KOMPUTEROWE PRZETWARZANIE WIEDZY

Kolekcja prac 2016/2017
pod redakcją Tomasza Kubika

KOMPUTEROWE PRZETWARZANIE WIEDZY

Kolekcja prac 2016/2017
pod redakcją Tomasza Kubika

Skład komputerowy, projekt okładki
Tomasz Kubik

ISBN 978-83-930823-9-1

Wydawca
Tomasz Kubik

Druk i oprawa
I-BiS sc., ul. Sztabowa 32, 50-984 Wrocław

Skład niniejszej książki wykonano w systemie LaTeX

SPIS TREŚCI

SŁOWO WSTĘPNE

Już po raz ósmy mam przyjemność zarekomendować Czytelnikom tom prac z dziedziny komputerowego przetwarzania wiedzy zredagowany przez dra inż. Tomasza Kubika. Tradycyjnie, autorami prac zebranych w tomie są studenci Wydziału Elektroniki Politechniki Wrocławskiej, studiujący w semestrze letnim roku akademickiego 2016/2017 na studiach II stopnia, na specjalności Robotyka kierunku Automatyka i robotyka. Ósmy tom pod tytułem „Komputerowe przetwarzanie wiedzy" liczy 56 stron druku i składa się z pięciu rozdziałów. Podobnie jak dotąd, ich celem jest przedstawienie wybranych metod i narzędzi do przetwarzania wiedzy oraz ich zastosowania do rozwiązania konkretnych zadań praktycznych. Autorzy kolejnych rozdziałów i tytuły ich prac są następujące:

1. Michał Prędkiewicz: *Identyfikacja biometryczna osób*
2. Kamil Bogus: *Użycie metadanych w semantycznym Internecie*
3. Paweł Jachimowski: *Wykorzystanie dalmierzy laserowych do nawigacji kwadrokopterem w zamkniętych pomieszczeniach*
4. Krzysztof Kwieciński: *Selekcja zawodników*
5. Witold Lipieta: *Techniczne wsparcie osób niewidomych i niedowidzących w odczytywaniu wzorów matematycznych*

- *Identyfikacja biometryczna osób*: Dokonano przeglądu znanych metod rozpoznawania ludzi na podstawie danych biometrycznych. Położono nacisk na rozpoznawanie odcisków palców, identyfikację tęczówki oka, rozpoznawanie geometrii dłoni i rozpoznawaniu głosu człowieka. Ze względu na duże znaczenie rozpoznawania głosu człowieka dla robotyki społecznej, przedstawiono implementację systemu do tego rodzaju rozpoznawania, działającego w oparciu o sieci neuronowe.
- *Wykorzystanie dalmierzy laserowych do nawigacji kwadrokopterem w zamkniętych pomieszczeniach*: Kwadrokoptery zyskują w naszych czasach znaczenie jednego z pod-stawowych typów robotów inspekcyjnych. Nawigacja tego rodzaju robotów opiera się na wykorzystaniu danych sensorycznych na temat otoczenia, w tym często danych pochodzących z dalmierzy laserowych. W rozdziale przedstawiono przegląd zastosowań dalmierzy laserowych w zadaniach nawigacji kwadrokopterów, ze szczególnym uwzględnieniem takich zadań, jak unikanie kolizji, mapowanie otoczenia, autonomiczne lądowanie, wy-

szukiwanie drogi w labiryncie i kalibracja czujników pokładowych. Wykonano badania w zakresie kalibracji na testowej konstrukcji kwadrokoptera.
- *Selekcja zawodników*: Na przykładzie zespołowej gry komputerowej Dota 2, wykorzystując zapisane historie rozgrywek, zaproponowano metodę optymalnego wyłonienia składu drużyny mającej wystąpić w meczu. Szczególną uwagę poświęcono wyborowi zawodników w ostatniej kolejce, w sposób optymalny uzupełniających stworzony dotąd zespół.

Podobnie jak poprzednie siedem, także niniejszy zbiór prac jest adresowany do Czytelników zainteresowanych praktycznymi aspektami komputerowego przetwarzania wiedzy. Życzę im, żeby zarówno ten, jak i poprzednie tomy pomogły Państwu znaleźć odpowiedź na nurtujące Państwa pytania.

Prof. Krzysztof Tchoń,
opiekun specjalności Robotyka,
Wrocław, październik 2017

ROZDZIAŁ 1

IDENTYFIKACJA BIOMETRYCZNA

M. Prędkiewicz

W rozdziale dokonano przeglądu metod automatycznej identyfikacji osób w oparciu o pomiary biometryczne. Opisano też próbę stworzenia własnego systemu rozpoznawania tożsamości na podstawie analizy zarejestrowanego głosu.

1.1. Wstęp

Biometria zajmuje się analizą pomiarów własności anatomicznych lub zachowania osób w celu rozpoznania ich tożsamości. Metody opracowane w tej dziedzinie mają szerokie spektrum zastosowań [1]. Wykorzystuje się je np. w kryminalistyce, w systemach zabezpieczeń dostępu do pomieszczeń, do ochrony danych wirtualnych (np. poprzez instalowanie czytnika linii papilarnych w laptopach). Służą one do budowy autonomicznych systemów, działanie których może bazować w części na wiedzy i doświadczeniach ekspertów, dla których nie bez pozostaje wydajność i skuteczność zaimplementowanych w nich algorytmów [2].

Specyfika zastosowań biometrii nakłada spore wymagania na wiarygodność otrzymywanych wyników. Na przykład całkowita pewność co do efektów identyfikacji niezbędna jest podczas prowadzenie różnego typu dochodzeń, gdyż w innym przypadku nie można byłoby ich uznać za dowód w sądzie. Również w przypadku zabezpieczenia dostępu do pomieszczeń, sejfów czy też danych wirtualnych o dużej wartości ważne jest, aby użytkownik systemu miał pewność, że system nie pozwoli na dostęp do jego własności osobom postronnym.

W przypadku kryminalistyki często mówi się o tzw. ustalaniu tożsamości. Opiera się ono na porównaniu zebranej próbki z całą bazą zebranych wcześniej wzorców (porównanie 1:N). Najważniejszym wymaganiem jest, by w jego trakcie nie wskazać osoby, która nie popełniła wykroczenia lub przestępstwa w analizowanej sprawie. Błędna decyzja mogłaby bowiem zaburzyć postępowanie i w efekcie doprowadzić do ukarania lub skazania niewinnej osoby.

W przypadku zabezpieczeń czyjejś własności chodzi o potwierdzenie tożsamości. Procedura polega wówczas na porównaniu zebranej próbki biometrycznej z pojedynczym wzorcem osoby uprawnionej do dostępu (porównanie 1:1). W jej trakcie może dojść do sytuacji, w której nawet osoba uprawniona nie zosta-

nie rozpoznana za pierwszym razem. Takie przypadki mogą wynikać z reguły, że osoby nieuprawniona muszą być odrzucane zawsze.

Systemy identyfikacji biometrycznej można kategoryzować według różnych kryteriów. W podstawowym podziale wymienia się [3, 4, 5]: i) systemy opierające swoje działanie na własnościach anatomicznych ludzkiego ciała oraz ii) systemy, które identyfikują tożsamość na podstawie charakterystycznych zachowań. W systemach z pierwszej grupy dokonuje się statycznego pomiaru danej cechy anatomicznej. Do cech wykorzystywanych w identyfikacji należą:

- DNA,
- odciski palców,
- geometria dłoni lub twarzy,
- termika dłoni lub twarzy,
- tęczówka oka,
- układ żył w dłoni, w naczyniówce lub białkówce oka.

W systemach z drugiej grupy rejestruje się efekty działania czy też zachowania człowieka. Podstawą do identyfikacji są zaobserwowane zależności czasowe pomiędzy wystąpieniami charakterystycznych zdarzeń, jak i cechy wyekstrahowane w czasie. Badane są tu świadome zachowania, choć nie jest to wymogiem. Do najczęściej wykorzystywanych można zaliczyć:

- odręczny podpis,
- rozpoznanie głosu osoby mówiącej,
- rytm uderzania w klawisze.

Możliwe jest również wykorzystanie takich zachowań, jak:

- fale mózgowe,
- dynamika gałki ocznej i źrenicy,
- sposób chodzenia,
- ruch warg w trakcie mowy.

1.2. Przegląd technik identyfikacji biometrycznej

1.2.1. Rozpoznawanie odcisków palców

Odciskami palców nazywa się ślady pozostawiane na dotkniętych powierzchniach. Ślady te odzwierciedlają nierówności występujące na skórze opuszków palców dotykającego i, ze względu na swą unikalność, pozwalają na wiarygodną identyfikację tożsamości (zobacz rysunek 1.1).

Podstawowymi elementami odcisków są grzbiety i doliny, czyli podłużne „żłobki" odpowiadające nierównościom na powierzchni skóry. Wyróżnionymi elementami odcisków są tak zwane punkty osobliwe. Zalicza się do nich wiry, pętle oraz delty. Są to najbardziej charakterystyczne części odcisku. Wysunięty najbardziej na północ wir lub pętla nazywany jest rdzeniem i służy do wyrównywania obrazów przed identyfikacją.

