego zbioru słów są optymalizowane z uwagi na jakość języka, narzuconą m.in. zdolnością adaptacyjną. Badam również szybkość ewolucji prowadzącej do odkrycia języka. Symulacja zaimplementowana jest w języku Python.

Słowa kluczowe

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.0 Matematyka, Informatyka
11.4 Sztuczna inteligencja

Klasyfikacja tematyczna

68T05 Learning and adaptive systems
68T37 Reasoning under uncertainity

Tytuł pracy w języku angielskim

The autonomous development of a communication language by an agent team

Spis treści

Wprowadzenie

Motywacja

Badania nad procesem powstawania i kształtowania języka prowadzone są przez lingwistów od dawna. Od 15 lat próbuje się symulować powstawanie języka za pomocą metod sztucznej inteligencji[1]. W ostatnich latach naukowcy powracają to próby realizacji pierwotnych celów powstania sztucznej inteligencji, czyli stworzenia inteligencji podobnej do inteligencji człowieka – tzw. ogólnej sztucznej inteligencji.

Wcześniej intensywnie eksplorowane były różne wąskie obszary, które pozwalały na uzyskanie satysfakcjonujących wyników, natomiast problem ogólnej inteligencji uważany był za zbyt złożony. Obecnie ogólna sztuczna inteligencja wydaje się wychodzić z powrotem na pierwszy plan o czym może świadczyć zainteresowanie konferencjami AGI[2] przez liderów światowego rynku w dziedzinie IT.

Jednym z przejawów inteligencji jest zdolność do porozumiewania się. Nauczenie maszyny rozumienia języka naturalnego jest podstawowym problemem współczesnej ogólnej sztucznej inteligencji. W tym celu bada się szereg procesów kształtujących język. Jednym z nich jest powstawanie map (słowników) w „mózgach" autonomicznych agentów, podczas procesu kształtowania języka od zera. Następnie można badać jak owe mapy zachowują się w zmiennym środowisku.

Niniejsza praca opiera się na założeniach, że język jest systemem ewolucyjnym, który podlega nieustannym zmianom. Żaden z członków populacji nie ma pełnej wiedzy o języku, każdy ma swoje subiektywne pojęcie o języku, które sam wypracował. Przy tych, wydaje się, że naturalnych założeniach, w naturalny sposób pojawia się problem *niepewności referencyjnej*[3], tzn. niemożliwości dokonania powiązania słowa z jego kompletnym i pełnym znaczeniem przez pojedynczego agenta w trakcie uczenia. Dziecko widząc rzecz i słysząc słowo dokonuje szybkiego przypisania. Zjawisko to nazywane po angielsku *fast mapping* opisane jest m.in. w pracach Blooma [10] i Carey'a [11]. Bowermann i Choi [12] uznają, że dziecko nie przypisuje kompletnego i poprawnego znaczenia słowa przy pierwszym kontakcie, a kształtuje je każdorazowo słysząc i widząc opisywany obiekt w różnych kontekstach. Poparciem tej tezy może być przykład uogólniania nowo poznanych słów. Zdarza się, że dzieci mówią „koń", gdy widzą duże zwierze czworonożne (koń, sarna itp.), bo nie znają innych nazw podobnych zwierząt. Dopiero, gdy poznają nowe słowo („sarna") zaczynają słowu „koń" przypisywać bardziej szczegółowe znaczenie i odróżniać oba pojęcia.

Problem ten jest zaadresowany w autorskim modelu symulacji językowej, przedstawionym w pracy. Struktura słownika (mapy, którą posługuje się każdy agent) została zaczerpnięta

[1] por. [2] – jedna z pierwszy prace Luca Steelsa na ten temat.

[2] ang. Artificial General Inteligence – konferencje w tej tematyce organizowane są od 2008, strona internetowa konferencji: http://agi-conf.org

[3] ang. referential uncertainity – więcej można przeczytać w [17] oraz o próbie radzenia sobie z tym problem w [4].

z pracy Wellensa, Loetzscha i Steelsa [1]. Definicja pojęcia bazuje na tej z prac Wellensa, jednak jest bardziej spójna z definicją zbioru rozmytego. Model eksperymentu został zbudowany łącząc zalety dwóch modeli opisanych w pracach Steelsa [2],[8],[9] oraz Wellensa [1]. Ogólny przebieg pojedynczej gry językowej jest podobny to tych, o których piszą Stells i Wellens. Metody, którymi posługują się agenty[4] są stworzone od przez autora samodzielnie lub opierając się na rozwiązaniach Steelsa [2], Welensa [1] i innych. Zupełnie nowym elementem symulacji jest wyszukiwanie reduktów. Agenty starają się nie opisywać dokładnie wszystkich cech widzianych obiektów, a jedynie te, które wyróżniają go na tle kontekstu.

Struktura pracy

Pierwsza część pracy (Rozdział 1) opisuje ścisły model matematyczny eksperymentu Steelsa [2]. Model ten budujemy w oparciu podstawowe pojęcia teorii zbiorów przybliżonych stworzonej przez Z.Pawlaka [5]. Rozdział 2 wprowadza fragment teorii zbiorów rozmytych wymagany do zrozumienia nowego modelu który także zdefiniujemy w tej pracy. Rozdział 3 poświęcony jest teorii zbiorów przybliżonych, opisany jest w nim aproksymacyjny algorytm wyszukiwania *reduktów* (pojęcie reduktu zdefiniowane jest w Rozdziale 1 oraz 3). W Rozdziale 4 proponuję nowy model eksperymentu (symulacji językowej) i opisuję opracowane i zaimplementowane algorytmy. Rozdział 5 przedstawia przykłady użycia narzędzia, które jest częścią tej pracy – modułu w języku Python, który umożliwia przeprowadzanie symulacji gier językowych oraz analizę ewolucji języka.

[4] *Agenty* jest to poprawna liczba mnoga od *agent* – słowa które w tej pracy używane jest w znaczeniu autonomicznej jednostki zdolnej do podejmowania określonych akcji.

Rozdział 1

Symulacja językowa

1.1. Definicje podstawowe

Zaczniemy od zdefiniowania *systemu decyzyjnego* oraz *reduktu*.

Definicja 1.1.1 Tabela decyzyjna *jest to struktura* $S = (U, A \cup \{\text{dec}\})$*, gdzie* $U = \{x_1, \ldots, x_n\}$ *jest skończonym zbiorem obiektów,* A *jest skończonym zbiorem atrybutów postaci* $a : U \to \overline{V}_a$*,* $\overline{V}_a = V_a \cup \{*\}$*, gdzie* V_a *jest zbiorem wartości atrybutu,* $*$ *– nieznaną wartością atrybutu, natomiast* $\text{dec}: U \to V_{\text{dec}}$ *jest wyróżnionym atrybutem decyzyjnym,* $\text{dec} \notin A$*.*

Przykładową *tabelę decyzyjną* reprezentującą system decyzyjny przedstawia Tabela 1.1.

obiekt	a_1 – rozmiar	a_2 – kolor	a_3 – kształt	dec – *decyzja*
x_1	$*$	niebieski	kwadratowy	0
x_2	duży	czerwony	kwadratowy	1
x_3	duży	niebieski	$*$	0
x_4	duży	czerwony	trójkątny	0

Tabela 1.1: Tabela decyzyjna reprezentująca system decyzyjny $S = (U, A \cup \{\text{dec}\})$, gdzie $U = \{x_1, x_2, x_3, x_4\}$, $A = \{a_1, a_2, a_3\}$, $a_1(x_1) = *$, $a_2(x_1) =$'niebieski', $a_3(x_1) =$'kwadratowy', $\text{dec}(x_1) = 0$, itd.

Definicja 1.1.2 *Dla danego systemu decyzyjnego* $S = (U, A \cup \{\text{dec}\})$ *oraz obiektów* $x, y \in U$ *mówimy, że* x*,* y *są rozróżnialne przez* $B \subseteq A$ *wtw, gdy* $\exists a \in B\ a(x) \neq a(y),\ a(x) \neq *$ *oraz* $a(y) \neq *$*.*

Definicja 1.1.3 *Zbiór atrybutów* $B \subseteq A$ *nazywamy* reduktem *systemu decyzyjnego* $S = (U, A \cup \{\text{dec}\})$ *wtw, gdy zachodzą warunki:*

- B zachowuje rozróżnialność *zbioru* U *względem decyzji* dec*, tj.* $(\forall x, y \in U\ \text{dec}(x) \neq \text{dec}(y)\ \wedge\ x, y$ *są rozróżnialne przez* $A) \Rightarrow x, y$ *są rozróżnialne przez* B*.*
- B *jest* nieredukowalny *(tj. żaden właściwy podzbiór* $B' \subsetneq B$ *nie zachowuje rozróżnialności zbioru* U*.*

Zbiór wszystkich reduktów systemu decyzyjnego S będziemy oznaczać przez $RED(S)$. Reduktem systemu decyzyjnego S reprezentowanego za pomocą Tabeli 1.1 jest

$$B = \{a_2, a_3\},$$

ponieważ każda para obiektów o różnych decyzjach rozróżnialna przez A jest rozróżnialna przez któryś z atrybutów z $\{a_2, a_3\}$.

Definicja 1.1.4 *Dla podzbioru atrybutów $B \subseteq A$ oraz obiektu $x \in U$ zbiór*

$$inf_B(x) = \{(a, a(x)) : a \in B\} \tag{1.1}$$

nazywamy B-sygnaturą *obiektu x.*

Na przykład B-sygnatura obiektu x_1 z Tabeli 1.1, gdzie $B = \{a_2, a_3\}$ ma postać

$$inf_B(x_1) = \{(a_2, a_2(x)), (a_3, a_3(x))\} = \{(\text{kolor, niebieski}), (\text{kształt, kwadratowy})\}.$$

Dalej pary $(a, v) \in A \times V_a$ będziemy nazywać *cechami (deskryptorami).*

Definicja 1.1.5 *Dla systemu decyzyjnego $S = (U, A \cup \{\text{dec}\})$ zbiór*

$$\mathcal{F}_S = \mathcal{P}(\bigcup_{a \in A} \bigcup_{v \in V_a} (a, v)) \tag{1.2}$$

nazywamy zbiorem potęgowym wszystkich cech.[1]

Definicja 1.1.6 *Dla $U_0 \subseteq U$ oraz $x \in U_0$ mówimy, że $D^* \in \mathcal{F}_S$ jest* zbiorem rozróżniającym[2] *x od $U_0 \setminus \{x\}$ wtw, gdy*

$$D^* \subseteq inf_A(x) \text{ oraz } \forall y \in U_0 \setminus \{x\} \; D^* \nsubseteq inf_A(y). \tag{1.3}$$

A-sygnatura oraz zbiór rozróżniający są zbiorami cech. Redukt, w odróżnieniu od zbioru rozróżniającego, jest zbiorem atrybutów. Zachodzi natomiast następujący fakt.

Fakt 1.1.1 *Dla obiektu $x_0 \in U$ oraz reduktu B_0 systemu decyzyjnego $S = (U, A \cup \{\text{dec}_{x_0}\})$, gdzie funkcja $\text{dec}_{x_0} : U \to \{0, 1\}$ jest definiowana wzorem*

$$\text{dec}_{x_0}(x) = \begin{cases} 1 & \text{dla } x = x_0, \\ 0 & \text{dla } x \in U \setminus \{x_0\}, \end{cases} \tag{1.4}$$

B_0-sygnatura $inf_{B_0}(x_0)$ jest minimalnym, pod względem zawierania, zbiorem rozróżniającym x_0 od $U \setminus \{x_0\}$.

□

1.2. Leksykon

Niech $R \subset W \times \mathcal{F}$ będzie dwuargumentową relacją. Przez

$$Dom(R) = \{w : \exists F \; (w, F) \in R\}, \tag{1.5}$$
$$Im(R) = \{F : \exists w \; (w, F) \in R\} \tag{1.6}$$

oznaczmy dziedzinę i przeciwdziedzinę relacji R.

[1] Dopuszczamy również „sprzeczne" zbiory cech, np. $F = \{(\text{kolor, niebieski}), (\text{kolor, czerwony})\} \in \mathcal{F}_S$.
[2] ang. *distinctive feature set* – por. Steels [2].

Funkcje, których postać zostanie sprecyzowana dalej,

$$\mathrm{freq}_R : R \to \mathbb{N} \setminus \{0\}, \tag{1.7}$$

będziemy nazywać odpowiednio *R-częstością*,

$$\mathrm{succ}_R : R \to \mathbb{N} \cup \{0\}, \tag{1.8}$$

R-skutecznością.

Definicja 1.2.1 *Dla systemu decyzyjnego S (Def. 1.1.1) relację* $L \subset \mathbb{W} \times \mathcal{F}_S$*, gdzie* $\mathcal{F}_S$ *jest zdefiniowane formułą 1.2, a* $\mathbb{W} = \Sigma^*$ *jest zbiorem słów nad alfabetem* Σ*, nazywamy* słownikiem (leksykonem, mapą słowo $\leftrightarrow$ znaczenie)*.*

Element $(w, F) \in L$ będziemy nazywać *wpisem*, w nazywamy *słowem*, F – *znaczeniem* słowa w.

Przykład słownika oraz funkcji częstości i skuteczności przedstawia Tabela 1.2.

Słowo	Znaczenie	freq_L	succ_L
aab	$F_1 = \{$(rozmiar, duży), (kolor, czerwony)$\}$	1	0
aab	$F_2 = \{$(kształt, trójkątny)$\}$	2	1
bb	$F_3 = \{$(kolor, niebieski)$\}$	10	2
cca	$F_3 = \{$(kolor, niebieski)$\}$	5	4
cca	$F_4 = \{$(kształt, trójkątny), (rozmiar, duży)$\}$	3	1

Tabela 1.2: Przykładowy słownik (leksykon) $L \subset \mathbb{W} \times \mathcal{F}_S$ ($\mathbb{W} = \Sigma^*$, $\Sigma = \{a, b, c\}$, $\mathcal{F}_S \supseteq \{F_1, F_2, F_3, F_4\}$) wraz z funkcjami L-częstości freq_L i L-skuteczności succ_L.

Definicja 1.2.2 *Dla danego systemu decyzyjnego S, słownika* $L \subseteq \mathbb{W} \times \mathcal{F}_S$ *oraz* freq_L *i* succ_L *definiujemy funkcję* L-oceny $\lambda_L(L_0)$ *następującym wzorem*

$$\lambda_L(L_0) = \sum_{(w,F)\in L_0} \frac{\mathrm{succ}_L|_{L_0}(w, F)}{\mathrm{freq}_L|_{L_0}(w, F)}, \tag{1.9}$$

gdzie $L_0 \subseteq L$*.*

Na przykład, dla relacji $L_0 = L$ oraz freq_L i succ_L zdefiniowanych Tabelą 1.2 L-ocena L_0 wynosi

$$\begin{aligned}\lambda_L(L_0) &= \sum_{(w,F)\in\{(aab,F_1),(aab,F_2),(bb,F_3),(cca,F_3),(cca,F_4)\}} \frac{\mathrm{succ}_L(w, F)}{\mathrm{freq}_L(w, F)} \\ &= 0 + \frac{1}{2} + \frac{1}{5} + \frac{4}{5} + \frac{1}{3} = \frac{11}{6}.\end{aligned}$$

Zdefiniujemy teraz funkcje *kodującą*, *interpretującą* i *dekodującą*.

Definicja 1.2.3 *Dla danego słownika* $L \subseteq \mathbb{W} \times \mathcal{F}_S$ *funkcję* $\mathrm{kod}_L : \mathcal{F}_S \to \mathcal{P}(\mathcal{P}(\mathbb{W} \times \mathbb{R}))$ *nazywamy* funkcją L-kodującą *i definiujemy formułą*

$$\mathrm{kod}_L(F) = \{L_0 : L_0 \subseteq L \ \wedge \bigcup Im(L_0) = F\}. \tag{1.10}$$

Wartość funkcji $\mathrm{kod}_L(F)$ *nazywamy* rodziną relacji opisujących F*.*

Dla słownika L funkcja L-kodująca zwraca *rodzinę relacji opisujących*, dany zbiór cech $F \in \mathcal{F}_S$. Na przykład, dla $F = \{$(kolor, niebieski), (kształt, trójkątny)$\}$ oraz słownika L zdefiniowanego Tabelą 1.2 funkcja L-kodująca zwraca

$$\begin{aligned}\mathrm{kod}_L(F) &= \mathrm{kod}_L(F_2 \cup F_3) = \\ &= \{\{(aab, F_2), (bb, F_3)\},\ \{(aab, F_2), (cca, F_3)\},\ \{(aab, F_2), (bb, F_3), (cca, F_3)\}\}.\end{aligned}$$

Jeśli $F \notin Im(L)$, wtedy $\mathrm{kod}_L(F) = \emptyset$. Na przykład, dla słownika L jak powyżej $kod_L(\{(ksztat, kwadratowy)\}) = \emptyset$. Mówimy wtedy, że zbiór cech $F \in \mathcal{F}_S$, $F \notin Im(L)$ jest *niewyrażalny* za pomocą słownika L.

Definicja 1.2.4 *Dla słownika $L \subset \mathbb{W} \times \mathcal{F}_S$* funkcją L-interpretującą *nazywamy funkcję $Int_L : Dom(L) \to Im(L)$ spełniającą warunek:*

$$\forall w \in Dom(L) \quad (w, Int_L(w)) \in L. \tag{1.11}$$

Funkcja L-interpretująca danemu słowu $w \in Dom(L) \subseteq \mathbb{W}$ przypisuje jedno z jego znaczeń $F \in Im(L) \subseteq \mathcal{F}_S$, $(w, F) \in L$. Na przykład, dla słownika L zadanego Tabelą 1.2 funkcja zdefiniowana następująco

$$\begin{aligned}Int_L(\text{aab}) &= \{\text{(kształt, trójkątny)}\},\\ Int_L(\text{bb}) &= \{\text{(kolor, niebieski)}\},\\ Int_L(\text{cca}) &= \{\text{(kształt, trójkątny), (rozmiar, duży)}\}\end{aligned}$$

jest funkcją L-interpretującą.

Definicja 1.2.5 *Dla słownika $L \subseteq \mathbb{W} \times \mathcal{F}_S$ funkcja* L-dekodująca $\mathrm{dekod}_L : \mathcal{P}(\mathbb{W}) \to \mathcal{P}(\mathcal{F})$ *określona jest formułą*

$$\mathrm{dekod}_L(W) = \{\bigcup Int_L(Dom(L) \cap W) : Int_L\}. \tag{1.12}$$

Funkcja L-dekodująca zwraca rodzinę tych wszystkich zbiorów cech, które są poprawnymi (wg słownika L) interpretacjami zadanej wypowiedzi $W \subseteq \mathbb{W}$. Na przykład, dla słownika zdefiniowanego Tabelą 1.2

$$\begin{aligned}\mathrm{dekod}_L(\{\text{aab, bb}\}) &= \{F_1 \cup F_3,\ F_2 \cup F_3\}\\ &= \{\{\text{(rozmiar, duży), (kolor, czerwony), (kolor, niebieski)}\},\\ &\quad \{\text{(kształt, trójkątny), (kolor, niebieski)}\}\} \subseteq \mathcal{F}.\end{aligned}$$

Jeśli $W \cap Dom(L) = \emptyset$, $W \subseteq \mathbb{W}$, wtedy $\mathrm{dekod}_L(W) = \emptyset$. Na przykład, dla słownika L zdefiniowanego jak wyżej, $\mathrm{dekod}_L(\{ccc, cac\}) = \emptyset$. Mówimy wtedy, że zbiór słów W jest *nieinterpretowalny* za pomocą słownika L. Zauważmy przy tym, że jeśli $W \cap Dom(L) \neq \emptyset$ oraz $W \not\subseteq Dom(L)$, wtedy $\mathrm{dekod}_L(W) = \mathrm{dekod}(W \cap Dom(L))$, np. $\mathrm{dekod}_L(\{\text{aab, bb, ccc}\}) = \mathrm{dekod}_L(\{\text{aab, bb}\})$.

1.3. Agent. Gra językowa

Niech $\mathcal{A} = \{\alpha_1, \ldots, \alpha_m\}$ będzie zbiorem *agentów*[3]. Agenty działają autonomicznie. Zadaniem każdego agenta $\alpha \in \mathcal{A}$ jest zbudowanie własnego słownika $L_\alpha \subset \mathbb{W} \times \mathcal{F}_S$ (Def. 1.2.1), za pomocą którego będzie mógł opisać dowolny obiekt $x \in U$ z przestrzeni obiektów U (Def. 1.1.1),

[3]agent – autonomiczna jednostka, zdolna do podejmowania określonych działań. Więcej można preczytać z książce Ferbera [18].

przy czym $\mathbb{W} = \Sigma^*$, gdzie Σ jest skończonym alfabetem. U oraz Σ są wspólne dla wszystkich $\alpha \in \mathcal{A}$.

Definicja 1.3.1 Grą językową (n-tą grą językową) *nazwiemy piątkę* $\Phi_n = \langle n, \alpha_i, \alpha_r, U_n, x_0 \rangle$, *gdzie* $n \in \mathbb{N}$ *jest* numerem porządkowym *gry,* $\alpha_i, \alpha_r \in \mathcal{A}$ *są agentami biorącymi udział w grze* (graczami)*; agenta* α_i *nazywamy* nadawcą (inicjatorem), α_r – odbiorcą*;* $U_n \subseteq U$ *jest zbiorem obiektów tworzących tzw.* kontekst *gry, a* $x_0 \in U_n$ *jest tzw.* tematem *gry – obiektem wyróżnionym z kontekstu.*

Z grą Φ_n kojarzymy system decyzyjny

$$S_n = (U_n, A \cup \{\mathrm{dec}_{x_0}|_{U_n}\}), \tag{1.13}$$

gdzie dec_{x_0} jest zdefiniowana wzorem (1.4).

Przez $L_\alpha(n)$ oznaczamy stan słownika L_α agenta $\alpha \in \mathcal{A}$ tuż przed rozpoczęciem n-tej gry oraz przez $\mathrm{freq}_{L_\alpha(n)}$, $\mathrm{succ}_{L_\alpha(n)}$ funkcje L_α-częstości, L_α-skuteczności zdefiniowane dla stanu słownika słownika $L_\alpha = L_\alpha(n)$.

Implementujemy trzy mechanizmy:

1. *Agenty adoptują powiązania pomiędzy słowem a jego znaczeniem od innych agentów. Dzięki temu słowa wymyślone przez każdego pojedynczego agenta mogą być propagowane na całą populację.* Tzn. jeśli agent $\alpha_0 \in \mathcal{A}$ posiada w swoim słowniku wpis $(\mathring{w}, F)$ (tj. $(\mathring{w}, F) \in L_{\alpha_0}$), wtedy podczas *gry językowej* (def. poniżej) wpis ten może zostać dodany do słownika innego agenta. Dzięki temu, po serii wielu gier, wpis ten może znaleźć się w słownikach pozostałych agentów $\alpha \in \mathcal{A}\backslash\{\alpha_0\}$ (por. **Opis symulacji**, punkt 9(b) poniżej).

2. *Agent może wygenerować nowe słowo i powiązać je z nieopisanym jeszcze zbiorem cech.* Tzn, agent $\alpha \in \mathcal{A}$ może utworzyć słowo $\widetilde{w} \in \mathbb{W}$, $w \notin Im(L_\alpha)$ (np. $\widetilde{w}$ ='abc') i nadać mu znaczenie $F \in \mathcal{F}_S$, $F \notin Im(L_\alpha)$ (np. F ={(kolor, czerwony), (kształt, trójkątny)}), tj. dodać parę $(\widetilde{w}, F)$ do swojego słownika (**Opis symulacji**, punkt 9).

3. *Mamy mechanizm sprzęgający wybór danego słowa z liczbą konwersacji zakończonych sukcesem, w których to słowo zostało użyte.* Tzn. jeśli agent $\alpha \in \mathcal{A}$ użyje wpisu $(w_0, F_0) \in L_\alpha$ do budowy wypowiedzi w danej grze (**Opis symulacji**, punkt 5), wtedy po zakończeniu tej gry L_α-częstość tego wpisu $\mathrm{freq}_{L_\alpha}(w_0, F_0)$ zostanie zwiększona o jeden. Ponadto jeśli ta gra zakończy się *sukcesem*, wtedy jeśli był to wpis *użyteczny*, to również L_α-skuteczność tego wpisu $\mathrm{succ}_{L_\alpha}(w_0, F_0)$ zostanie zwiększona o jeden (por. **Opis symulacji**, punkt 8).

   Agent α budując wypowiedź, preferuje wpisy o większym stosunku L_α-skuteczności do L_α-częstości (**Opis symulacji**, punkt 5, formuła (1.15)). Zatem w czasie serii gier *skuteczne* wpisy będą zdobywały przewagę nad nieskutecznymi.

Przebieg gry językowej $\Phi_n = \langle n, \alpha_i, \alpha_r, U_n, x_0 \rangle$ przedstawiają następujące kroki:

- Nadawca wskazuje temat i buduje wypowiedź opisującą wskazany temat. Jeśli nie ma odpowiednich słów, by opisać temat, może utworzyć nowe słowo i dodać je do swojego słownika. Działania te przedstawiają punkty 3, 4, 5 (tworzenie wypowiedzi) oraz 9(a) (tworzenie nowych słów) **Opisu symulacji** poniżej.

- Odbiorca, widząc temat oraz kontekst, interpretuje wypowiedź (punkt 6). Przy czym jeśli nie zna któregoś ze słów wypowiedzi, próbuje wydedukować jego znaczenie z kontekstu (punkt 9(b)).

- W zależności sukcesu w komunikacji (punkt 7), gracze modyfikują częstości i użyteczności wpisów w słownikach (punkt 8).

Eksperyment (symulacja) polega na przeprowadzeniu serii gier $(\Phi_n)_{n=1,\ldots,N}$ pomiędzy różnymi graczami $\alpha \in \mathcal{A}$.

1.4. Opis symulacji

Przed pierwszą grą (dla $n = 1$) słowniki wszystkich agentów są puste, tzn.

$$\forall \alpha \in \mathcal{A} \;\; L_\alpha(1) = \emptyset.$$

Natomiast funkcje L-częstości i L-skuteczności są na początku nieokreślone.

Jeśli nie napiszemy inaczej, to zakładamy, że słowniki oraz funkcje skuteczności zachowują swój stan pomiędzy grami n i $n+1$, tj.

$$L_\alpha(n+1) = L_\alpha(n),\ \mathrm{freq}_{L_\alpha(n+1)} = \mathrm{freq}_{L_\alpha(n)},\ \mathrm{succ}_{L_\alpha(n+1)} = \mathrm{succ}_{L_\alpha(n)}\ \forall \alpha \in \mathcal{A}.$$

Dla $n = 1, \ldots, N$, gdzie N jest liczbą iteracji, symulacja przebiega następująco:

1. Losujemy: graczy $\alpha_i, \alpha_r \in \mathcal{A}$, kontekst $U_n \subseteq U$ oraz temat $x_0 \in U_n$. Wylosowane elementy są wejściem dla gry $\Phi_n = \langle n, \alpha_i, \alpha_r, U_n, x_0 \rangle$.

2. Obliczamy zbiór reduktów $\mathcal{B} = RED(S_n)$, gdzie S_n jest zdefiniowane wzorem (1.13). Jeśli $\mathcal{B} = \emptyset$ to znaczy, że nie da się odróżnić tematu od kontekstu. W takim wypadku przerywamy rozgrywkę n i przechodzimy do rozgrywki $n+1$.