Do grupy punktów charakterystycznych należą także minucje (detale Galtona): zakończenia linii, rozwidlenia linii, oczka, odcinki, skrzyżowania, styki boczne, haczyki, mostki, mostki bliźniacze, linie przechodzące. Do cech szczególnych odcisków zalicza się jeszcze: pory, kształty linii, linie zdegenerowane, zmarszczki, brodawki, blizny. Elementy z tej grupy pozwalają na identyfikację w przypadku posiadania tylko części odcisku, natomiast wymagają większej rozdzielczości (1000 dpi).

Rys. 1.1: Przykładowy odcisk palca wraz z wyszczególnionymi punktami: 1 – zakończenie, 2 – rozwidlenie, 3 – odcinek, 4 – oczko, 5 – dolina, 6 – grzbiet, 7 – linia przechodząca, 8 – styk boczny, 9 – haczyk, 10 – pętla [źródło: `https://commons.wikimedia.org/wiki/File%3AFingerprint_picture.svg`, by The Photographer (own work)]

Istnieją różne metody pozyskiwania cyfrowych obrazów odcisków palców:

1. metody off-line:
   - digitalizacja odcisków z kart daktyloskopijnych (ze starszych baz danych),
   - digitalizacja odcisków utajonych (np. w trakcie śledztw z wykorzystaniem daktyloskopijnego magnetycznego proszku fluorescencyjnego).
2. metody on-line:
   - pojemnościowe,
   - optyczne,
   - naciskowe,
   - termiczne,
   - ultradźwiękowe,
   - bezdotykowe.

Obrazy odcisków pozyskane z czujników poddaje się obróbce wstępnej. Obróbka ta polega na uwypukleniu grzbietów i dolin, wyrównaniu jasności oraz ostrości obrazu. Ważne jest również, by w trakcie tej obróbki odfiltrowane zostały zakłócenia pochodzących z różnych źródeł:

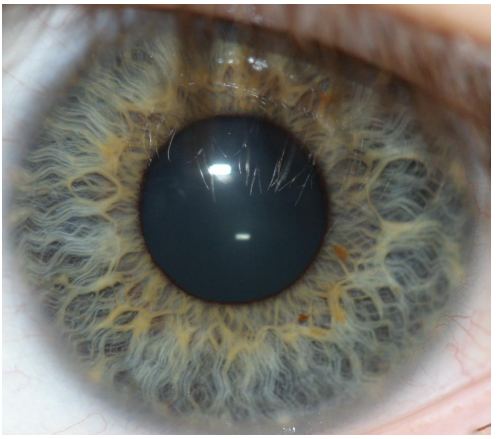
- deformacje zależne od siły nacisku na czujnik,
- mapowania odcisku 3D do 2D,
- wilgotności skóry,
- odcisków utajonych pozostawionych na urządzeniu w trakcie poprzednich pomiarów,
- niedokładności urządzenia.

Klasyfikacja właściwa może odbywać się na trzech poziomach:

1. globalnym (filtracja obrazów + kwantyzacja wyników filtracji),
2. lokalnym (porównywanie map położeń i orientacji minucji),
3. szczegółowym (porównywanie położenia, orientacji, ilości oraz kształtów szczegółów).

1.2.2. Identyfikacja tęczówki oka

Większość współcześnie stosowanych systemów identyfikacji i weryfikacji tożsamości na podstawie pomiaru cech oka użytkownika bazuje na wyglądzie tęczówki. Tęczówka (rys. 1.2) pełni w oku rolę przesłony, która w razie nadmiernego oświetlenia ogranicza ilość światła wpadającego przez źrenicę do wnętrza gałki ocznej. Jej wygląd oraz ruchy rzęskowe są indywidualne dla każdego człowieka.



Rys. 1.2: Przykładow tęczówki oka ludzkiego [źródło: `https://commons.wikimedia.org/wiki/File%3AColourIris.png` by Smhossei (own work)]

Identyfikacja na bazie wyglądu tęczówki polega na wykonaniu zdjęcia tęczówki osoby badanej i porównaniu jej ze wzorcem (weryfikacja) lub wzorami (identyfikacja) zawartymi w bazie systemu. Zdjęcia wykonuję się w spektrum światła widzialnego lub w bliskiej podczerwieni. Następnie algorytm wycina ze zdjęcia wszystkie nadmiarowe informacje: rzęsy, powieki, źrenicę czy białkówkę oka. Tak przygotowany obraz zostaje przekształcony na wzorzec, który porównuje się z bazą.

Zaletą omawianej metody, w porównaniu np. do metody identyfikacji na podstawie odcisków palców, jest dużo mniejsza szansa na wystąpienia zmian w badanym materiale. Brak tych zmian redukuje niebezpieczeństwo popełnienia pomyłki. Do wad tej metody zalicza się: cenę, trudność dopasowania skanera do wzrostu użytkowników oraz możliwość oszukania np. przez podłożenie dobrej jakości zdjęcia. Jednak największą wadą jest duża inwazyjność – pomiar jest nieprzyjemny, gdyż wymaga mocnego oświetlania szeroko otwartego oka.

1.2.3. Rozpoznawanie geometrii dłoni

Systemy wykorzystujące geometrię dłoni bazują głównie na unikalności jej kształtu. Do analizy porównawczej bierze się pod uwagę szerokość dłoni, wysokość dłoni, długości palców oraz szerokości palców. Pomija się natomiast inne cechy dłoni: odciski palców, znamiona czy kolor skóry.

Ten sposób identyfikacji jest bardzo wrażliwy na różnego typu zakłócenia. Przykładowo silniejszy docisk dłoni do czujnika może spowodować rozszerzenie opuszków i w konsekwencji błędne odczytanie szerokości dłoni i palców. Dodatkowo jeżeli działanie wykorzystywanego czytnika biometrycznego polega na wykonywaniu i analizie fotografii, duży wpływ na pomiar ma oświetlenie. Może ono spowodować, że cienie rzucane przez dłoń będą ją znacząco poszerzały lub całkowicie uniemożliwiają pomiar. Dlatego też podczas identyfikacji opartej na biometrykach omawiane systemy mogą służyć jedynie jako uzupełniające źródła informacji .

Zaletami identyfikacji geometrii dłoni jest to, że zarówno pomiar, przetwarzanie jak i porównanie z bazą wzorców są bardzo szybkie. Ponadto wykonywanie zdjęć podczas zbierania próbki jest bardzo proste i całkowicie nieinwazyjne dla użytkownika, a używane stanowiska pomiarowe nie wymagają dostosowywania do wzrostu użytkownika (jak w przypadku biometrii oka).

Rys. 1.3: Wygląd czytnika geometrii dłoni [źródło: `https://upload.wikimedia.org/wikipedia/commons/8/8b/Physical_security_access_control_with_a_fingerprint_scanner.jpg` by Lgate74 (own work)]

Podstawą czujnika jest pole, na którym użytkownik kładzie dłoń. Na tym polu rozmieszczone są ograniczniki wymuszające odpowiednie ułożenie dłoni, co ma zapewnić powtarzalność pomiarów oraz zebranie wszystkich cech potrzebnych do identyfikacji. Aby wykonywać jednoczesne zdjęcie dłoni od góry i od boku bez konieczności użycia dwóch aparatów fotograficznych często stosuje się lustro. Umieszczone obok dłoni kieruje obraz boku dłoni do obiektywu aparatu umieszczonego nad dłonią.

Przetwarzanie pobranego zdjęcia rozpoczyna się od binaryzacji, czyli przetworzenia obrazu do postaci, w której występują tylko dla kolory czarny i biały. Proces ten opiera się o proste i szybko wykonywalne progowanie piksel po pikselu. Poziom progowania musi być dobrany do natężenia świata oświetlającego dłoń w trakcie wykonywania zdjęcia. Następnym etapem jest ekstrakcja cech ze zdjęcia binarnego. Znajomość położenia ograniczników dłoni na polu pomiarowym pozwala w łatwy i szybki sposób zidentyfikować poszczególne palce oraz śródręcze (poprzez zawężenie obszaru poszukiwań). Wyznaczane są najczęściej szerokości wszystkich palców w dwóch lub trzech miejscach, długości wszystkich palców, szerokość śródręcza oraz wysokość dłoni w dwóch lub trzech miejscach.

Do porównywania uzyskanych cech z bazą danych wykorzystuje się najczęściej sztuczne sieci neuronowe lub komparatory progujące różnice pomiędzy wielkościami cech próbki i wzorców. Jako, że cech jest najczęściej poniżej dwudziestu, proces porównywania trwa bardzo krótko.

1.2.4. Rozpoznawanie głosu użytkownika

Systemy rozpoznające użytkowników po głosie są najczęściej stosowanymi systemami do identyfikacji biometrycznej według cech behawioralnych – czyli cech odzwierciedlających zachowanie człowieka w czasie, a nie cech odpowiadających jego aparycji czy budowie. Cechami behawioralnymi obsługiwanymi w biometrycznych zastosowaniach są: styl chodzenia, wystukiwanie rytmu, ruchy gałki ocznej itp.

Systemy badające wystukiwany rytmu (melodii) są bardzo zbliżone do systemów rozpoznawania głosu i w miarę proste do wdrożenia. Natomiast budowa rozwiązań umożliwiających identyfikację na postawie stylu poruszania się jest znacznie skomplikowana i droższa w implementacji – wymaga dobrze wyposażonego środowiska sprzętowego (z wieloma kamerami) oraz zaawansowanych algorytmów analizy i przetwarzania obrazów. Do zalet tego podejścia zalicza się prawie całkowita odporność na próby podrobienia wzorca („złamanie" systemu wymagałoby podmiany wzorca w bazie testowej). Biometria ruchów gałki ocznej zaś wymaga dosyć długiego i niewygodnego oświetlania oka w trakcie pomiaru.

Obecnie systemy identyfikacji głosu nie należą już do rzadkości. Można je spotkać w obszarze zabezpieczeń telefonów komórkowych czy laptopów. Nie wymagają złożonych obliczeń. Ich zaletą jest całkowicie intuicyjne i wygodne użytkowanie oraz małe zapotrzebowanie na dodatkowy sprzęt (mikrofon). Są one najtańszymi we wdrożeniu systemami identyfikacji biometrycznymi. Do ich wad można zaliczyć spadek skuteczności w hałaśliwym otoczeniu (np. na ulicy), wrażliwość na jakość mikrofonu oraz możliwość oszukania nagraniem wykonanym/zmontowanym wcześniej. Tego typu systemy można podzielić na takie, które: wymagają wypowiedzenia konkretnej frazy, wylosowują frazę do odczytania; działają niezależnie od tego, jakie słowa wypowie użytkownik. Te ostatnie są najtrudniejsze w implementacji oraz najbezpieczniejsze (najtrudniej oszukać je podstawionymi nagraniami).

Proces identyfikacji rozpoczyna się od nagrania próbki głosu użytkownika. Próbka jest najpierw poddawana filtracji w celu podwyższenia amplitudy wyż-

szych częstotliwości. Następnie nagranie dzielone jest na krótkie odcinki (o długości około 20-40 ms), dla których wykonywana jest szybka transformata Fouriera. Dla wszystkich odcinków wartości amplitud widma przeliczna są na skale melową. Jest to skala odpowiadająca reakcji ucha ludzkiego na dźwięk. Kolejne obliczenia polegają na przefiltrowaniu sygnału bankiem filtrów, których częstotliwości środkowe są dźwiękami słyszanymi jako dwa razy wyższy dźwięk od poprzedniego. Następnie wynik jest logarytmowany i obliczana jest energia poszczególnych pasm częstotliwości w analizowanej próbce głosu.

Uzyskiwanym wynikiem przetwarzania jest wektor współczynników cepstralnych (ang. *mel-frequency cepstral coefficients*, MFCCs) o długości równej liczbie filtrów w banku filtrów melowych, wyliczonych dla każdego odcinka nagrania. Wektor ten poddawany jest analizie porównawczej z innymi próbkami w bazie danych. Najczęściej wykorzystywane są do tego sztuczne sieci neuronowe. Możliwe jest również badanie głosu użytkownika za pomocy algorytmów predykcyjnych LPC (ang. *linear predictive coding*) i PLP (ang. *Perceptual Linear Prediction*) oraz po zastosowaniu transformacji falkowej – lecz metody te wymagają znacznie bardziej złożonych obliczeń. Transformata falkowa pozwala stworzyć klasyfikator, którego wyniki będą o kilka procent lepsze od metody melowych współczynników cepstralnych.

1.3. Implementacja systemu do identyfikacji osób na podstawie głosu

Implementację autorskiego system oparto na metodzie MFCC. Na początek nagrania normowane są w nim do jednakowej długości poprzez dodawanie na końcu nagrania ciszy (zerowego sygnału). Następnie dzielone są na 30 ms odcinki z 10 ms zakładkami. Dla każdego z odcinków obliczony zostaje wektor 20 współczynników cepstralnych.

Jako, że nagrania uczące i testowe mają około 3 s długości, więc po podzieleniu ich na 30 ms odcinki z zakładkami czasowymi otrzymano około 200 dwudziestoelementowych wektorów współczynników. Ostatecznie dawało to blisko 4000 wartości. Stworzenie sztucznej sieć neuronowej o takiej liczbie wejść jest nieracjonalne, bo ani nie zapewnia wydajności, ani nie daje dobrych efektów klasyfikacji (sieć jest zdecydowanie przeuczona wzorcami). Dlatego zdecydowano się na uśrednianie wszystkich wektorów do jednego, zawierającego 20 elementów. Przy takim zestawie cech perceptron bardzo szybko i skutecznie uczył się rozpoznawać poszczególne osoby.

Początkowo zastosowano sieć posiadającą 20 wejść, jedną warstwę ukrytą oraz wyjścia w liczbie równej liczności modelowanych mówców. Sieć ta była uczona różną liczbą nagrań mówców. Dane podawane na jej wyjścia (oczekiwane dane na wyjściu używane w procesie uczenia) generowano w następujący sposób: jeżeli na wejście podawane były współczynniki i-tego mówcy, wtedy na wyjście nr i podawano wartość 1, a na pozostałe wyjścia – wartość 0. W trakcie badań okazało się, że sieć o takiej konstrukcji, pomimo iż poprawnie rozpoznaje zamodelowanych mówców miedzy sobą (testowanie zbiorem testowym odręb-

nym od uczącego), zawsze wybiera jednego z nich – nawet jeśli wprowadzone zostanie nagranie osoby nie należącej do bazy. Sieć wybiera po prostu najbardziej prawdopodobnego mówcę. Ze względu na takie zachowanie sieci postanowiono zmienić architekturę rozwiązania.

W efekcie skonstruowano osobne sieci dla każdego z mówców. Każda z tych sieci posiada tylko jedno wyjście, które przyjmuje wartość 1 – jeżeli sieć uczona jest nagraniem swojego mówcy, i 0 – gdy uczona jest jakimkolwiek innym nagraniem. Z badań takiego systemu wynika, że sieci odrzucają nagrania niewłaściwych osób (innych niż te, dla których przeprowadzono uczenie). Wadą takiego systemu jest duża złożoność obliczeniowa – wymaga on stworzenia osobnego perceptronu dla każdego z użytkowników. Wynika z tego, że taka konfiguracja systemu sprawdzi się tylko dla małej liczby osób uprawnionych do dostępu oraz że nie nadaje się do identyfikacji, a jedynie do weryfikacji tożsamości.

Zbadana została również liczba nagrań mówców potrzebnych do nauczenia sieci. Wyniki testów sugerują, że już od trzech przebiegów uczących sieć jest w stanie rozpoznawać, czy ma do czynienia z właściwą osoba z pewności szacowaną na 0.9995. Gdy na sieć zostanie wprowadzona próbka danych wygenerowanych dla innej osoby, sieć zwróci wartość nie większą od 0.005. Gdy sieć uczona jest tylko dwoma lub jednym nagraniem pewność rozpoznawania znacznie spada. Jakość rozpoznawanie osób zadeklarowana w sieci plasuje się na poziomie od 0.9 od 0.95, co dalej jest dobrym wynikiem. Niemniej podczas testów z nagraniami innych osób pojawiały się wartości rzędu nawet 0.4 – co jest już znaczącym pogorszeniem. Ponadto sieć ma niekiedy problem z odrzuceniem osoby, której w ogóle „nie poznała".

W trakcie badań nad liczbą neuronów w warstwie ukrytej sprawdzono wartości od 1 do 200 i nie zauważono znaczących zmian w dokładności działania sieci. Główną różnicę stanowił czas nauki perceptronu, który przy sieciach o większej liczbie neuronów znacząco rósł. Największa liczba epok potrzebna do nauczenia sieci wystąpiła przy około 20 – 30 neuronach warstwy ukrytej.

Literatura

[1] A.K. Jain, A.A. Ross, K. Nandakumar. *Introduction to Biometrics*. Springer Publishing Company, Incorporated, 2011.

[2] J.L. Wayman, A.K. Jain, D. Maltoni, D. Maio. *Biometric Systems: Technology, Design and Performance Evaluation*. Springer Publishing Company, Incorporated, wydanie 1st, 2010.

[3] A.K. Jain, P. Flynn, A.A. Ross, redaktorzy. *Handbook of Biometrics*. Springer Publishing Company, Incorporated, 2007.

[4] L. Wang, X. Geng, redaktorzy. *Behavioral Biometrics For Human Identification: Intelligent Applications*. Medical Information Science Reference, Hershey, New York, 2009.

[5] S.Z. Li, A.K. Jain. *Encyclopedia of Biometrics*. Springer Publishing Company, Incorporated, wydanie 2nd, 2015.

ROZDZIAŁ 2

UŻYCIE METADANYCH W SEMANTYCZNYM INTERNECIE

K. Bogus

W rozdziale podjęto temat zastosowania metadanych do semantycznego opisywania zasobów publikowanych w Internecie. Dokonano w nim przeglądu standardów służących do budowy i publikowania modeli metadanych oraz wymieniono popularne słowniki używanych do tworzenia semantycznych opisów. Zaprezentowano też własne rozwiązanie, które umożliwia pozyskiwanie metadanych w postaci strukturalnej z opublikowanych dokumentów HTML.