3. *Nadawca* α_i wybiera dowolny najkrótszy redukt $B_0 \in \mathcal{B}$ (tj. $|B_0| \leqslant |B|\ \forall B \in \mathcal{B}$) oraz oblicza zbiór rozróżniający temat od kontekstu (por. Fakt 1.1.1)

$$B^* = inf_{B_0}(x_0).$$

4. Jeśli $L_{\alpha_i} = L_{\alpha_i}(n) \neq \emptyset$, nadawca oblicza *rodzinę relacji opisujących* B^*

$$\mathcal{R} = \mathrm{kod}_{L_{\alpha_i}}(B^*), \tag{1.14}$$

   w przeciwnym wypadku $\mathcal{R} = \emptyset$.

   Rodzina składa się z podzbiorów słownika nadawcy L_{α_i} i posłuży mu do konstrukcji wypowiedzi.

5. Jeśli $\mathcal{R} \neq \emptyset$, nadawca tworzy *wypowiedź* $W_0 \subseteq Dom(L_{\alpha_i}(n))$ w następujących krokach:

   - by wypowiedź była jak najkrótsza, dokonuje selekcji pewnej rodziny relacji minimalnych $\mathcal{R}_{\min} \subseteq \mathcal{R}$:

$$R_{\min} \in \mathcal{R}_{\min} \;\Leftrightarrow\; R_{\min} \in \mathcal{R} \wedge (\forall R \in \mathcal{R} \;\; |R_{\min}| \leqslant |R|),$$

- aby wypowiedź zawierała słowa o jak największym współczynniku *skuteczności* do *częstości* (por. Zaimplementowane mechanizmy, punkt 3 na str. 11), dla $L_{\alpha_i} = L_{\alpha_i}(n)$ wybiera relację $R_0 \in \mathcal{R}_{\min}$ ($R_0 \subseteq L_{\alpha_i}$) z maksymalną L_{α_i}-oceną (Def. 1.2.2), tj. taką, że

$$\lambda_{L_{\alpha_i}}(R_0) \geqslant \lambda_{L_{\alpha_i}}(R) \text{ dla każdego } R \in \mathcal{R}, \tag{1.15}$$

- wypowiedź $W_0 = Dom(R_0)$.

W przeciwnym wypadku (gdy $\mathcal{R} = \emptyset$) $W_0 = \emptyset$. Oznacza to, że nadawca nie potrafił zbudować wypowiedzi, ponieważ nie miał odpowiednich wpisów w słowniku.

6. Jeśli $L_{\alpha_r} = L_{\alpha_r}(n) \neq \emptyset$, *odbiorca* α_r *dekoduje* wypowiedź

$$\mathcal{F}_0 = \text{dekod}_{L_{\alpha_r}}(W_0),$$

w przeciwnym wypadku $\mathcal{F}_0 = \emptyset$.

7. Gra G kończy się:

   - *sukcesem*, jeśli
   $$\mathcal{F}_0 \cap \mathcal{B}^* \neq \emptyset. \tag{1.16}$$
   Oznacza to, że nadawca i odbiorca byli wstanie się ze sobą porozumieć, tj. rodzina zbiorów cech $\mathcal{F}_0$ odkodowanych przez odbiorcę zawiera któreś ze zbiorów rozróżniających temat od kontekstu $B^* \in \mathcal{B}^*$, gdzie
   $$\mathcal{B}^* = \{inf_B(x_0) : B \in \mathcal{B}\} \text{ (por. Def. 1.1.4).} \tag{1.17}$$

   - *porażką*, jeśli
   $$\mathcal{F}_0 \cap \mathcal{B}^* = \emptyset. \tag{1.18}$$

   W tym wypadku, jeśli $\mathcal{F}_0 = \emptyset$, to znaczy, że odbiorca nie potrafił odkodować znaczenia słów, ponieważ nie miał odpowiednich wpisów w swoim słowniku L_{α_r}. Jeśli natomiast $\mathcal{F}_0 \neq \emptyset$, ale $\mathcal{F}_0 \cap \mathcal{B}^* = \emptyset$, mamy niezgodność znaczeń, tj. odbiorca błędnie zinterpretował wypowiedź nadawcy.

8. Gracze α_i, α_r aktualizują funkcje L_{α_i}-, L_{α_r}-częstości oraz L_{α_i}-, L_{α_r}-skuteczności (tzn. obliczają ich wartości dla $n+1$), a mianowicie:

   - Jeśli $W_0 \neq \emptyset$, *nadawca* α_i zwiększa o jeden L_{α_i}-częstość *wpisów* $(w, F) \in \mathcal{R}_f^+$ użytych do konstrukcji (*kodowania*) wypowiedzi W_0 oraz zwiększa o jeden L_{α_i}-skuteczność wpisów *użytecznych* $(w, F) \in \mathcal{R}_s^+$, tzn. dla $\alpha = \alpha_i$ oraz

   $$R_f^+ = R_0 \qquad (R_0 \text{ zdefiniowane wzorem (1.15) w punkcie 5}),$$
   $$R_s^+ = \{(w, F) \in R_0 : F \in \mathcal{F}_0 \cap \mathcal{B}^*\} \qquad (\mathcal{B}^* \text{ zdefiniowane formułą (1.17) w punkcie 7}),$$

   funkcje $\text{freq}_{L_\alpha(n+1)}$, $\text{succ}_{L_\alpha(n+1)}$ zdefiniowane są następująco:

$$\text{freq}_{L_\alpha(n+1)}(w, F) = \begin{cases} \text{freq}_{L_\alpha(n)}(w, F) + 1 & \text{dla } (w, F) \in R_f^+, \\ \text{freq}_{L_\alpha(n)}(w, F) & \text{dla } (w, F) \in L_\alpha(n) \setminus R_f^+, \end{cases} \tag{1.19}$$

$$\text{succ}_{L_\alpha(n+1)}(w, F) = \begin{cases} \text{succ}_{L_\alpha(n)}(w, F) + 1 & \text{dla } (w, F) \in R_s^+, \\ \text{succ}_{L_\alpha(n)}(w, F) & \text{dla } (w, F) \in L_\alpha(n) \setminus R_s^+. \end{cases} \tag{1.20}$$

- Jeśli $\mathcal{F}_0 \neq \emptyset$ *odbiorca* α_r zwiększa L-częstość i L-skuteczność *wpisów* użytych w *dekodowaniu* wypowiedzi W_0, tj. dla

$$R_f^+ = \{(w, F) \in L_\alpha(n) : F \in \mathcal{F}_0\},$$
$$R_s^+ = \{(w, F) \in L_\alpha(n) : F \in \mathcal{F}_0 \cap \mathcal{B}^*\},$$

gdzie $\alpha = \alpha_r$, funkcje $\mathrm{freq}_{L_\alpha(n+1)}$, $\mathrm{succ}_{L_\alpha(n+1)}$ definiuje odpowiednio formułami (1.19), (1.20).

9. W przypadku porażki (warunek (1.18) w punkcie 7) gracze α_r, α_r modyfikują słowniki, oraz funkcje częstości i skuteczności.

   (a) Jeśli $W_0 = \emptyset$ (por. punkt 5), to znaczy, że nadawca α_i nie miał odpowiednich *wpisów* w słowniku by *zakodować* wypowiedź. Wtedy z ustalonym małym prawdopodobieństwem[4] p gracz może wymyślić nowe słowo i dodać je do swojego słownika, tzn. dla $\alpha = \alpha_i$ i każdego $F \in \mathcal{B}^*$ dodać *wpis*:

$$L_\alpha(n+1) = L_\alpha(n) \cup \{(\widetilde{w}, F)\},$$
$$\mathrm{freq}_{L_\alpha(n+1)}(\widetilde{w}, F) = 1,$$
$$\mathrm{succ}_{L_\alpha(n+1)}(\widetilde{w}, F) = 0.$$

   gdzie $\widetilde{w} \in \mathbb{W}$, $\widetilde{w} \notin Dom(L_\alpha(n))$ – nowe losowe słowo, $\mathbb{W}$ jak w podrozdziale 1.3.

   (b) Gdy $W_0 \neq \emptyset$, ale $\mathcal{F}_0 \cap \mathcal{B}^* = \emptyset$ (por. punkt 6), to znaczy, że odbiorca nie potrafił poprawnie *zdekodować* wypowiedzi. Jeśli W_0 zawiera słowa nieznane dla odbiorcy

$$\mathring{W} = W_0 \setminus Dom(L_{\alpha_r}(n)) \neq \emptyset \tag{1.21}$$

   oraz jeśli

$$|\mathring{W}| = 1, \tag{1.22}$$

   odbiorca może wydedukować znaczenie nieznanego słowa $\mathring{w} \in \mathring{W}$ z kontekstu, tzn. dla $\alpha = \alpha_r$ i każdego $F \in \mathcal{B}^* \setminus \mathcal{F}_0$ może dodać do swojego słownika *wpis*:

$$L_\alpha(n+1) = L_\alpha(n) \cup \{(\mathring{w}, F)\},$$
$$\mathrm{freq}_{L_\alpha(n+1)}(\mathring{w}, F) = 1,$$
$$\mathrm{succ}_{L_\alpha(n+1)}(\mathring{w}, F) = 0.$$

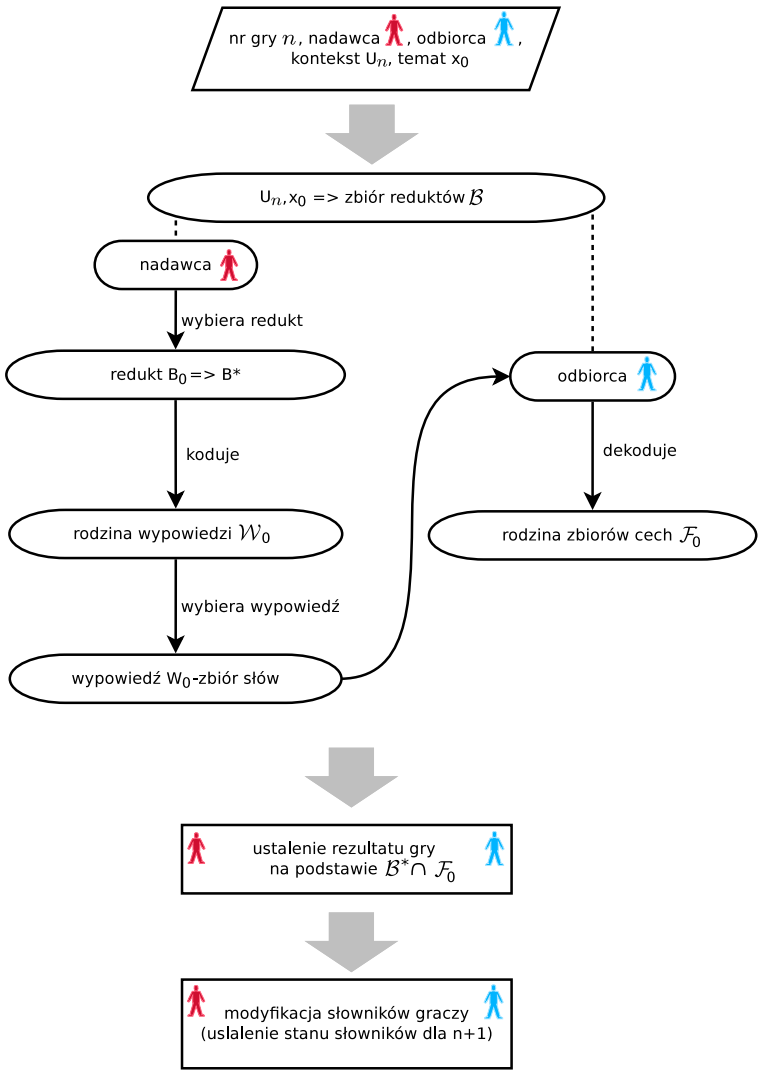
   Jeśli natomiast $|\mathring{W}| > 1$, to odbiorca uznaje, że w wypowiedzi W_0 jest za wiele słów niezrozumiałych i nie dodaje nowych wpisów do słownika.

Przebieg n-tej gry przedstawia Rysunek 1.1.

1.5. Podsumowanie modelu

Matematyczny model symulacji językowej, który sformułowaliśmy w poprzednim podrozdziale, pozwala na zaobserwowanie zjawisk, widocznych również w języku naturalnym. Może to stanowić jedną z motywacji do dalszego badania tego modelu. Wspomniane zachowania obserwowane w języku naturalnym obejmują:

[4] Prawdopodobieństwo stworzenia nowego słowa przez gracza jest jednym z parametrów symulacji zadanym na początku.



Rysunek 1.1: Schemat n-tej gry językowej $\Phi_n = \langle n, \alpha_i, \alpha_r, U_n, x_0 \rangle$.

1. *Powstawanie wyrazów wieloznacznych.* Przyczyny tego możemy doszukać się analizując punkt 9 opisu symulacji z poprzedniego podrozdziału 1.4. Zacznijmy od nadawcy, czyli punktu 9(a). Wystarczy zauważyć, że jeśli $\mathcal{B}^*$ (por. punkt 6, str. 13) zawiera więcej niż jeden zbiór cech rozróżniający *temat* od *kontekstu*, którego nadawca nie potrafi opisać, tj. $|\mathcal{B}^* \setminus Im(L_{\alpha_i})| \geqslant 2$ (czyli, gdy w punkcie 9(b) $W_0 = \emptyset \wedge |\mathcal{B}^*| \geqslant 2$), wtedy do słownika może zostać dodanych kilka wpisów – wszystkie wiążące nowo wymyślone słowo $\widetilde{w} \in \mathbb{W}$ ze zbiorami cech $F_1, F_2 \in \mathcal{B}^*$, $F_1 \neq F_2$. Jeśli poprzez serię gier $(\Phi_n)_{n=1,\ldots,N}$ *wpisy* $(\widetilde{w}, F_1)$, $(\widetilde{w}, F_2)$ rozprzestrzenią się na dużą część populacji agentów $\alpha \in \mathcal{A}$, to słowo $\widetilde{w}$ stanie się *wyrazem wieloznacznym*.