2.1. Wstęp

Mimo upływu lat i rozwoju technologii informatycznych Internet pozostaje wciąż miejscem stosunkowo mało uporządkowanym. Czasem ciężko jest znaleźć w nim treści obiektywne i istotne, które mogą posłużyć rozwojowi ludzkości, zaś bezwartościowe artykuły pojawiają się niemal na każde zawołanie. W dużej mierze odpowiedzialność za ten brak porządku spoczywa na barkach dostarczycieli treści oraz wszelkiego rodzaju wyszukiwarek, które faworyzują niektóre zasoby zgodnie z zaimplementowanymi w nich algorytmami. Samo indeksowanie i rankingowanie opublikowanych zasobów jest mocno wrażliwe na wszelkiego rodzaju zabiegi związane z tzw. pozycjonowaniem. Zajmują się tym firmy dążące do wykreowania czyjegoś wizerunku – osoby bądź też przedsiębiorstwa, realizując przy tym różne cele. Automatyczna analiza przydatność opublikowanych zasobów wciąż bazuje na tzw. *backlinkach*. Bot, czy też programowy agent, szuka informacji o tym, jak wiele linków poleca daną treść oraz o tym, jak bardzo są one wiarygodne. Jeśli taki bot obdarzy „zaufaniem" jakieś źródło informacji, to wtedy łatwo daje się oszukać przez podwiązywanie bezwartościowych treści do treści uznanych wcześniej za istotne. Pewną receptą na tę sytuację mają być metadane oraz algorytmy semantycznego wnioskowania.

Poszukując wyjaśnienia pojęcia **metadanych** łatwo trafić na następującą definicję [1]: *metadane to ustrukturyzowane, czytelne maszynowo dane zawierające charakterystykę cyfrowych obiektów informacyjnych służącą ich efektywnemu*

i trafnemu wyszukiwaniu, zarządzaniu nimi i ich wartościowaniu. Metadane, a więc „dane o danych", mogą służyć do opisania znaczenia danych, ich jakości, powiązaniach z innymi zasobami, czy też zarządzania danymi. Sztandarowym przykładem systemów, w których wykorzystuje się metadane, są katalogi biblioteczne. Przechowywane w nich pozycje literaturowe są opisywane informacjami o autorze, tytule itd.

Chociaż metadane były wykorzystywane od samego zarania systemów informatycznych, zainteresowanie nimi w gronie użytkowników końcowych początkowo było mizerne. Pierwsze zmiany w tej dziedzinie zaobserwowano w latach 90. Szybki wzrost liczby publikowanych zasobów spowodował, iż podjęto kroki by oddzielić treści od tego, w jaki sposób są one prezentowane. Bardzo intensywnie zaczęto rozwijać nowe technologie wspierające wykorzystanie metadanych, takie jak np. XML, SOAP, RSS, UDDI. Projektowano je w celu ułatwienia dostępu, dystrybucji i manipulacji informacją niezależną od opisywanego zasobu.

Kolejnym znaczącym krokiem było opracowanie technologii, które pozwoliły na tworzenie i konsumowanie metadanych przez maszyny. Dostarczyły one semantycznej warstwy w wykorzystywanych modelach, tak, że opisywane treści nie tylko dało się automatycznie zidentyfikować, ale także właściwie zrozumieć i zinterpretować. Technologie te zaczęto nazywać technologiami sieci semantycznych web, zaś Internet, w którym są one wykorzystywane – semantycznym Internetem (ang. *Semantic Web*). Do głównych standardów sieci semantycznych web zalicza się m.in. RDF, RDFS czy OWL.

Przyjęto, że w trakcie budowy semantycznego Internetu w pierwszej kolejności powinna pojawiać się wiedza na temat danych, a dopiero później ich treść. Przy tej okazji opracowano zasady budowy semantycznych modeli informacyjnych, które nazwano ontologiami. Ontologie te stanowią formalną reprezentację wiedzy, na którą składa się zapis zbioru pojęć a także relacjami między nimi. Dzięki ontologiom można opisywać pewne zagadnienia, zasoby czy też dziedziny wiedzy w sposób zrozumiały przez komputery. Ontologie definiują więc to, jak opisywać daną dziedzinę i jej składowe oraz jak te opisy udostępniać. Dzięki ontologiom powstała szansa na przetwarzanie danych wcześniej nieprzetwarzalnych bądź trudnych do przetworzenia przez komputery, a co za tym idzie, na automatyczne zdobywanie wiedzy i rozwiązanie specyficznych problemów .

2.2. Modele metadanych

Proste systemy posługujące się semantycznym opisem zasobów można odnaleźć chociażby w bibliotekach cyfrowych. Owe zasoby, w postaci artykułów, raportów, książek itp. są opisywane za pomocą takich atrybutów, jak np. tytuł czy autor. Jak jednak przenieść taki opis do skali globalnego Internetu, zapewniając przy tym jednoznaczność i spójność z całym jego zakresie? Patrząc na ogrom publikowanych treści oraz różnorodność typów udostępnianych zasobów osiągnięcie tak postawionego celu zdaje się być niemożliwe. Na przestrzeni lat powstało jednak wiele rozwiązań, które uczyniły ten cel realistycznym. Część z nich omówiono poniżej.

2.2.1. Dublin Core

Powstały w roku 1995 standard Dublin Core jest jednym z najprostszych i najbardziej popularnych standardów metadanych [2]. W swojej podstawowej wersji definiuje 15 elementów używanych do opisu zasobów: tytuł, twórca, współtwórcy, temat, opis, wydawca, data, rodzaj, format, identyfikator, źródło, język, powiązanie, zasięg oraz prawa. Widać tutaj pewną analogię do systemów katalogowania pozycji bibliotecznych. Niestety, prostota użytego modelu przełożyła się również na obszar jego zastosowań. Standard Dublin Core w wersji podstawowej pozwala np. wyodrębnić wydawcę danego dzieła, jednak nie daje możliwości utworzenia jego szerszego opisu przez podanie adresu siedziby.

Powstanie standardów sieci semantycznych Web było jednym z impulsów do zweryfikowania zakresu modelu metadanych. Po kilkunastu latach liczba zadeklarowanych elementów zwiększyła się z 15 do 55. Aby odróżnić je od elementów oryginalnych zamiast nazwy elementy (ang. *elements*) przyjęto nazwę terminy (ang. *terms*). Terminy definiują zarówno nowe właściwości, jak i zawierają w sobie wszystkie stare elementy (niektóre z nich doczekały się uszczegółowienia). Oprócz 55 terminów standard ten definiuje również klasy i schematy zapisu treści. Poszczególne pola mogą być wypełnione wolnym tekstem, ich zawartość można ograniczyć do słownika „haseł wzorcowych" bądź opisać regułami kodowania (określającymi np. format wprowadzanych dat).

Rozwój standardu Dublin Core przełożył się na zwiększenie złożoności wykorzystywanego modelu. Aby go opanować oprócz znajomości poszczególnych elementów czy terminów należy również zapoznać się z wszystkimi klasami, schematami, a także szczegółami składni języka. Być może z tych powodów nowsza wersja standardu nie cieszy się tak dużą popularnością jak starsza wersja sprzed ponad 20 lat. Możliwe jednak, że to się zmieni. Warto podkreślić, że słownik Dublin Core publikowany jest w formacie RDF, a więc formacie pozwalającym na użycie pojęć z tego słownika w semantycznym Internecie.

2.2.2. RDF

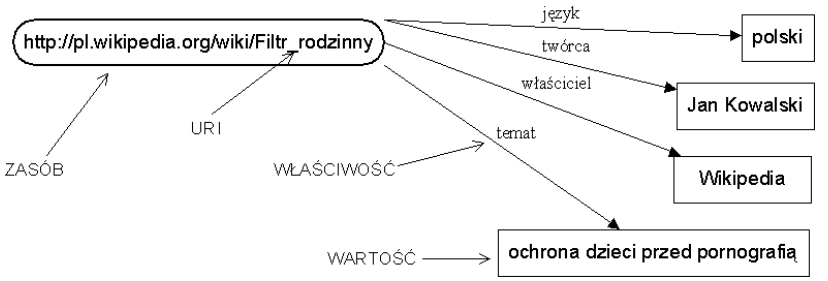
RDF (ang. *Resource Description Framework*) należy kojarzyć z zestawem standardów definiujących model opisu zasobów sieci Web, jak również języki zapisu tego modelu w postaci czytelnej dla człowieka i maszyn [3].

Generalnie modele tworzone w standardzie RDF mają postać skierowanych grafów. Węzły w tym grafach reprezentują zasoby (identyfikowane przez URI) bądź wartości (reprezentowane przez literały), zaś łuki reprezentują właściwości (predykaty). Wyjaśniając to dokładniej: grafy budowane są z tzw. trójek (ang. *triples*): podmiotu, predykatu (właściwości) oraz obiektu, który może przyjąć formę wskazywanego zasobu bądź literału.

W grafach RDF można gromadzić wiedzę z różnych dziedzin. Grafy te można wizualizować na interfejsach użytkownika (istnieją aplikacje, które pozwalają nawigować po grafach RDF), jednak najczęściej serializowane są one do plików przeznaczonych do maszynowego przetwarzania. Standardowo serializację modeli wykonuje się korzystając z języka RDF/XML. Jego składnia jest oparta na ję-

zyku XML. Istnieją również inne języki serializacji, jak np. Turtle czy JSON-LD (więcej na temat wzorców zapisu połączonych ze sobą danych, w tym także wspomnianego modelu grafowego, można znaleźć w [4]).

Na rysunku 2.1 przedstawiono przykład prostego modelu RDF. Uwidoczniono w nim wszystkie trzy opisane wcześniej elementy: podmiot (czyli zasób opisany w postaci linku), predykat (np. właściwość autor) oraz obiekt (np. wartość imię i nazwisko). Model ten można zapisać do pliku w formacie RDF/XML jak na listingu 2.1.



Rys. 2.1: Przykład opisu danych wykonanego zgodnie ze standardem RDF (źródło: [5])

Listing 2.1: Przykład serializacji danych w języku RDF/XML (źródło: [5])

```
<?xml version="1.0"?>
<RDF>
<Description about="http://pl.wikipedia.org/wiki/Filtr_rodzinny">
    <autor>Jan Kowalski</autor>
    <utworzono>1 stycznia 1970</utworzono>
    <zmodyfikowano>1 stycznia 2000</zmodyfikowano>
</Description>
</RDF>
```

Celem przyświecającym twórcom RDF było utworzenie ogólnoświatowego standardu do opisu zasobów publikowanych w Internecie. Standard ten pozwala nie tylko tworzyć metadane, ale również zapisywać rekordy samych danych. Rzem z RDFa oferuje więcej możliwości adnotowania treści w dokumentach HTML (tj. więcej możliwości zanurzania metadanych na stronach internetowych) niż sam znacznik `<meta>` zdefiniowany w języku HTML.

2.2.3. RDFa

Standard RDFa dostarcza metody osadzenia metadanych zgodnym z modelem RDF w plikach HTML. Dokładniej – definiuje on zestaw atrybutów z regułami ich użycia, dzięki którym treści zamieszczane w dokumentach HTML można automatycznie ekstrahować do trójek RDF. Już sama nazwa standardu – RDFa (ang. *RDF in (HTML) attributes*) odzwierciedla jego istotę. Opracowanie tego standardu miało pozwolić na wzbogacenie opisu treści zamieszczanych w dokumentach HTML, by stały się one zrozumiałe nie tylko dla ludzi, ale i dla maszyn.

Poniżej zamieszczono fragmenty kodu obrazujące sposób zagnieżdżania informacji za pomocą atrybutów HTML.

Fragment kodu na listingu 2.2) zawiera treści przeznaczone głównie dla ludzi. Wszystkie informacje zawarte w tym kodzie pojawią się na wynikowej stronie internetowej. Korzystając z RDFa w dokumencie HTML można zanurzyć metadane. Dodatkowe atrybuty, niewidoczne na renderowanej stronie, mogą posłużyć do automatycznej interpretacji wspomnianych treści przez maszyny. Na listingu 2.3 pokazano, jak tego dokonać. Po automatycznym przetworzeniu tak zmodyfikowanego kodu można pozyskać trójki jak na listingu 2.4 (przedstawione w notacji Turtle). Oczywiście przykład ten nie jest zbyt skomplikowany. Ilustruje jednak dobrze samą ideę dostarczania metadanych razem z opisywaną nimi treścią.

Listing 2.2: Przykład fragmentu kodu HTML bez adnotacji (źródło: [6])

```
...
  <h2>The Trouble with Bob</h2>
  <p>Date: 2011-09-10</p>
...
```

Listing 2.3: Przykład fragmentu kodu HTML z adnotacjami (źródło: [6])

```
...
  <h2 property="http://purl.org/dc/terms/title">
    The Trouble with Bob
  </h2>
  <p>
    Date:
    <span property="http://purl.org/dc/terms/created">
      2011-09-10
    </span>
  </p>
  ...
```

Listing 2.4: Trójki RDF pozyskane z fragmentu kodu HTML

```
@prefix dc: <http://purl.org/dc/terms/> .
[]  dc:title """The Trouble with Bob""";
    dc:created """2011-09-10""" .
```

2.2.4. OWL

OWL (ang. *Web Ontology Language*) jest językiem opartym na składni XML, umożliwiającym przetwarzanie informacji w sieci WWW. Choć bazuje na modelu RDF, daje większe możliwości interpretacyjne, bogatszy słownik i składnię niż sam RDF. Ontologie zredagowane w języku OWL mogą zawierać następujące elementy:

- relacje pomiędzy klasami,
- właściwości typu danych, stosowane do opisu atrybutów elementów klas (ich wartościami są literały),

- właściwości obiektów, stosowane do opisu związków pomiędzy elementami klas (ich wartościami są identyfikatory URI),
- instancje klas,
- instancje właściwości.

Standard języka OWL charakteryzuje się strukturą warstwową o rosnącej sile ekspresji. W zależności od potrzeb i przeznaczenia użytkownicy mogą wykorzystywać następujące dialekty tego języka:

- Lite – umożliwia jedynie tworzenie prostych relacji hierarchizujących pojęcia,
- DL – pozwala na tworzenie złożonych struktur pojęciowych poprzez nakładanie wielu rodzajów więzów na pojedynczą relację,
- Full – zawiera pełne słownictwo OWL, znacznie szerzej interpretowane niż w wersji DL, a także posiada pełne możliwości języka RDF.

Należy podkreślić fakt, iż wszystkie wymienione dialekty języka OWL powstały na bazie modelu RDF, tak więc generowane za ich pomoca opisy również przyjmują postać skierowanego grafu.

W 2012 roku opublikowano drugą, obecnie obowiązującą wersję standardu: OWL 2 Web Ontology Language, nieformalnie OWL 2. Wyróżnia się trzy jego dialekty (tzn. podzbiory języka), które charakteryzują się różną siłą wyrazu oraz różnym stopniem złożoności opisywanych nimi modeli. Są to:

- EL – umożliwia tworzenie ontologii z dużą liczbę klas i właściwości przy jednoczesnym zapewnieniu wielomianowego czasu przetwarzania problemów wnioskowania względem rozmiaru tych ontologii. Sprawdza się bardzo dobrze w przypadku dużych struktur danych czy licznego zbioru definiowanych klas, na których wykonywane są proste operacje.
- QL – służy do tworzenia modeli wiedzy z dużą liczbą instancji. Postawiono w nim nacisk na obsługę zapytań w najważniejszych procesach wnioskowania (sprawdzanie: spójności tych ontologii, pojęciowego zawierania się klas (ang. *class subsumption*), wystąpień instancji). Może być zaimplementowany za pomocą języka zapytań do relacyjnej bazy danych. Zawdzięcza też temu swoją nazwę: ang. *Query Language*).
- RL – jest ukierunkowany na budowanie rozwiązań, w których główny nacisk kładzie się na ekspresję przechowywanych danych, zaś na dalszy plan schodzi wydajność. Pozwala on na stosowanie standardowego opisu regułowego.

Różnice pomiędzy dialektami, z punktu widzenia korzystającego z nich użytkownika, polegają ograniczeniach w stosowaniu wybranych konstrukcji i terminów. Dobór właściwego dialektu uzależniony jest od potrzeb: ilości przetwarzanych zasobów, dopuszczalnego czasu przetwarzania, zakresu wymaganych funkcji itp.

Podobnie jak w przypadku RDF na bazie standardu OWL można tworzyć różne dziedzinowe ontologie – a więc zestawy pojęć (klas i właściwości), które mają zastosowanie w modelowaniu specyficznych fragmentów rzeczywistości.

2.2.5. FOAF

FOAF (ang. *Friend of a Friend*) jest to dziedzinowa ontologia zbudowana na modelu RDF (i OWL). Zebrano w niej zestaw pojęć (klas i właściwości) służących do opisu osób oraz relacji między nimi. Ontologia ta jest używana głównie do gromadzenia wiedzy dotyczącej sieci społecznościowych. Niekiedy ontologie dziedzinowe nazywane są też słownikami – a to z racji możliwości wykorzystania zdefiniowanych w nich pojęć w innych ontologiach.

Poniżej zamieszczono fragment kodu zawierający zestaw danych osobowych zredagowany z wykorzystaniem ontologii FOAF (elementy opatrzone przedrostkiem `foaf` to węzły reprezentujące pojęcia zdefiniowane w tej ontologii).