   Podobnie sytuacja wygląda u nadawcy, tzn. jeśli $\mathcal{B}^* \setminus \mathcal{F}_0$ zawiera więcej niż jeden zbiór cech, wszystkie one zostaną powiązane z nowo poznanym słowem $\mathring{w} \in \mathring{W}$ (por. punkt 9(b)).

2. *Powstawanie synonimów.* Przyczynę możemy znaleźć analizując punkt 9(b), a mianowicie odbiorca może powiązać nowo poznane słowo $\mathring{w} \in \mathring{W}$ ze zbiorem cech $F_0 \in \mathcal{F}_0 \cap \mathcal{B}^*$, który znajduje się już w jego słowniku, tj. $\exists w' \in \mathbb{W}$ $(w', F_0) \in L_{\alpha_i}$. Poznaje wtedy

kolejne określenie na zbiór cech F_0. Jeśli poprzez serię gier *wpisy* $(\mathring{w}, F_0)$ i (w', F_0) rozprzestrzenią się na dużą część populacji agentów $\alpha \in \mathcal{A}$, to słowa $\mathring{w}$ i w' stają się *synonimami*.

Jednak bezpośrednie zastosowanie tego modelu w środowisku bardziej złożonym jest problematyczne. Wraz ze wzrostem liczby cech wykładniczo rośnie liczba możliwych definicji słowa w słowniku. Ponadto Steels [2] nie korzystał z definicji reduktów, a posługiwał się rodzinami zbiorów rozróżniających, które mogą być bardzo liczne (tj. wykładniczo dużo w stosunku do liczby cech obiektu), a ich wyznaczanie sprowadza się do sprawdzania wszystkich możliwych podzbiorów cech. Co prawda wyznaczenie reduktu jest problemem NP–trudnym, ale znane są algorytmy aproksymacyjne, które umożliwiają ich szybsze obliczanie. W Rozdziale 3 omawiamy algorytm aproksymacyjny zaproponowany przez A. Skowrona [7].

Kolejną trudnością jest konstrukcja wypowiedzi, czyli działanie funkcji kod_L i dekod_L (Def. 1.2.3 oraz 1.2.5). Implementując je przeglądamy wykładniczą liczbę możliwość. Wiąże się to z innym problemem – reprezentacją słowników. Ponieważ słowo jest definiowane przez zbiór zbiorów cech, trudno jest wyszukiwać w nim odwrotnie, tzn. musimy przeszukiwać mapowania zbiorów cech na słowa.
Podsumowując, musimy poradzić sobie z następującymi problemami:

- obliczanie reduktów (por. Def. 1.1.3),
- obliczanie funkcji kod_L i dekod_L (Def. 1.2.3, 1.2.5),
- reprezentacja słownika (Def. 1.2.1).

W analizowanej grze językowej pominięty jest też jeden aspekt – niepewność referencyjna[5] (o której wspomina Wellens i inni [1]). Istotnie, agent przekazując komunikat wskazuje od razu opisywany obiekt. A ponieważ rodzina zbiorów rozróżniających jest współdzielona przez nadawcę i odbiorcę, to odbiorca wie dokładnie jakich zbiorów cech dotyczy komunikat. Co więcej, cytując za Borewmanem i Choi [12] przykład z „życia", dziecko słysząc nowe słowo nie ma zadanego zbioru znaczeń które może mu przypisać, więc musi bazować na własnych obserwacjach. Z kolei współdzielenie rodziny zbiorów rozróżniających powoduje automatyczne ich narzucenie.

W przeprowadzanym eksperymencie brak jest mechanizmu, który pozwala zaadoptować słowa o niepewnym znaczeniu – jeśli w odebranej wypowiedzi nie znamy definicji co najmniej dwóch słów, nie tworzymy nowych powiązań w swoim słowniku (por. punkt 9(b) w opisie symulacji na str. 14). Zatem w tym przypadku ignorujemy informację jaka do nas dotarła.

Aby uwzględnić powyższe uwagi i ominąć wspomniane problemy, do konstrukcji nowego modelu użyjemy logiki rozmytej oraz teorii zbiorów przybliżonych. Logikę rozmytą wykorzystamy w konstrukcji słowników, a teorię zbiorów przybliżonych przy szybkim znajdowaniu reduktów. W Rozdziale 4 przedstawimy model zbudowany na bazie właśnie opisanego oraz innego zaproponowanego przez Wellensa i innych [1].

[5] ang. *referential uncertainity* – termin omówiony we Wstępie.

Rozdział 2

Logika rozmyta[1]

2.1. Definicje

Definicja 2.1.1 Zbiorem rozmytym *nazywamy parę* (A, f_A)*, gdzie* $A \subset X$ *jest zbiorem (*X *jest przestrzenią wszystkich elementów),* $f_A : X \to [0,1]$ *jest funkcją przynależności do zbioru. Wartość funkcji* $f_A(x)$ *nazywamy* stopniem przynależności *elementu* x *do zbioru* A *(ozn.*[2] $x \in A \Leftrightarrow f_A(x) > 0$*).*

Definicja 2.1.2 Nośnik *zbioru rozmytego* A *(ozn.* supp$A$*) jest to „normalny" zbiór elementów* A*, tzn.* supp$A = \{x \in X | f_A(x) > 0\}$.

Podstawowe operacje na zbiorach rozmytych definiuje się, jak poniżej.

Definicja 2.1.3 Przecięcie *zbiorów rozmytych* A *i* B *(ozn.* $A \cap B$*), to zbiór z poniższą funkcją przynależności*

$$f_{A\cap B} \stackrel{df}{=} f_A \wedge f_B, \tag{2.1}$$

gdzie $\wedge$ *oznacza minimum z dwóch wartości.*

Stopień przynależności elementu do przecięcia jest mniejszym ze stopni przynależności do przecinanych zbiorów.
Analogicznie definiujemy sumę zbiorów rozmytych.

Definicja 2.1.4 Suma *zbiorów* A *i* B *(ozn.* $A \cup B$*), to zbiór z następującą funkcją przynależności*

$$f_{A\cup B} \stackrel{df}{=} f_A \vee f_B, \tag{2.2}$$

gdzie $\vee$ *to maksimum.*

Stopień przynależności elementu do sumy zbiorów A i B jest to większy spośród stopni przynależności do każdego ze składników sumy.

Definicja 2.1.5 Dopełnienie A *(ozn.* A'*) definiujemy zadając*

$$f_{A'} \stackrel{df}{=} 1 - f_A. \tag{2.3}$$

Wtedy różnica $A - B = (A \cap B)' \cap A$ *ma funkcję przynależności*

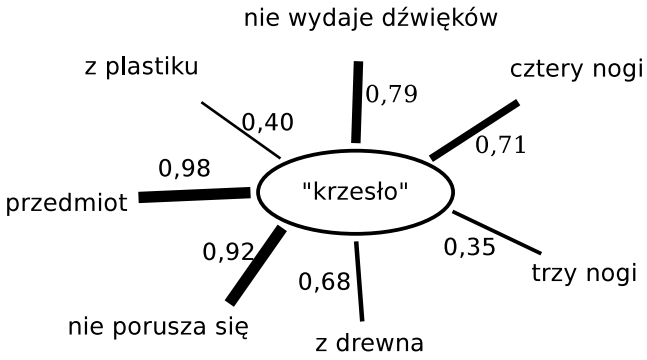
$$f_{A-B} = (1 - (f_A \wedge f_B)) \wedge f_A. \tag{2.4}$$

[1] Więcej informacji można znaleźć w pracy prof. Zadeha [13].

[2] Ogólnie przynależność do zbioru definiuje się, jako przekroczenie pewnego ustalonego progu aktywacji ε, tzn. $x \in A \Leftrightarrow f_A(x) > \varepsilon$. U nas $\varepsilon = 0$.

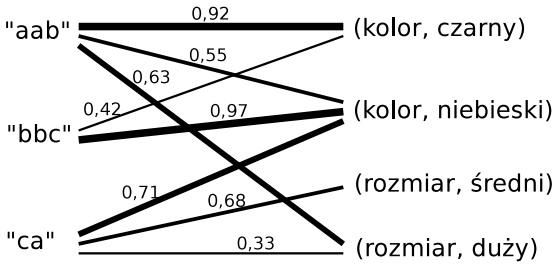
2.2. Koncept

Zbiory rozmyte będziemy interpretować, jako pojęcia – Rysunek 2.1.



Rysunek 2.1: Przykład pojęcia rozmytego. A – "krzesło", $f_A(trzy\ nogi) = 0.35$, $f_A(przedmiot) = 0.98$, itd. Pojęcie to mogłoby się pojawić w słowniku agenta, który obserwując obiekty nazywane "krzesło" widział najczęściej przedmioty zrobione z drewna albo plastiku, które miały najczęściej cztery nogi, nie poruszały się i raczej nie wydawały dźwięków.

Z takich pojęć zbudowane są słowniki graczy w nowym modelu symulacji językowej (por. Rozdział 4). Przykład słownika przedstawia Rysunek 2.2. Są to dwustronne mapy słów na cechy (i odwrotnie – cech na słowa). Każdemu słowu odpowiada zbiór rozmyty cech (i odwrotnie – każdej cesze odpowiada zbiór rozmyty słów).



Rysunek 2.2: Fragment słownika pojęć rozmytych.

Taka reprezentacja słownika ma tę przewagę nad omawianą poprzednio (Rozdz. 1.2), że łatwo jest w nim wyszukiwać w obie strony (jest symetryczny) i łatwiej też jest go reprezentować w pamięci. Jest to graf dwudzielny o z góry ograniczonej przez *system decyzyjny* $S = (U, A \cup \{\text{dec}\})$ (Def. 1.1.1, str. 7) liczbie wierzchołków oznaczających cechy (wierzchołki po prawej stronie na Rys. 2.2). Liczba tych wierzchołków jest nie większa niż $|\mathcal{F}_S|$, gdzie $\mathcal{F}_S$ jest zbiorem potęgowym wszystkich cech systemu decyzyjnego S (Def. 1.1.5).

Rozdział 3

Obliczanie reduktów. Teoria zbiorów przybliżonych[1]

W tym rozdziale przedstawimy zachłanny algorytm[2] wyszukiwania reduktów, zaproponowany przez A. Skowrona i C. Ruszera w [7]. Przedstawimy jego wydajną implementację pochodzącą z pracy H. S. Nguyena [3]. Zaczniemy od zdefiniowania *funkcji rozróżniającej* i przytoczenia równoważnej do podanej wcześniej (por. Def. 1.1.3) definicji reduktu systemu decyzyjnego.

Definicja 3.0.1 *Dla systemu decyzyjnego* $S = (U, A \cup \{\text{dec}\})$ *(Def. 1.1.1) oraz podzbioru atrybutów* $B \subseteq A$ *przez* $INF(B)$ *oznaczamy zbiór* B-sygnatur *(Def. 1.1.4)*

$$INF(B) = \{inf_B(x) : x \in U\}.$$

Definicja 3.0.2 Funkcją rozróżniającą *systemu decyzyjnego* $S = (U, A \cup \{dec\})$ *nazywamy miarę* $disc : \mathcal{P}(A) \to \mathbb{N}$, *która dla danego zbioru atrybutów* $B \subseteq A$, *przypisuje liczbę różnych par obiektów* rozróżnialnych *(Def. 1.1.2) przez ten zbiór, które należą do różnych klas decyzyjnych. Definiujemy ją formułą*

$$disc(B) = \frac{1}{2}|\{(x, y) \in U \times U : x, y \text{ są rozróżnialne przez } B \ \wedge \ dec(x) \neq dec(y)\}|. \tag{3.1}$$

Definicja 3.0.3 *Zbiór* $B \subseteq A$ *nazywamy* reduktem względem funkcji rozróżnialności *disc wtw, gdy* B *jest najmniejszym pod względem zawierania zbiorem atrybutów takim, że* $disc(B) = disc(A)$, *tj.* $disc(B') < disc(B)$ *dla każdego właściwego podzbioru* $B' \subsetneq B$.

Definicja 3.0.4 *Mówimy, że atrybut* a *należy do* rdzenia *systemu decyzyjnego* $S = (U, A \cup \{dec\})$ *wtw, gdy*

$$disc\,(A \setminus \{a\}) < disc(A), \tag{3.2}$$

co oznacza, że a *należy do każdego reduktu tabeli* S.

Algorytm zachłanny znajdujący redukty zaczyna od zbioru atrybutów $B \subseteq A$ z rdzenia systemu decyzyjnego. W kolejnych iteracjach do zbioru B dodawane są atrybuty najbardziej zwiększające rozróżnialność $disc(B)$. Na końcu, gdy rozróżnialność B osiągnie wartość maksymalną, tzn. $disc(B) = disc(A)$ sprawdzamy jeszcze, czy nie dodaliśmy wcześniej nadmiarowego w tym momencie atrybutu. Działanie algorytmu ilustruje poniższy pseudokod.

[1] Więcej można przeczytać w książce Z. Pawlaka [5] – twórcy teorii zbiorów przybliżonych.