Listing 2.5: Przykład opisu, w którym wykorzystano ontologię FOAF (źródło: [7])

```
<?xml version="1.0">
<rdf:RDF
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
       xmlns:foaf="http://xmlns.com/foaf/0.1/">
<foaf:Person rdf:ID="me">
<foaf:name>Marek Zima</foaf:name>
<foaf:givenname>Marek</foaf:givenname>
<foaf:family_name>Zima</foaf:family_name>
<foaf:nick>zimek</foaf:nick>
<foaf:mbox_sha1sum>5d05cd897b9fb4b4c7734ed5cbe0cb986c5eb2fe
        </foaf:mbox_sha1sum>
<foaf:workplaceHomepage rdf:resource="http://adresnoexists.btw/"/>
<foaf:workInfoHomepage
        rdf:resource="adres przykładowy - niepoprawny"/>
<foaf:knows>
<foaf:Person>
<foaf:name>monika</foaf:name>
<foaf:mbox_sha1sum>e2685f60c81e66c00044e0225442d157c2b9ce94
        </foaf:mbox_sha1sum>
</foaf:Person>
</foaf:knows>
</foaf:Person>
</rdf:RDF>
```

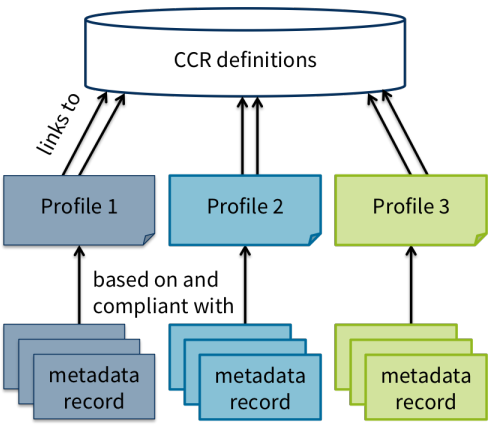
Dzięki użyciu unikalnej przestrzeni nazw parser tego źródła kodu będzie w stanie właściwie zinterpretować prezentowaną w nim treść. Oczywiście w skład standardu wchodzi większa liczba atrybutów i właściwości niż te zaprezentowane w przykładzie. Za ich pomocą można zapisać różne informacje o osobach i relacjach między nimi.

2.2.6. CMDI

CMDI (ang. *Component MetaData Infrastructure*) jest koncepcją rozwiązania pozwalającego na rejestrowanie i używanie elementów metadanych do opisywania zasobów informacji pochodzących z różnych dziedzin, opracowaną i wdrożoną przez konsorcjum CLARIN [8]. W koncepcji tej główną rolę odgrywają sza-

blony, które po zarejestrowaniu mogą służyć do definiowania profili metadanych, które z kolei służą za bazę do opisywania zasobów. W ten sposób rozwiązano problem złożoności niektórych standardów, jak np. Dublin Core – zamiast tworzyć jeden profil zawierający kilkadziesiąt własności zdecydowano się na tworzenie wielu profili, które można definiować i dobierać w zależności od potrzeb, a więc opisywać zasoby w sposób jednolity i minimalny, a jednocześnie wystarczający dla danej klasy zasobu.

Na rysunku 2.2 przedstawiono model wykorzystywany w CMDI. Tworzone zestawy metadanych bazują na zarejestrowanych wcześniej profilach zapisanych w składni języka XML. Dodatkowo, każdy z tworzonych rekordów metadanych, prócz pól definiujących właściwości i wartości, zawiera również odnośnik do profilu, na podstawie którego został utworzony.



Rys. 2.2: Model CMDI (źródło: [8])

2.2.7. Szablon XSL

XSL (ang. *EXtensible Stylesheet Language*) to, w wolnym tłumaczeniu „rozszerzalny język arkuszy styli” [9]. Można powiedzieć, że XSL jest tym samym dla XML, czym CSS dla HTML. Wyjaśniając to dokładniej: język XML w odróżnieniu od np. języka HTML nie posiada predefiniowanych znaczników. Nazwy i znaczenie poszczególnych węzłów i atrybutów w dokumentach XML określa sam autor. Może to prowadzić do problemów podczas analizy zawartości takich dokumentów. Rozwiązaniem tego problemu dostarcza język XSL. Za jego pomocą można zadeklarować sposób reprezentacji danych w plikach XML oraz reguły ich przekształcania. W skład rodziny języka XSL wchodzą 4 główne elementy:

- XSL - słownik opisujący formatowanie dokumentów,
- XSLT - język służący przekształcaniu dokumentów XML,
- XPath - język opisujący dostęp i sposób odwoływania się do elementów dokumentów XML,
- XQuery - język zapytań.

Skupiając się na języku XSLT – pozwala on na przetłumaczenie dokumentów zapisanych w języku XML na dowolny inny format zgodny z jego składnią, jak chociażby XHTML. W celu przekształcenia jednego dokumentu XML w inny należy oczywiście zdefiniować „szablon", który stanowi zbiór reguł opisujących sposób, w jaki dokument na wejściu ma zostać potraktowany i jak ma wyglądać dokument wynikowy. XSLT to prosty język funkcyjny, pozwala jednak na definiowanie własnych funkcji, zmiennych, szablonów czy używanie szeregu wbudowanych komend i instrukcji pozwalających na wykonywanie prostych operacji. Sam plik zawierający definicje transformacji musi być poprawnie sformatowanym dokumentem XML.

2.3. Implementacja

W celu przetestowania opisanych technologii w praktyce stworzono projekt niewielkiej aplikacji. Jej głównym zadaniem jest analiza zasobów internetowych i wyłuskiwanie z kodu źródłowego stron www zawartych tam metadanych, ich wizualizacja oraz zapis przy użyciu kilku różnych notacji.

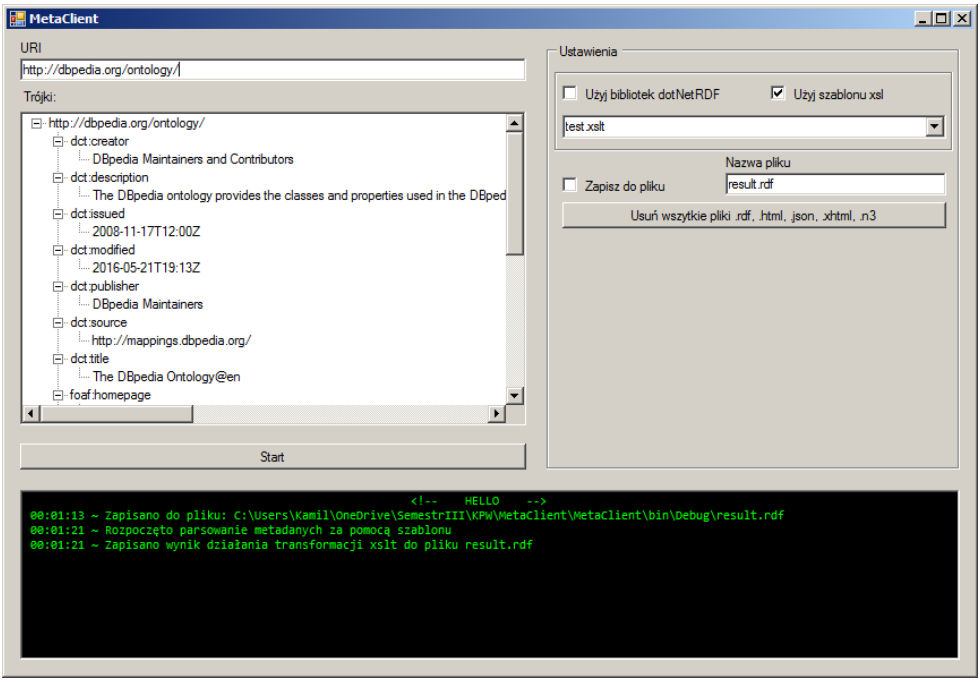
2.3.1. Zastosowane narzędzia i wykorzystane biblioteki

Do rozwoju aplikacji wykorzystano środowisko Visual Studio 2017 z językiem C# oraz bibliotekami WinForms. Wybór ten podyktowany był oferowanymi przez to środowisko ułatwieniami w tworzeniu graficznych interfejsów użytkownika. Ze względu na specyfikę zadania pomocnym okazało się użycie kilku opisanych dalej bibliotek, czy też raczej pakietów nie wchodzących w skład środowiska Visual Studio 2017. Są one dziełem społeczności programistów i zostały udostępnione w Internecie na wolnej licencji.

Źródła stworzonego rozwiązania uzupełniono plikami użytych pakietów wraz z plikami tekstowymi zawierającymi postanowienia licencyjne (w razie ich usunięcia z projektu zostaną one automatycznie pobrane podczas kompilacji przez menadżera pakietów NuGet – o ile taka opcja nie została wcześniej zablokowana w środowisku deweloperskim).

Spośród użytych bibliotek wartymi opisania są:

- `dotNetRDF` – jest to otwarta biblioteka służąca w głównej mierze do operowania na plikach RDF. Zawiera wiele przydatnych struktur pozwalających w łatwy sposób przechowywać, edytować czy też tworzyć pliki zawierające metadane w wielu różnych formatach, jak RDF, XML czy JSON.
- `VDS.Common` oraz `Newtonsoft.Json` – również udostępniane na darmowej licencji biblioteki, które w swoich działaniach wykorzystuje pakiet `dotNetRDF`, tak więc ich obecność jest wymagana do poprawnego działania aplikacji,
- `Microsoft.Xml.SgmlReader` – biblioteka służąca do operowania na plikach xml. Dzięki niej możliwe było „naprawienie" pliku html stanowiącego odpowiedź odpytywanej strony internetowej tak, aby spełniał on zasady formatowania plików xml (co było konieczne do przeprowadzenia transformacji za pomocą szablonu xls).



Rys. 2.3: Gotowa aplikacja w C#

- `HtmlAgilityPack` – jest to biblioteka, której główną zaletą jest mnogość implementacji dających możliwość przesyłania i odbierania pakietów poprzez sieć. Tutaj posłużyła ona odpytaniu docelowej strony internetowej.