[2] Inne algorytmy wyszukiwania reduktów można znaleźć w dokumentacji projektu RSES (Rough Set Exploration System), m.in. w [6].

```
ObliczRedukt(o, U_n)

B ← zbiór atrybutów z rdzenia podsystemu decyzyjnego U_n
A ← zbiór wszystkich atrybutów w U_n

while B ≠ A do
    a_max ← argmax: disc(B ∪ {a}) for a ∈ A\B
    if disc(B ∪ {a_max}) > disc(B) then
        B ← B ∪ {a_max}
    else break

for a in B do
    if disc(B) = disc(B \ {a}) then
        B ← B \ {a}

return B
```

Wartość funkcji $disc(B)$ policzymy ze wzoru 3.3, którego wyprowadzenie można znaleźć w [3].

$$disc(B) = \frac{1}{2}\left(n^2 - \sum_k n_k^2\right) - \frac{1}{2}\sum_{v \in INF(B)}\left[\left(\sum_k n_{v,k}\right)^2 - \sum_k n_{v,k}^2\right], \tag{3.3}$$

gdzie $n_{v,k} = |\{x \in U : inf_B(x) = v \ \wedge \ dec(x) = k\}|$, a $n_k = |CLASS_k| = \sum_v n_{v,k}$ jest rozmiarem k-tej klasy decyzyjnej. Zaznaczmy przy tym, że szukamy reduktów lokalnych podzbioru systemu decyzyjnego $S = (U, A \cup \{dec_{x_0}\})$ – podsystemu $S' = (U', A \cup \{dec_{x_0}\})$ (por. Def. 1.4). Chcemy odróżnić temat $x_0 \in U' \subseteq U$ od pozostałych obiektów kontekstu $U' \setminus \{x_0\}$. Mamy więc dwie klasy decyzyjne $CLASS_1 = \{x_0\}$ oraz $CLASS_2 = U' \setminus \{x_0\}$. Zatem $n_1 = 1$, gdzie $n_2 = n - 1$, $n = |U'|$. Podstawiając do wzoru 3.3 otrzymujemy

$$\begin{aligned} disc(B) =& \frac{1}{2}(n^2 - 1 - (n-1)^2) - \frac{1}{2}\sum_{v \in INF(B)}[(n_{v,1} + n_{v,1})^2 - n_{v,1}^2 - n_{v,2}^2)] && (3.4)\\ =& (n-1) - \sum_{v \in INF(B)} n_{v,1} n_{v,2}. && (3.5) \end{aligned}$$

□

Przedstawiony algorytm może być użyty podczas konstrukcji wypowiedzi (por. metoda `BudujWypowiedź`, podrozdział 4.2.3). Dzięki temu budowanie wypowiedzi jest znacznie mniej kosztowne. Złożoność obliczeniowa jest rzędu $O(mn \log n)$, gdzie $m = |A|$ jest liczbą atrybutów, a $n = |U'|$ jest liczbą obiektów (por. H. S. Nguyen [3]).

Rozdział 4

Nowy model symulacji językowej

Adresując problemy przedstawionej w Rozdziale 1 symulacji językowej (por. rozdz. 1.5) zaproponujemy nowy model ewolucji języka.
W nowym modelu realizacja pojęć, jako zbiory rozmyte cech, oraz reprezentacja słownika – jako dwustronnej mapy jest zaczerpnięta z prac Wellensa i innych[1].

Algorytmy którymi posługują się gracze są wynikiem pracy autora i wywodzą się z algorytmów zaproponowanych w [1]. Główną ich różnicą od tych wspomnianych wcześniej, jest uwzględnianie reduktów cech obiektów opisywanych w trakcie gry.

Została również zdefiniowana inna miara podobieństwa dwóch zbiorów rozmytych, z której korzysta się we wspomnianych wyżej algorytmach.

4.1. Miara podobieństwa

Podczas symulacji będziemy wielokrotnie obliczać podobieństwo słów:

- podczas konstrukcji wypowiedzi. *Nadawca* maksymalizuje podobieństwo (*Similarity*) budowanej wypowiedzi do *tematu*, minimalizując jednocześnie podobieństwo do pozostałych obiektów z *kontekstu* (por. metoda `BudujWypowiedź`, podrozdział 4.2.3).
- podczas interpretacji wypowiedzi. *Odbiorca*, typując temat, wybiera obiekt o największym współczynniku podobieństwa do otrzymanej *wypowiedzi*, spośród obiektów z kontekstu (por. metoda `InterpretujWypowiedź`, podrozdział 4.2.4).

Do obliczeń tego podobieństwa Wellens, Loetzsch i Steels [1] posługują się inną niż zdefiniowana wcześniej, *niesymetryczną* definicją przecięcia dwu zbiorów rozmytych:

Definicja 4.1.1 Niesymetryczne przecięcie *zbiorów A i B (ozn. $A \sqcap B$) to zbiór, którego funkcja przynależności zadana jest wzorem*

$$f_{A\sqcap B}(x) \stackrel{df}{=} \begin{cases} f_A(x) & \text{jeśli } x \in B \\ 0 & \text{jeśli } x \notin B \end{cases} . \tag{4.1}$$

Użyta też jest inna definicja *różnicy* dwóch zbiorów rozmytych.

Definicja 4.1.2 *Różnica zbiorów rozmytych A i B (ozn. $A \setminus B$), to zbiór, którego funkcja przynależności zadana jest wzorem*

$$f_{A\setminus B}(x) \stackrel{df}{=} \begin{cases} f_A(x) & \text{jeśli } x \notin B \\ 0 & \text{jeśli } x \in B \end{cases} . \tag{4.2}$$

Wtedy podobieństwo[1] dwu zbiorów definiujemy w sposób następujący:

Definicja 4.1.3

$$Sim(A,B) = \frac{\sum(A \sqcap B)\sum(B \sqcap A) - \sum(A \setminus B)\sum(B \setminus A)}{\sum(A)\sum(B)}, \tag{4.3}$$

gdzie $\sum(A)$ *oznacza sumę stopni przynależności wszystkich elementów zbioru A, tj.* $\sum(A) = \sum_{x \in X} f_A(x)$ *(por. Def. 2.1.1).*

Podobieństwo dwóch zbiorów rozmytych odpowiada podobieństwu dwóch słów.

Należy jednak zauważyć, że zdefiniowana powyżej miara podobieństwa nie rozróżnia zbiorów składających się z tych samych elementów o **różnym** stopniu przynależności, np. jeśli

$$\begin{aligned} X &= (\{a,b\};\ f_X(a) = 0.1,\ f_X(b) = 0.7) \\ Y &= (\{a,b\};\ f_Y(a) = 0.5,\ f_Y(b) = 0.6), \end{aligned}$$

wtedy

$$Sim(X,Y) = \frac{\sum(X \sqcap Y)\sum(Y \sqcap X) - \sum(X \setminus Y)\sum(Y \setminus X)}{\sum(X)\sum(Y)} = \frac{0.8 \cdot 1.1 - 0 \cdot 0}{0.8 \cdot 1.1} = 1.$$

Dlatego zaproponujemy następującą miarę podobieństwa:

Definicja 4.1.4

$$Similarity(A,B) = \frac{\sum(A \cap B) - \sum(A \setminus B) - \sum(B \setminus A)}{\sum(A \cup B)}. \tag{4.4}$$

Wtedy

$$Similarity(X,Y) = \frac{\sum(X \cap Y) - \sum(X \setminus Y) - \sum(Y \setminus X)}{\sum(X \cup Y)} = \frac{0.1 + 0.6 - 0 - 0}{0.5 + 0.7} = \frac{7}{12}.$$

Mianownik we wzorze 4.4, podobnie jak w 4.3, pełni rolę normującą. Dzięki temu $\forall A, B$: $|Similarity(A,B)| \leqslant 1$.

4.2. Symulacja

4.2.1. Dygresja

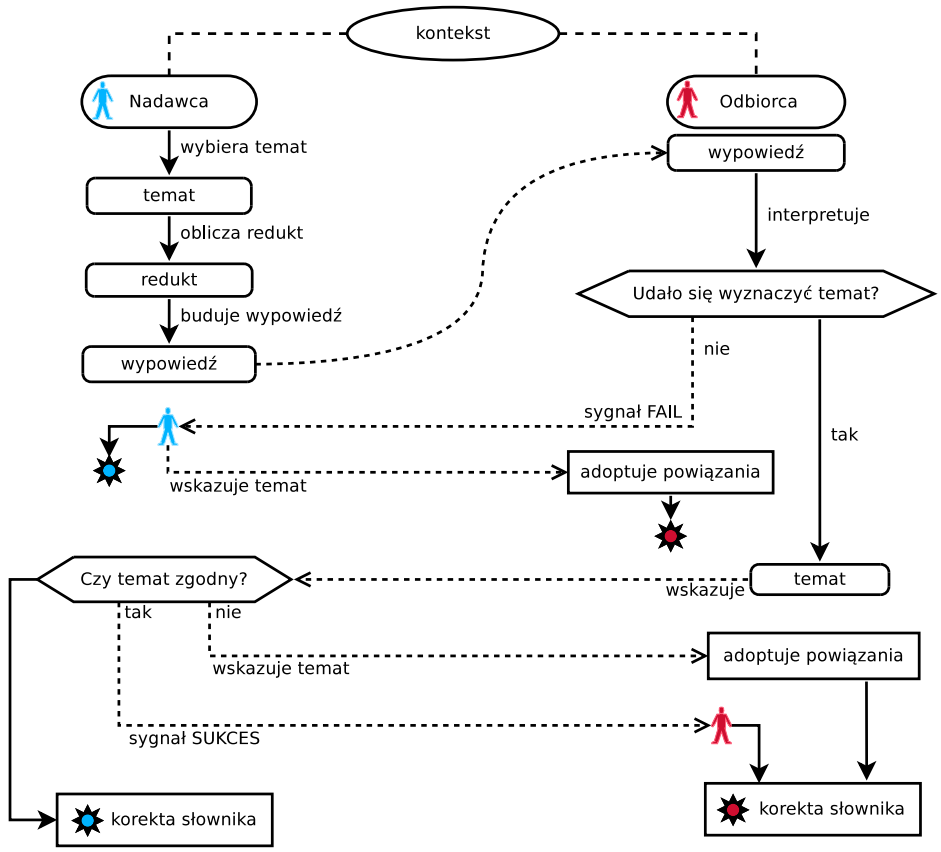
Istotną różnicą w przebiegu nowej symulacji w porównaniu do tej opisanej w Rozdziale 1 jest to, że *nadawca* nie wskazuje na początku opisywanego obiektu, jak to miało miejsce w poprzednim modelu. W związku z tym gracze nie współdzielą *rodziny zbiorów rozróżniających* tematu od kontekstu (por. Rozdz. 1, Def. 1.1.6).

Ponadto agenty, budując wypowiedź, starają się nie dołączać słów, które w danym kontekście nie wnoszą dodatkowej informacji. Na ogół, gdy opisujemy pewną rzecz podajemy jej cechy szczególne, pomijając nieistotne detale. W tym sensie eksperyment symuluje sytuację bardziej naturalną.

[1] Podobieństwo dwóch zbiorów rozmytych można liczyć na wiele sposobów. Więcej na ten temat można przeczytać w [14], [15].

4.2.2. Oznaczenia

W tym rozdziale będziemy stosować zestaw oznaczeń podobny jak poprzednio (Rozdz. 1.3). U będzie zbiorem wszystkich obiektów jakie mogą obserwować agenty. $U_n \subseteq U$ będzie *kontekstem* gry, x_0 – *tematem*. $S_n = (U_n, A \cup \{dex_{x_0}|U_n\})$ – systemem decyzyjnym skojarzonym z grą (por. wzór (1.13), str. 11). Wypowiedź nadawcy jest zbiorem słów, oznaczamy ją przez $W_0 \subseteq \mathbb{W}$, gdzie $\mathbb{W} = \Sigma^*$, Σ jest wspólnym dla wszystkich graczy alfabetem (por. Def. 1.2.1). Przebieg nowej symulacji ilustruje Rysunek 4.1.



Rysunek 4.1: Schemat symulacji. Strzałkami przerywanymi oznaczono interakcje pomiędzy agentami.

Na początku każdej gry gracze współdzielą ze sobą tylko kontekst $U_n \subseteq U$, temat x_0 jest wybierany przez *nadawcę*.
Nadawca wskazuje odbiorcy temat x_0 tylko wtedy, gdy odbiorca nie będzie potrafił poprawnie zinterpretować *wypowiedzi*.

4.2.3. Budowa wypowiedzi

Nadawca oblicza redukt B wyróżniający temat x_0 od kontekstu U_n, tj. redukt systemu decyzyjnego $S_n = (U_n, A \cup \{dex_{x_0}\})$ (metoda `ObliczRedukt`, Rozdział 3), a następnie zbiór rozróżniający $B^* = inf_B(x_o)$ (Def. 1.1.4, str. 8) i tworzy wypowiedź (zbiór słów) W_0, która opisuje zbiór rozróżniający B^* (metoda `BydujWypowiedź` poniżej). Danymi wejściowymi metody budującej są : *temat* – x_0, *kontekst* – U_n i obliczony uprzednio redukt B.

BudujWypowiedź(x_0,U_n,B)

$B^* \leftarrow inf_B(x_0)$
W ← słowa ze słownika będące w relacji z B^*
for w *in* W do
    score(w) ← Similarity(w,x_0) - max(Similarity(w,$U_n \backslash \{x_0\}$))
$W_0 \leftarrow \emptyset$
while W $\neq \emptyset$ do
    v ← argmax: score(w) for w ∈ W
    if score($W_0 \cup$ {v}) > score(W_0) then
        $W_0 \leftarrow W_0 \cup$ {v}
        W ← W \ {v}
    else
        break
return W_0

4.2.4. Interpretacja wypowiedzi

Interpretując wypowiedź odbiorca wybiera, z kontekstu U_n, obiekt najbardziej podobny (względem miary *Similarity*, Def. 4.1.4) do opisu F. Jeśli podobieństwo jest zbyt małe (< 0) wtedy odbiorca stwierdza, że nie potrafi zinterpretować wypowiedzi W_0. W takim wypadku *nadawca* wskazuje mu temat. Pseudokod funkcji interpretującej `InterpretujWypowiedź` przedstawiamy poniżej.