Wszystkie te biblioteki są dostępne na zasadach licencji *open-source*. Można je pobrać za pomocą menadżera pakietów NuGet dostępnego w środowisku Visual Studio. Każdą z nich dołączano do projektu w formie biblioteki ładowanej dynamicznie `.dll` (bez kompletu tych bibliotek aplikacja nie będzie funkcjonować). Aby zapobiec ewentualnym niezgodnościom przy zmienianie kodu źródłowego bibliotek przez ich twórców w plikach projektu zapisano nazwy użytych bibliotek wraz z ich wersjami.

2.3.2. Projekt aplikacji

Na rysunku 2.3 przedstawiono wygląd interfejsu użytkownika powstałej aplikacji. Sam interfejs jest stosunkowo prosty: po lewej znajduje się pole na adres URL analizowanego zasobu oraz przycisk START służący do uruchamiania algorytmów parsowania danych o analizowanym zasobie. W polu tekstowym widocznym na dole wyświetlane są podstawowe komunikaty. Aplikację wyposażono w mechanizm obsługi wyjątków. Dzięki niemu udało się zabezpieczyć użytkownika przed wprowadzeniem złych danych oraz dostarczyć metodę wyświetlanie informacji o niepowodzeniu podejmowanych akcji (np. przy łączeniu się z daną witryną czy też podczas parsowania źle sformatowanych danych). Dodatkowo

na interfejsie przewidziano miejsce na wyświetlenie „trójek” w postaci drzewa. Ze względu na dość duży koszt generowania takiej reprezentacji danych zdecydowano, że będą one w niej pokazywane jedynie w przypadku wybrania opcji użycia pakietu `dotNetRDF` (pakiet ten zawiera obsługę struktur przechowujących dane w sposób pozwalający na łatwe operowanie nimi).

Użytkownik ma możliwość wybrania nazwy pliku docelowego, metody analizy danych czy też użytego szablonu xsl (w sekcji po prawej). Oczywiście celem projektu nie było tworzenie uniwersalnego rozwiązania, dającego się zastosować produkcyjnie. Skupiono się w nim głównie na implementacji podstawowych funkcji. Dlatego w implementacji pominięto wiele aspektów, które mogłyby ułatwić pracę. Stąd też wynika zastosowany zapis plików wyjściowych do bieżącego katalogu oraz potraktowanie plików umieszczanych w folderze `Szablony` jako wzorców szablonów.

Na wymienionym wcześniej rysunku można dostrzec drzewo wygenerowane dla przykładowej strony. Dane w postaci trójek zostały tutaj posortowane oraz pogrupowane. Gdy dany zasób wymagał użycia predykatów o tym samym znaczeniu (jak na przykład konieczność wypisania wielu autorów za pomocą predykatu `dbc:author`) poszczególne węzły zostały połączone do jednej gałęzi w celu zmniejszenia rozmiaru generowanego drzewa.

Program powstał z założeniem możliwości zapisu danych wynikowych w różnych formatach. Należy zwrócić uwagę na fakt, iż szablon stylów przygotowano jedynie dla transformacji odpowiedzi witryny z postaci HTML do pliku RDF. Jest to jedyna dopuszczalna postać plików wyjściowego. W przypadku wykorzystania pakietu `dotNetRDF` do dyspozycji mogłoby trafić dużo więcej opcji (pakiet ten obsługuje wiele formatów zapisu danych).

2.3.3. Szablon stylów XSLT

Na potrzeby zadania sporządzono prosty arkusz stylów pozwalających na zapis metadanych otrzymanych z serwisu `DBpedia` [10] w formacie RDF. Obsłużono przy tym tylko najbardziej popularne standardy metadanych jak DC czy FOAF. W przypadku pojawienia się innych elementów metadanych szablon może nie zadziałać poprawnie. Gdyby to się stało, użytkownik zostanie o tym poinformowany. W polu tekstowych aplikacji zwrócona zostanie informacja o napotkaniu niezadeklarowanej przestrzeni nazw. Sytuację można będzie naprawić uzupełniając dyrektywy zapisane na początku pliku o dany standard i przypisaną mu przestrzeń nazw.

Poniżej opisano budowę pliku wykorzystywanego przez aplikację, zawierającego definicje transformacji xsl.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
          xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
          xmlns:x="http://www.w3.org/1999/xhtml"
          xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
          xmlns:foaf="http://xmlns.com/foaf/0.1/">
```

Plik powinien się zaczynać dyrektywami definiującymi używany preprocesor transformacji oraz definicjami przestrzeni nazw dla używanych standardów.

```
<xsl:template match="/">
  <rdf:RDF>
    <xsl:apply-templates/>
  </rdf:RDF>
</xsl:template>
```

XLS jest językiem funkcyjnym, w którym programista pisze reguły, czy też szablony, definiujące sposób przetwarzania danych wejściowych na dane wyjściowe. Gotowy skrypt w tym języku powinien składać się ze zbioru szablonów, które będą regulowały jak przebiegać ma przetwarzanie poszczególnych elementów pliku wejściowego – w tym przypadku instrukcja `match="/"` odnosi się do węzła głównego dokumentu.

```
<xsl:template name="metadane">
  <xsl:for-each select="x:td/x:ul/x:li/x:span/x:span">
    <xsl:element name="{@property}">
      <xsl:copy-of select="@*[name()!='property']"/>
      <xsl:value-of select="normalize-space(current())"/>
    </xsl:element>
  <xsl:text>&#10;</xsl:text>
  </xsl:for-each>
</xsl:template>
```

Powyżej przedstawiono część instrukcji pozwalających na przetworzenie danych zawartych w tabeli. Jak widać nie są to instrukcje bardzo skomplikowane. Ze względu na małą intuicyjność języka mogą jednak sprawić kłopot. Być może z tego powodu, oraz kłopotów w debugowania dokonujących się transformacji, język XSL jest stosunkowo słabo wykorzystywany przez producentów IDE (ang. *integrated development environment*).

Podczas redagowania reguł należy również pamiętać, iż w przypadku nieobsłużenia jakiegoś elementu zostanie on potraktowany w sposób standardowy, co wiąże się z wypisaniem na wyjście jego zawartości tekstowej. W tym przypadku możliwe jest napisanie szablonu, który pozwala na pominięcie tekstu niepowiązanego z analizowanymi metadanymi (w przykładzie zamieszczonym poniżej widnieje reguła pozwalająca pominąć nieistotne z punktu widzenia metadanych elementy źródłowej strony html).