IterpretujWypowiedź(W_0,U_n)

F ← suma wszystkich znaczeń *znanych* słów w W_0
temat ← argmax: Similarity(F,a) for a ∈ A
if Similarity(F,temat) > 0 then
    return temat
else
    FAIL

4.2.5. Formowanie słowników

Po wstępnym zbudowaniu wypowiedzi W_0, *nadawca* przed wysłaniem jej do odbiorcy sprawdza, czy sam potrafi ją odczytać. *Nadawca* interpretuje W_0 (metoda `InterpetujWypowiedź`) i ewentualnie, gdy podobieństwo do tematu jest zbyt małe[2], może utworzyć nowe słowa i dodać je do wypowiedzi.

Aby zapobiec zbytniemu rozrastaniu się słowników, *nadawca* dodaje nowe wpisy z pewnym prawdopodobieństwem – tym większym im bardziej zinterpretowana wypowiedź różniła się od tematu. Wpisy dodawane są z małymi wagami, by ponownie nie dopuścić do zbytniego rozrostu słowników. Przedstawiony scenariusz realizuje metoda `NoweSłowo`.

NoweSłowo(W_0,x_0,U_n)

temat ← IterpretujWypowiedź(W_0,U_n)
if temat $\neq x_0$ then
    F ← suma wszystkich znaczeń *znanych* słów w W_0

[2]mniejsze od ustalonego progu akceptacji μ – parametr symulacji

```
    p ← liczba losowa z przedziału (0,1)
    if Similarity(F,temat) - Similarity(F,x_0) > p then
        noweZnaczenie ← Cechy(x_0) \ F
        NowyWpis(losowyNapis, noweZnaczenie, waga = mała_stała)
```

Jeśli *odbiorca* w otrzymanej wypowiedzi W_0 napotkał nieznane słowa, próbuje wydedukować ich znaczenie z kontekstu, tzn. tworzy nowe wpisy w słowniku (z małymi wagami) wg poniższego algorytmu:

AdoptujPowiązania(x_0,U_n,W_0, p = stała < 1)

```
W' ← słowa w wypowiedzi W_0, których nie ma w słowniku
for w in W' do
    for f in Cechy(x_0) do
        waga_f ← 1/(liczba obiektów w U_n, które posiadają cechę f)
        NowyWpis(w, f, waga_f*p)
```

Po każdej iteracji symulacji obaj gracze wprowadzają drobne korekty do swoich słowników (metoda KorektaSłownika), aby podkreślić użyteczne powiązania i osłabić te przeszkadzające w komunikacji. Gdy wartość danego powiązania spadnie do zera, dany wpis jest usuwany ze słownika.
Ponadto w przypadku gry zakończonej porażką, *odbiorca* dodaje nowe powiązania do swojego słownika.

Funkcjonuje również stworzony przez nas mechanizm zapominania słów. Słowa, które długo nie były używane są powoli zapominane i usuwane ze słownika.
Opisane wyżej działania realizuje poniższa metoda.

KorektaSłownika(x_0,W_0,gracz)

```
p = stała < 1
wspólne ← Cechy(W_0) ∩ Cechy(x_0)

for w in W_0 do
    for f in Cechy(w) do
        if f ∈ wspólne then
            WzmocnijPowiązanie(w,f)
        else
            ZmniejszPowiązanie(w,f)
            if Powiązanie(w,f) < 0 then
                UsuńWpis(w, f)

if gracz = odbiorca and wynikGry = FAIL then
    rozłączne ← Cechy(W_0) \ Cechy(x_0)
    for w in W_0 do
        for f in rozłączne do
            NowyWpis(w,f,p)

for w in Slownik do
    if w in W_0 then
        Utrwal(w)
    else
        StopniowoWymazuj(w)
```

Rozdział 5

Eksperymenty

5.1. Narzędzia

Narzędzie do symulacji ewolucji języka dołączone do niniejszej pracy to zestaw udokumentowanych modułów w języku Python. Do tego dołączone są skrypty GNU/Octave, które zostały użyte do wygenerowania wykresów prezentowanych na stronach niniejszej pracy.
W skład pakietu wchodzą:

- `lgame.py` – główny moduł definiujący klasę gry (`Game`) oraz klasę agenta (`Agent`),
- `play.py` – przykładowy program przeprowadzający symulacje opisane w tym rozdziale,
- `lfuzzy.py` – moduł zawierający definicje pojęć bazujących na zbiorach rozmytych używanych przez agenta. Definiuje klasy:
  - `lfuzzy.Concept` – klasa bazowa dla pojęcia rozmytego,
  - `lfuzzy.WordConcept` – klasa pochodna od `Concept` zawiera definicję słowa, które jest zapisywane w słowniku,
  - `lfuzzy.Dictionary` – klasa definiująca interfejs słownika używanego przez agenta,
  - `lfuzzy.SteelsConcept` – klasa definiująca pojęcie jak w artykule [1] (inaczej niż w pracy),
  - `lfuzzy.SteelsWordConcept` – definiuje słowo (jak `WorcConcept`, ale na bazie `SteelsConcept`),
  - `lfuzzy.NonLinearConcept` – definiuje pojęcie z nieliniową inkrementacją[1] powiązań między słowami a cechami,
- `simplereducts.py` – definiuje klasy znajdujące redukty oraz `NullReductFinder` zwracającą wszystkie cechy.

By uruchomić przykładowy scenariusz, wybieramy katalog `tests` i uruchamiamy plik `play.py`, podając w jedynym argumencie liczbę iteracji:

```
> python play.py 1000
```

Na ekran będzie wypisywany postęp w procentach. Po zakończeniu w katalogu `run` zostanie utworzony katalog z wynikami. Poprzednie wyniki nie zostaną nadpisane, zostanie utworzony katalog z numerem o jeden wyższym. W katalogu z wynikami znaleźć można następujące pliki:

[1] por. funkcja KorektaSłownika w rozdz. 4.2.5.

- `game.log` – przebieg całej symulacji: dokładny zapis każdej gry oraz parametrów symulacji,
- `game.result` – i-ty wiersz pliku przedstawia rezultat i-tej iteracji gry,
- `game.succ` – i-ty wiersz pliku to 1 w przypadku gry zakończonej sukcesem, 0 – porażką,
- `dict.size-k` – i-ty wiersz pliku przedstawia rozmiar słownika (liczba wpisów) k-tego gracza po i-tej grze,
- `dict.full` – plik przedstawiający słowniki wszystkich graczy po ostatniej iteracji gry.

Parametry symulacji możemy ustawić zmieniając wartości pól klasy `Agent` w pliku ze scenariuszem symulacji `play.py`, są to:

- `Agent.CONCEPT` – fabryka definiująca pojęcia. Wartość tego pola można ustawić na `lfuzzy.Concept`, `lfuzzy.SteelsConcept` lub `lfuzzy.NonLinearConcept`. Są to trzy zdefiniowane przez nas klasy,
- `Agent.WORD_CONCEPT` – ustawienie fabryki definiującej wpisy w słowniku. Zdefiniowaliśmy: `lfuzzy.WordConcept`, `lfuzzy.SteelsWordConcept`, `lfuzzy.NonLinearWordConcept`,
- `DICTIONARY` – ustawienie używanego modelu słownika: `lfuzzy.Dictionary`,
- `Agent.DEFAULT_ADOPTION_MEMBERSHIP` – liczba rzeczywista z przedziału $(0, 1)$: współczynnik mówiący o sile powiązań w słownikach, które są adoptowane od innych graczy w przypadku gry zakończonej porażką. Im większa wartość tym adoptowane powiązania są silniejsze,
- `Agent.DEFAULT_DISTINCT_GUESSING_FACTOR` – liczba rzeczywista > 0: współczynnik mówiący o tym jak, bardzo zróżnicowane będą tworzone powiązania podczas próby odgadywania znaczenia słowa z kontekstu, gdy po porażce *nadawca* wskaże temat gry. Im większa wartość tym algorytm zgadujący będzie mógł bardziej różnicować siłę powiązań odgadywanych cech z nieznanym słowem. Efekt działania zależy od implementacji metody agenta `Agent.adopt_concepts`,
- `Agent.DEFAULT_NEW_CONCEPT_MEMBERSHIP` – liczba rzeczywista z przedziału $(0, 1)$: początkowa waga powiązania nowo utworzonego słowa z jego znaczeniem,
- `Agent.DEFAULT_NEW_CONCEPT_EPS` – liczba rzeczywista z przedziału $(0, 1)$: im większa wartość tym większa szansa, że agent będzie potrafił wymyśleć nowe słowo, gdy będzie mu go brakowało do opisania tematu. Prawdopodobieństwo, zależy też od tego jak dobrze aktualne zdanie opisuje temat. Wpływ na działanie symulacji zależy od implementacji metody agenta `Agent.try_invent_new_word`. Domyślnie sprawdzany jest warunek
  `if similarity(B*, topic) + NEW_CONCEPT_EPS < RANDOM(0,1)`

  B^* – por. rozdz. 4.2.3,
- `Agent.ASOCIATION_STEP` – liczba rzeczywista z przedziału $(0, 1)$: współczynnik o który zwiększane bądź zmniejszane są siły powiązań w słowniku w zależności od wyniku gry. Wpływ na działanie symulacji zależy od implementacji słownika (pole `DICTIONARY` powyżej),

- `Agent.DEFAULT_SCORE_INC` – liczba rzeczywista z przedziału $(0, 1)$: współczynnik, który mówi o szybkości utrwalania użytecznych słów w słowniku. Im większa wartość tym słowa są szybciej utrwalane w pamięci agentów,
- `Agent.DEFAULT_SCORE_DEC` – liczba rzeczywista z przedziału $(0, 1)$: współczynnik, który mówi o szybkości zapominania rzadko używanych słów w słowniku. Im większa wartość tym słowa są szybciej zapominane,
- `Agent.REDUCT_FINDER` – ustawienie fabryki służącej do wyszukiwania reduktów. Zdefiniowaliśmy:
  - `simplereducts.NullFinder` -zwraca wszystkie atrybuty tematu,
  - `simplereducts.BruteForceFinder` -wyszukuje wszystkie redukty, od najkrótszych do najdłuższych.

W pliku `play.py` zdefiniowane są ponadto:

- mapa `OBJECTS` – obiekty obserwowane przez agenty,
- mapa `AGENTS` – zbiór wszystkich agentów.

Do narzędzi dołączony jest też skrypt GNU/Octave generujący wykresy z danych wynikowych. Aby wygenerować wykresy sukcesów gier oraz zmiany wielkości słowników należy uruchomić plik `graph.octave` z następującymi parametrami: *numer_symulacji, lista_numerów_agentów*. Na przykład,
`> octave graph.octave 667 1 2 3 4 5`
wygeneruje wykresy wyników symulacji z katalogu `run/667` przedstawiające średnią kroczącą liczby gier zakończonych sukcesem (średnia ze 100 kolejnych gier) oraz zmiany rozmiarów słowników pierwszych pięciu agentów (lista: `1 2 3 4 5`).

5.2. Sukces komunikacyjny. Rozmiar słownika

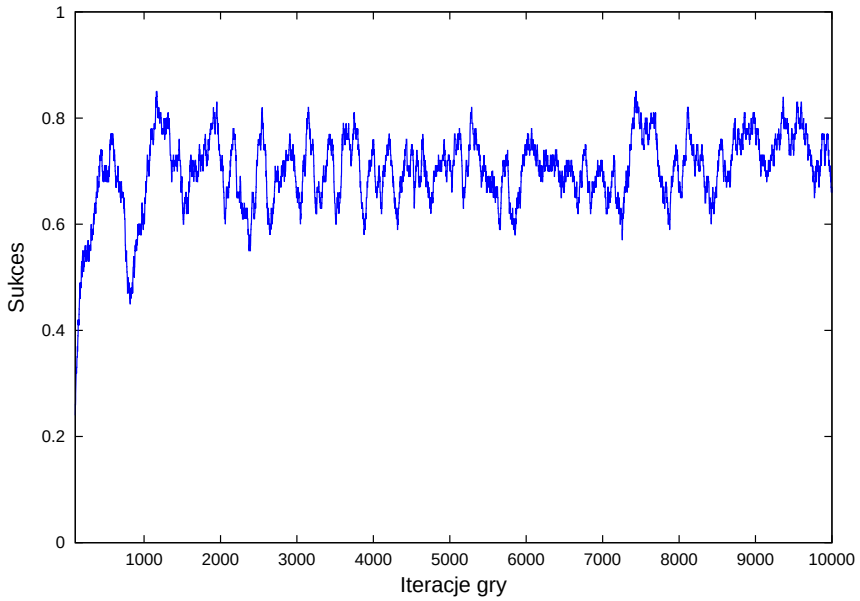
Wszystkie eksperymenty zakładają brak ogólnego nadzoru. Agenty są autonomiczne i mają wgląd tylko i wyłącznie w swoje własne słowniki. Uruchamiając symulację możemy zaobserwować kilka ciekawych zjawisk.
Pierwsze wnioski można wyciągnąć analizując sukces komunikacyjny oraz wielkości słowników. Wykresy opisujące sukces komunikacyjny przedstawiają średnią kroczącą liczby sukcesów ze 100 kolejnych gier. Zapisy z przebiegu symulacji scenariuszy opisanych w tym podrozdziale znajdują się w katalogach `lgame/tests/run/`[*litera_scenariusza*].

5.2.1. Scenariusz A. Zapominanie

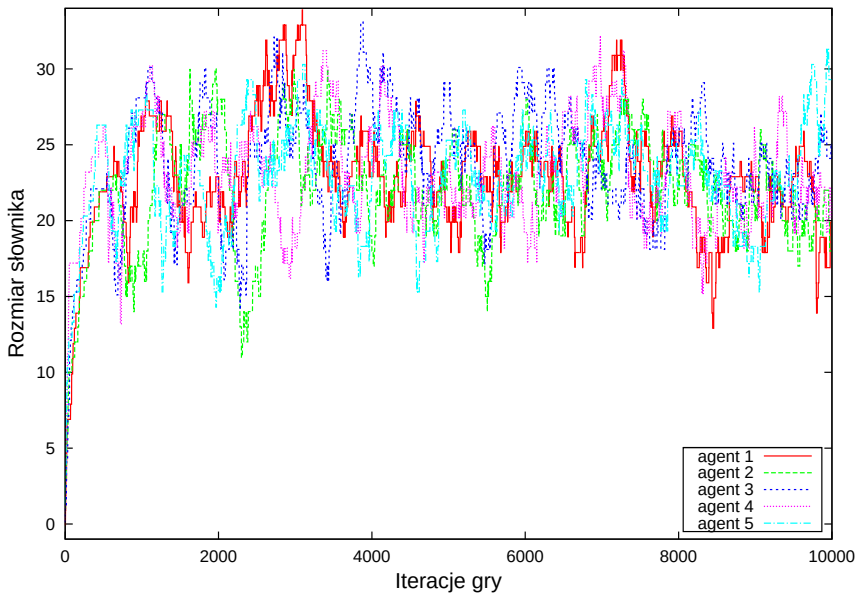
Mechanizm zapominania służy eliminacji nieużywanych słów z języka (por. parametry `Agent.DEFAULT_SCORE_DEC`, `Agent.DEFAULT_SCORE_INC`). Początkowo wszystkie agenty mają puste słowniki i by opisać nowo poznane obiekty wymyślają nowe słowa. Bez mechanizmu zapominania słowniki bardzo się rozrastają. Z drugiej strony zbyt krótka pamięć uniemożliwia stabilizację języka.

Rysunki 5.1, 5.2 przedstawiają symulację, w której każde użycie słowa utrwala je w pamięci, natomiast brak użycia danego słowa powoduje stopniowe jego zapominanie.

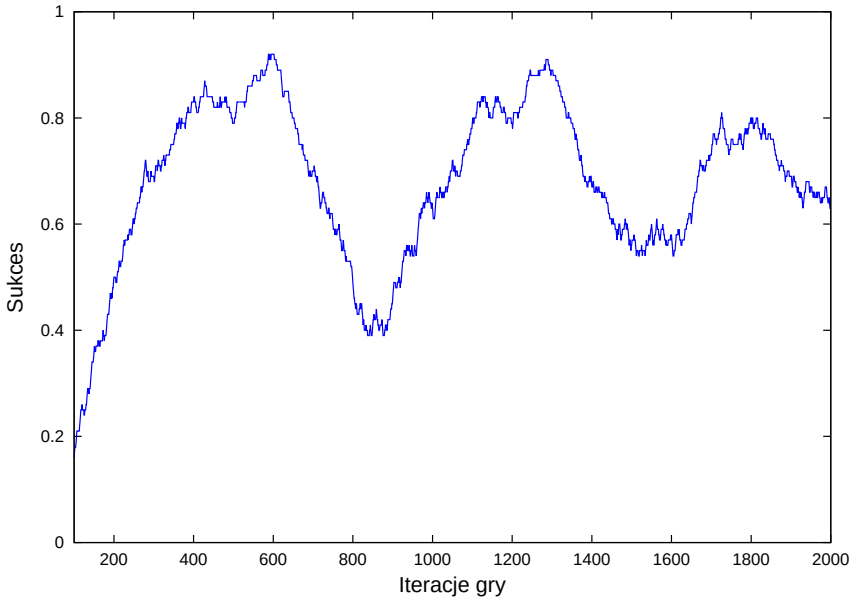
Sukces komunikacyjny z początku szybko rośnie i w trakcie kolejnych gier oscyluje w granicach 60-80%. Wielkości słowników oscylują w przedziale 15-30 słów.



Rysunek 5.1: Scenariusz A. Sukces komunikacyjny gry językowej agentów z dużym współczynnikiem zapominania (parametr `DEFAULT_SCORE_DEC`).



Rysunek 5.2: Scenariusz A. Zmiany rozmiarów słowników pierwszych 5-ciu agentów z dużym współczynnikiem zapominania (parametr `DEFAULT_SCORE_DEC`).



Rysunek 5.3: Scenariusz A. Sukces komunikacyjny gry językowej agentów z dużym współczynnikiem zapominania (parametr `DEFAULT_SCORE_DEC`) – w mniejszej skali: 2000 iteracji

Przyglądając się rysunkom 5.3 i 5.4, które są wykresami symulacji o identycznych warunkach tylko w mniejszej liczbie iteracji, widać korelację pomiędzy sukcesem komunikacji, a wielkością słowników. Agent operujący na pewnej przestrzeni obiektów, zapomina słowo, które opisuje rzadziej obserwowany fragment tej przestrzeni (rzadziej występujące pewne cechy obiektów). Wtedy albo wymyśla nowe słowa, albo „przypomina" już istniejące (od agentów, które je jeszcze pamiętają). Stąd wahania wielkości słowników, a przy zbyt małych słownikach spadek sukcesów w komunikacji.

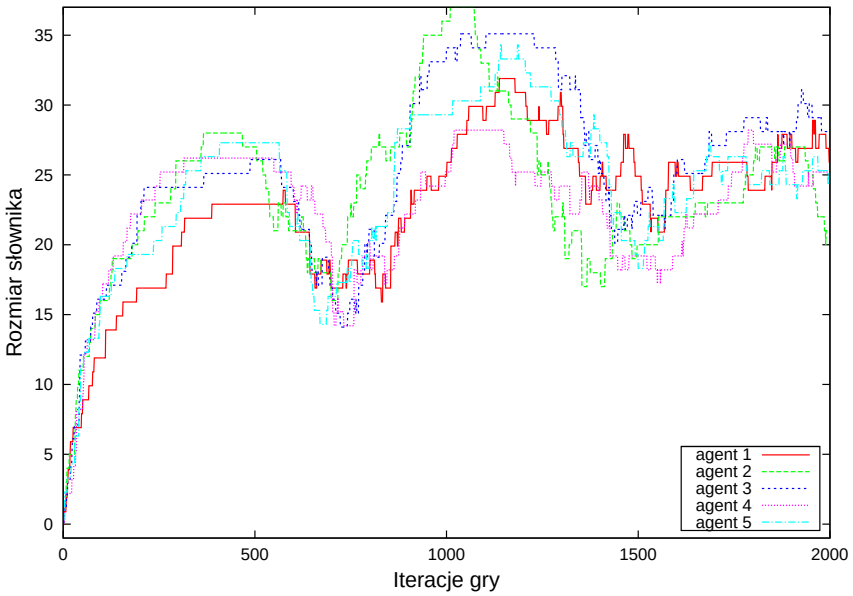
5.2.2. Scenariusz B. Dobieranie słów

Rysunki 5.5 oraz 5.6 pokazują, jak bardzo ważna jest metoda dobierania słów. Pozwolono tu graczom na dobieranie słów, które nie pogarszają, ale nie koniecznie zwiększają *ocenę* wypowiedzi (por. metoda `BudujWypowiedź`, podrozdział 4.2.3). W wyniku tego agenty zaczęły używać zbyt wielu słów.

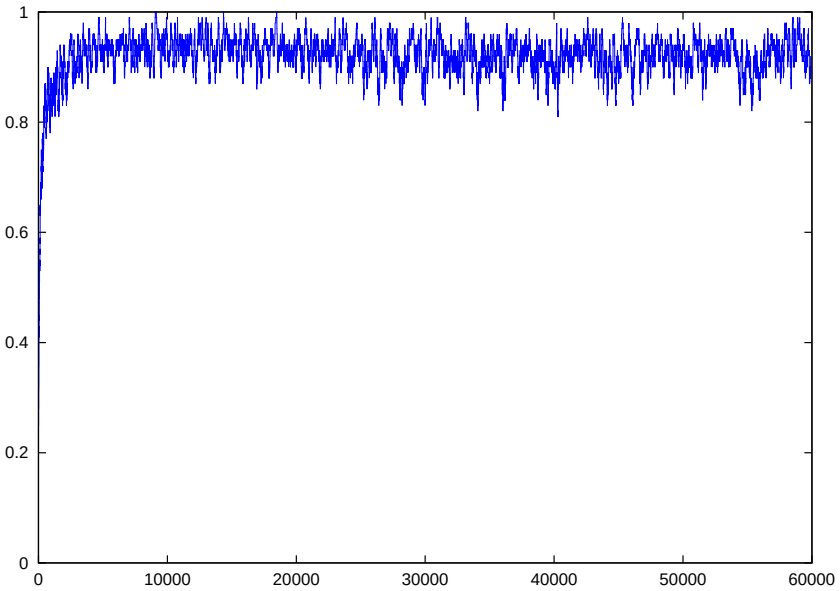
5.2.3. Scenariusz C.1. Ewolucja języka

Wreszcie wykresy na Rysunkach 5.7, 5.8 przedstawiają ewolucję słowników graczy z dobrze dobranymi parametrami odpowiadającym za pamięć (por. `DEFAULT_SCORE_DEC`, `DEFAULT_SCORE_INC`, podrozdział 5.1) oraz oszczędnym algorytmem dobierania słów (do wypowiedzi dodawane są tylko te słowa, które powiększają ocenę wypowiedzi, por. metoda `BudujWypowiedź`, podrozdział 4.2.3). Na Rysunku 5.8 widać początkowy szybki wzrost wielkości słowników, do momentu, gdy sukces komunikacyjny (Rys. 5.7) nie zatrzyma się na wysokim poziomie (około 1500 iteracji).

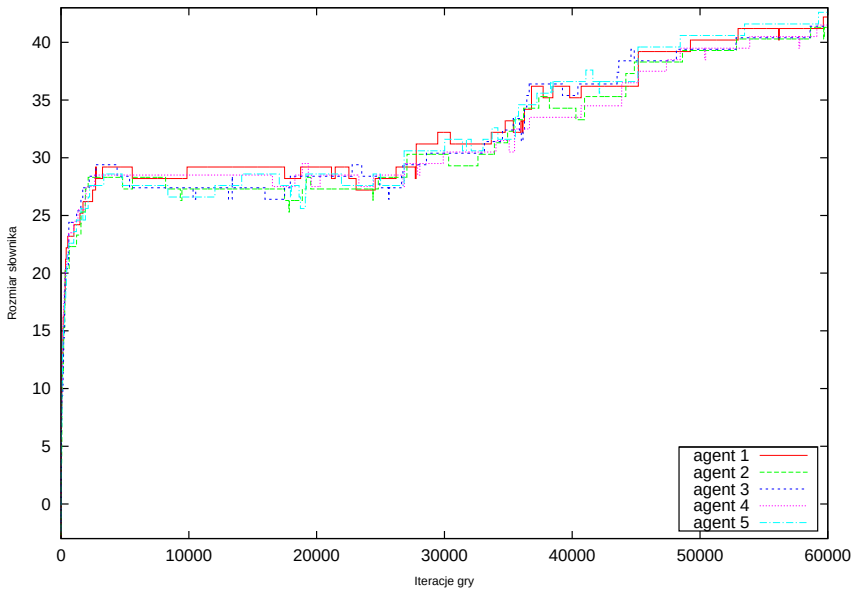
Później, dzięki mechanizmowi zapominania, następuje wygaszanie słów, które są rzadko używane (na Rys. 5.8 widać wyraźny spadek wielkości słowników w okolicach 2000 iteracji, przy zachowaniu wysokiego sukcesu komunikacyjnego – Rys. 5.7). Od 3000 iteracji zmiany w słownikach przestają być gwałtowne – od 3000 do 16000 (jest to bardzo długi okres) iteracji



Rysunek 5.4: Scenariusz A. Zmiany rozmiarów słowników pierwszych 5-ciu agentów z dużym współczynnikiem zapominania (parametr `DEFAULT_SCORE_DEC`) – w mniejszej skali: 2000 iteracji.



Rysunek 5.5: Scenariusz B. Sukces komunikacyjny gry językowej ze zmodyfikowanym algorytmem dobierania słów (metoda `BudujWypowiedź`).



Rysunek 5.6: Scenariusz B. Ciągły wzrost liczby słów w słowniku agentów ze zmodyfikowanym algorytmem dobierania słów (metoda `BudujWypowiedź`).

na wykresie 5.8 da się zaobserwować fazę stabilizacji. Po długim czasie ok. 16000 iteracji słowniki uzyskują stabilny poziom i od tego momentu praktycznie nie ewoluują. Wtedy liczba słów w słownikach odpowiada liczbie różnych obiektów.

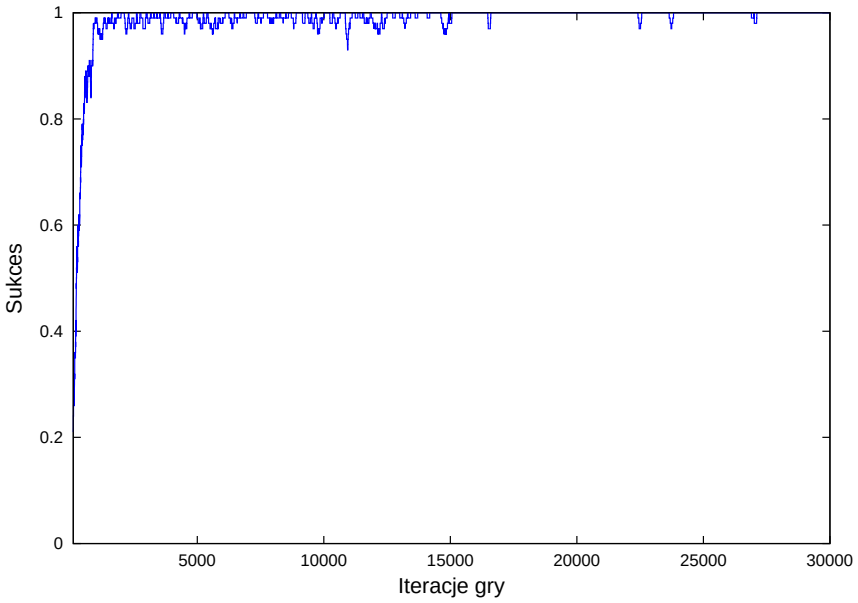
Podczas długiej obserwacji pewnej liczby obiektów (nie przekraczającej zdolności zapamiętywania agentów[2]), agenty ustalają bardzo szczególne słowa dokładnie opisujące każdy obiekt – powstają nazwy własne. .

5.2.4. Scenariusz C.2. Zmienne środowisko

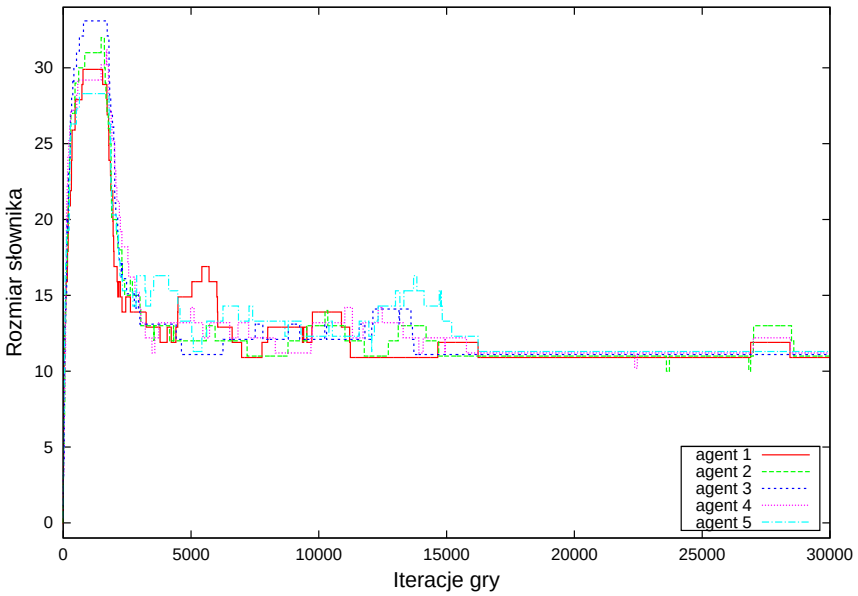
Powtarzając symulacje opisane w poprzednim paragrafie (5.2.3), w zmiennym środowisku (tzn. zmieniając przestrzeń obserwowanych obiektów), możemy zauważyć następujące zjawisko. Liczba słów w słowniku początkowo szybko rośnie. Później wygasają słowa nadmiarowe i ewolucja dalej biegnie, ale już nie tak gwałtownie.

Porównując słowniki graczy, z tymi z poprzedniego Scenariusza C.1. możemy zauważyć, że przetrwały słowa bardziej ogólne – nie będące dokładnymi opisami danych obiektów. Słowniki nie mogły zbiec do listy słów odpowiadających konkretnym obiektom, ponieważ nie było stałego zbioru obiektów.

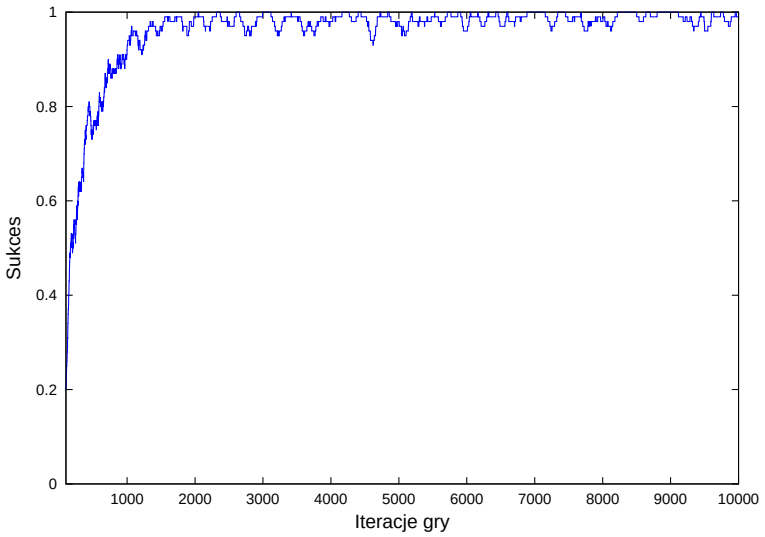
[2]Zdolność zapamiętywania zależy od implementacji funkcji `KorektaSłownika`, w szczególności od metod `Utrwal`, `StopniowoWymazuj` (por. Rozdz. 4.2.5)



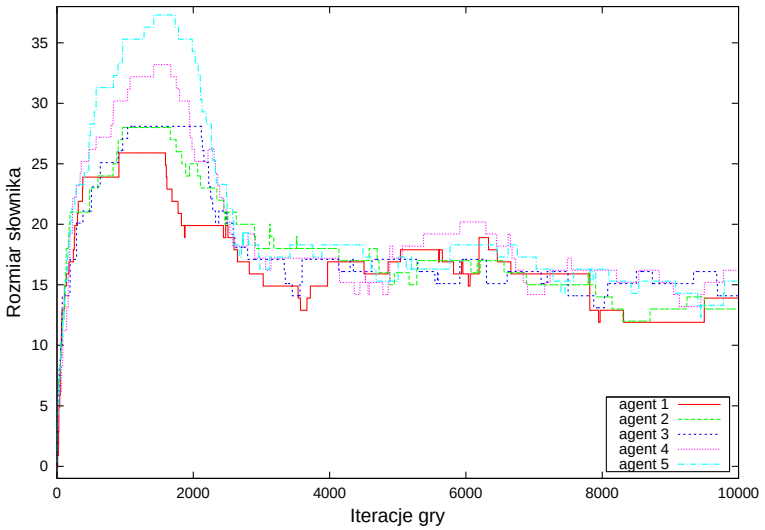
Rysunek 5.7: Scenariusz C.1. Sukces komunikacyjny gry językowej.



Rysunek 5.8: Scenariusz C.1. Widoczne trzy fazy ewolucji języka.



Rysunek 5.9: Scenariusz C.2. Sukces komunikacyjny gry językowej w zmiennym środowisku.



Rysunek 5.10: Scenariusz C.2. Szybka ewolucja słowników na początku.

Rozdział 6

Podsumowanie

W Rozdziale 1 omówiliśmy dokładnie model symulacji językowej Steelsa [2] oraz przedstawiliśmy problemy związane z jego implementacją. Aby poradzić sobie z problemami (por. podrozdział 1.5):

- obliczania reduktów i zbiorów rozróżniających,
- kodowania i dekodowania wypowiedzi,
- reprezentacją słownika,

w Rozdziale 4 zaprezentowaliśmy inny model symulacji językowej, zbudowany na bazie modelu Steelsa oraz Wellensa i innych [1].
Następnie przeprowadziliśmy serię eksperymentów, prezentującą możliwości nowego modelu. Symulacje pozwoliły zaobserwować zjawiska zachodzące podczas ewolucji języka. Zauważyliśmy, że:

- W niezmieniającym się środowisku (tzn, jeśli agenty długo obserwują próbki z pewnej niezmieniającej się przestrzeni obiektów) pojęcia w słownikach agentów zbiegają do nazw własnych, tzn. pojęć dokładnie odwzorowujących cechy obiektów. Zachowanie to przedstawia wykres na Rysunku 5.8 – po 15000 grach słowniki już nie ewoluują. Poszczególne wpisy w słownikach niemal dokładnie odzwierciedlają odpowiednie obiekty z tabeli, której wycinki były pokazywane graczom.
- Dzięki wprowadzonemu przeze mnie mechanizmowi zapominania, agenty usunęły ze swoich słowników nadmierne mapowania. Jest to widoczne, na Rysunkach 5.10 oraz 5.8, gdzie po 2000 iteracji obserwujemy wyraźny spadek liczby słów w słownikach przy jednoczesnym zachowaniu wysokiej skuteczności w komunikacji (por. Rys. 5.9, 5.7).
- W zmieniającym się środowisku (tablica obiektów, której wycinki są obserwowane przez agenty, nie jest stała) obserwujemy ciągłą ewolucję języka, przy jednoczesnym zachowaniu wysokiego poziomu skuteczności komunikacji. Przedstawiają to wykresy na Rysunkach 5.10, 5.8.

Ponadto, dzięki temu, że podczas gry językowej, nadawca wysyłając komunikat nie wskazuje od razu tematu, w symulacjach uwzględniliśmy niepewność referencyjną (por. Wstęp oraz podrozdział 1.5) – problem występujący podczas nauki języka przez ludzi.

Opisany nowy model został zaimplementowany w języku programowania Python, w sposób umożliwiający jego łatwe rozszerzanie (por. Rozdział 5), tzn. z sposób zmodularyzowany. Oprócz definiowania parametrów początkowych symulacji, możliwe jest dodanie lub modyfikacja dowolnego modułu lub podmodułu (np. implementacji konceptu, implementacji słownika, implementacji metod agenta) oraz zmiana scenariusza gry. Można też przeprowadzać symulacje, w których agenty różnią się od siebie (tzn. mają zaimplementowane inne metody, podpięte różne realizacje słowników, ustawione inne parametry, itd.)

Podsumowując, przedstawiony model, dokładniej niż model Steelsa [2], symuluje ewolucję słowników. Ponadto wymaga znacznie mniejszej mocy obliczeniowej, dzięki zaimplementowaniu słowników jako zbiory rozmyte oraz zastosowaniu algorytmu aproksymacyjnego wyznaczania reduktu.

Natomiast jedną z możliwości dalszych badań ewolucji języka w zespole agentów (co stanowi część badań nad *ogólną sztuczną inteligencją*[1]) jest próba poznania, jak języki konkurują ze sobą, np. zderzając ze sobą dwie lub więcej populacji agentów z wyewoluowanymi wcześniej językami i obserwując kształtowanie się słowników podczas kolejnych gier.

[1] Więcej o ogólnej sztucznej inteligencji można przeczytać we Wstępie.

Bibliografia

[1] P. Wellens, M. Loetzsch and L. Steels, *Flexible Word Meaning in Embodied Agents*, Connection Science Vol. 20(2-3): 1-18, 2008

[2] L. Steels, *The Spontaneous Self-organization of an Adaptive Language*, Machine intelligence 15, Oxford University Press, Oxford 1996

[3] H. S. Nguyen, *Approximate Boolean reasoning: Foundations and Applications in data mining*, Transactions on Rough Sets V: Journal Subline, LNCS Vol. 4100: 380-389, 473-474, Springer, Heidelberg 2007

[4] T. Oates, *Grounding Word Meanings in Sensor Data: Dealing with Referential Uncertainty*, HLT-NAACL-LWM '04 Proceedings of the HLT-NAACL 2003 workshop on Learning word meaning from non-linguistic data Vol. 6: 62-69, Association for Computational Linguistics Stroudsburg, PA, USA 2003

[5] Zdzisław Pawlak, *Rough sets*, International Journal of Parallel Programming, 11(5): 341-356, 1982

[6] Jan Bazan, Hung Son Nguyen, Sinh Hoa Nguyen, Piotr Synak, and Jakub Wróblewski. *Rough set algorithms in classification problems.* In , Rough Set Methods and Applications: New Developments in Knowledge Diskody in Information Systems, Studies in Fuzziness and Soft Computing Vol. 56:49-88. Physica-Verlag, Heidelberg, Germany, 2000

[7] A.Skowron, C. Rauszer, *The Dircernibility Matrices and Functions In Information Systems.* In R. Słowiński, *Inteligent Deciosion Support – Handbook of Applications And Advences of the Rough Sets Theory*, Kluwer Academic Publishers, Dordrecht, Netherlands, 1992

[8] L. Steels, *Language as a Complex Adaptive System*, Perallel Problem Solving from Nature PPSV, LNCS Vol. 1917: 17-26, Springer, Heidelberg 2000

[9] L. Steels, *Evolving Grounded Communication for Robots*, Trends in Cognitive Sciences Vol. 7(7): 308-312, July 2003

[10] P. Bloom, *How Children Learn the Meanings of Words*, MIT Press, Cambridge, MA, 2000

[11] S. Carey, *The child as word learner*, Linguistic Theory and Psychological Reality: 264-293. MIT Press, Cambridge, Ma, 1978

[12] M. Bowerman and S. Choi, *Shaping meanings for language: Universal and language-specific in the acquisition of spatial semantic categories*, Language Acquisition and Conceptual Development: 132-158. Cambridge University Press, Cambridge, 2001

[13] L. A. Zadeh, *Fuzzy sets*, Information and Control Vol. 8(3): 338-353, Elsevier Inc. 1965

[14] Z. C. Johanyák, S. Kovács, *Distance based similarity measures of fuzzy sets* Department of Information Technology, Kecskemét College, Hungary

[15] M. Setnes, R. Babuska, U. Kaymak, and H. R. van Nauta Lemke, *Similarity Measures in Fuzzy, Rule Base Simplification*, IEEE Transactions on systems, man, and cybernetics – part B: Cybernetics, Vol. 28, No. 3, June 1998

[16] E. Szmidt, J. Kacprzyk, *A Measure of Similarity for Intuitionistic Fuzzy Sets*, Polish Academy of Sciences, Warsaw, Poland

[17] CJ. Johnson *Cognitive components of naming in children: effects of referential uncertainty and stimulus realism*, Journal of Experimental Child Psychology, Vol. 53(1): 24-44, University of Western Ontario, London, Canada, 1992

[18] J. Ferber *Multi-Agent System: An Introduction to Distributed Artificial Intelligence*, Harlow, Addison Wesley Longman, 1999