```
<xsl:template match="x:div[starts-with(@class, 'navbar')] |
           x:div[starts-with(@class, 'text-muted')] |
           x:div[starts-with(@class, 'page-resource-uri')] |
           x:p[starts-with(@class, 'lead')] |
           x:title |
           x:h1[starts-with(@id, 'title')]">
</xsl:template>
```

Stworzony szablon przetestowano na kilku witrynach w domenie [10]. Obsłużył on wszystkie przypadki wspierane przypadki. Jednakże nie można go było zastosować do wszystkich stron (co wynikało z opisanych ograniczeń w rozpoznawanych elementów z różnych przestrzeni nazw).

2.3.4. Rezultaty projektu

Wśród rezultatów projektu można wymienić dwie rzeczy: i) rozpoznanie możliwości dostarczanych przez wykorzystane biblioteki (dzięki nim można korzystać z wielu różnych formatów wyjściowych); ii) dostarczenie arkusza XSL pozwalającego pozyskiwać z dokumentów HTML dane w postaci RDF.

Oczywiście pliki generowane przez stworzoną aplikację różnią się od plików generowanego przez profesjonalne pakiety do ekstrakcji metadanych. Przechodzą jednak proces walidacji (mimo kilku komunikatów sugerującego wystąpienie błędów) oraz dostarczają odnalezione trójki.

Do trudniejszych zadań podczas implementacji było takie sformatowanie dokumentów wyjściowych, aby wyglądały one przyjaźnie dla oka. Niestety sposób generowania plików wynikowych nie pozwala na jego automatyczne formatowanie wewnątrz kodu aplikacji. W efekcie plik wynikowy był pozbawiony wcięć, a gdzieniegdzie zdarzały się nadmiarowe białe znaki. Należy podkreślić, że to jednak nie wpływało na samą logiczną strukturę dokumentu. Przyznać jednak należy, że ostateczny wynik projektu nie jest do końca zadowalający. Pisanie szablonów XSL nie należy do łatwych czynności. Trudno jest znaleźć przyjazne środowisko, które służyłoby wsparciem i pozwalało na szerszą walidację i debugowanie napisanego kodu zamiast dawać jedynie informacje o sukcesie lub porażce. Co prawda wspomniana biblioteka `dotNetRDF` rozwiązuje zadanie selekcji informacji o danym zasobie w sposób programowy (zawiera bardzo pokaźny zasób klas i funkcji służących do parsowania, konwertowania i przetrzymywania zebranych danych). Jednakże założeniem projektu było rozwiązanie tego zadania poprzez szablon stylów.

Być może lepszym założeniem byłoby oparcie projektu na oprogramowaniu `Apache Any23`, którego kod źródłowy oraz wszelka dokumentacja znajdują się na stronie [11]. W zasobach tym można odczytać znaczenie samej nazwy `Any23` – ang. *anything to triples*. Oprogramowanie to powstało z użyciem języka Java i jest udostępnione na licencji `Apache v2.0`. W jego skład wchodzą zarówno biblioteki przeznaczone dla programistów, jak również narzędzia uruchamiane z linii poleceń oraz serwis webowy. Przeszukując repozytorium projektu można natrafić na plik `rdfa.xslt` stanowiący szablon stylów przeznaczony właśnie do celów parsowania dokumentów w formacie RDFa, a tym samym wydobywania z nich informacji w postaci trójek. Sam szablon jest bardzo złożony i rozbudowany – w przeciwieństwie do tego utworzonego w ramach realizowanego przez autora pracy projektu. Jest również bardziej uniwersalny i pozwala on na przekształcanie każdego dokumentu zawierającego informacje w formacie RDFa.

Porównując jednak wyniki, uzyskane za pośrednictwem szablonu stanowiącego część projektu `Any23` oraz tych uzyskanych przy użyciu bibliotek `dotNetRDF` można zauważyć, iż nie są one jednakowe. Pierwszy z wymienionych projektów zwraca wynik w postaci szeregu niezależnych trójek, traktując każdą z nich jako osobne drzewo. Drugi zaś je „grupuje" – wszystkie trójki opisujące jeden obiekt mieszczą się wewnątrz tego samego elementu, co tworzy strukturę bardziej czytelną. To jednak nie stanowi wielkiego problemu (standard RDF umożliwia łatwą konwersję danych do innych notacji, dzięki czemu możemy przedstawić opisany

graf w bardziej czytelnej formie). Drugą różnica polega na tym, iż wynik działania szablonu pochodzącego z projektu Any23, prócz odnalezienia trójek opisujących dany zasób, zawiera w sobie również kilka pustych węzłów, które w swojej strukturze w znacznikach <HEAD> zawiera witryna DBpedia, które wykorzystuje na swoje potrzeby i nie stanowią one opisu danego zasobu.

2.4. Podsumowanie

Wizja Semantycznego Internetu wygląda całkiem obiecująco. Być może, że dla przeciętnego użytkownika komputera jest jeszcze zbyt abstrakcyjna i odległa, jednakże już teraz zauważalne są skutki jej wdrażania, chociażby w wynikach pozycjonowania stron w wyszukiwarkach. Biorąc pod uwagę problemy wyszukiwania informacji zastosowanie ujednoliconego systemu opisu zasobów wydaje się świetnym ich rozwiązaniem. Może to zapewnić dobre podstawy do wdrażania narzędzi do przechowywania i, co najważniejsze, ukierunkowanego przeszukiwania zbiorów danych. Tego typu problemy próbowano rozwiązywać już wcześniej, chociażby w bibliotekach, stosując katalogi. Dzięki nim można było indeksować najważniejsze dane bibliograficzne dotyczące zbiorów oraz informacje o miejscu ich składowania.

W ramach prac nad projektem zgłębiono wiedzę o sposobie przechowywania i magazynowania metadanych. W trakcie jego realizacji powstała również prosta aplikacja spełniająca nakreślone założenia. Dzięki niej można pozyskiwać metadane z serwisu DBpedia. Opracowany szablon transformacji nie jest jednak uniwersalny i może zawieść przy analizie dokumentów HTML o strukturze innej niż oczekiwana.

Literatura

[1] M. Nahotko. Metadane i ontologie w systemach zarządzania wiedzą, Marzec 2008. http://skryba.inib.uj.edu.pl/~nahotko/metadane/10iinibuj.pdf, [dostęp dnia 13.06.2017].

[2] M. Zieliński. Standardy metadanych: Dublin Core, Maj 2013. http://www.pilsudski.org/portal/pl/nowosci/blog/441-standardy-metadanych-dublin-core, [dostęp dnia 13.06.2017].

[3] RDF Working Group. Resource Description Framework (RDF), Luty 2014. https://www.w3.org/RDF/, [dostęp dnia 13.06.2017].

[4] I. Davis, L. Dodds. *Linked Data Patterns A pattern catalogue for modelling, publishing, and consuming Linked Data.* Maj 2012. http://patterns.dataincubator.org/book/linked-data-patterns.pdf, [dostęp dnia 13.06.2017].

[5] Resource Description Framework, Listopad 2016. Artykuł redagowany przez społeczność https://pl.wikipedia.org/wiki/Resource_Description_Framework, [dostęp dnia 13.06.2017].

[6] B. Adida, I. Herman, M. Sporny, M. Birbeck. RDFa 1.1 Primer - Third Edition. Rich Structured Data Markup for Web Documents. *W3C Working*

Group Note, Marzec 2015. https://www.w3.org/TR/rdfa-primer/, [dostęp dnia 13.06.2017].

[7] L. Dodds. An introduction to FOAF. *XML.com*, Luty 2004. https://www.xml.com/pub/a/2004/02/04/foaf.html, [dostęp dnia 13.06.2017].

[8] CLARIN. Strona projektu https://www.clarin.eu/, [dostęp dnia 13.06.2017].

[9] J. Kobusiński. XSL w przykładach, Czerwiec 2013. http://www.cs.put.poznan.pl/jkobusinski/xslt.html, [dostęp dnia 13.06.2017].

[10] DBpedia. Strona projektu http://dbpedia.org, [dostęp dnia 13.06.2017].

[11] Apache Any23. Strona projektu https://any23.apache.org/, [dostęp dnia 13.06.2017].

ROZDZIAŁ 3

WYKORZYSTANIE DALMIERZY LASEROWYCH DO NAWIGACJI KWADROKOPTEREM W ZAMKNIĘTYCH POMIESZCZENIACH

P. Jachimowski

W rozdziale dokonano przeglądu zastosowań laserowych czujników odległości w sterowaniu latającymi modelami. W rozważanych przypadkach wyróżniono maszynę typu kwadrokopter. Na niej przeprowadzono testy zainspirowane omówionymi rozwiązaniami.

3.1. Wstęp

W ciągu ostatnich kilkunastu lat dużą popularność zyskały zdalnie sterowane modele wielowirnikowe (nazywane też dronami). Wzbudziły one zainteresowanie nie tylko społeczności hobbystów, ale również zwykłych osób, które na co dzień nie poświęcają technicznym zagadnieniom zbyt wiele czasu. Rozwój elektroniki umożliwił rozwój coraz lepszych i tańszych konstrukcji, dlatego widok kwadrokoptera w supermarkecie obok półki z pieczywem nie budzi już sensacji. Niestety, ścieżka komercjalizacji modeli wielowirnikowych zorientowana na opanowanie szerokiego rynku odbiorców doprowadziła do ryzyka i niebezpiecznych zdarzeń. Konsekwencje można zaobserwować na popularnych stronach internetowych, w postaci setek filmów przedstawiających mniej lub bardziej spektakularne wypadki z udziałem wielowirnikowców. Coraz częściej osoby niedoświadczone próbują sterować tego typu konstrukcjami latającymi przekonanie, że to tylko proste, bezpieczne zabawki. Jest to jeden z powodów, dla którego wciąż opracowywane są nowe, bezpieczniejsze sposoby sterowania dronami z wykorzystaniem szerokiej gamy czujników, skanerów i kamer. Niestety główną wadą takich rozwiązań jest cena, podczas gdy w wielu przypadkach być może wystarczyłoby kilka tanich i odpowiednio zamontowanych dalmierzy laserowych.

W niniejszym rozdziale podjęto próbę przeprowadzenia analizy, jak wiele informacji można uzyskać na podstawie danych z prostych czujników odległości i jak je skutecznie wykorzystać do nawigacji najpopularniejszym typem drona cy-

wilnego, czyli kwadrokopterem. W podrozdziale 3.2 opisano podstawowe aspekty sterowania maszyną tego typu. Następnie, w podrozdziale 3.3, przedstawiono różne możliwe zastosowania dalmierzy laserowych zamontowanych na dronie. Niektóre z nich wybrano do testów na rzeczywistym kwadrokopterze, których wyniki opisano w podrozdziale 3.4. Podczas redakcji rozdziału wykorzystano pojęcia anglojęzyczne spopularyzowane wśród konstruktorów modeli latających.

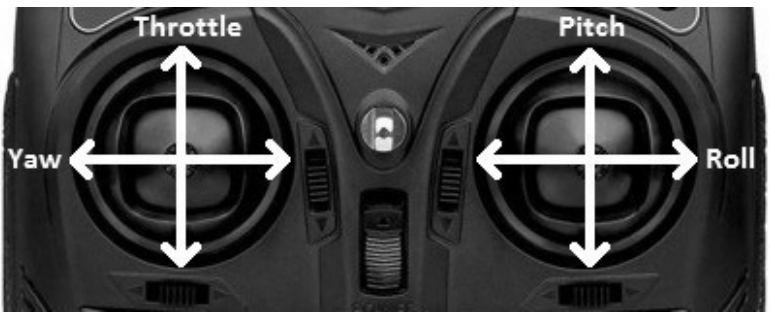
3.2. Sterowanie

3.2.1. Aparatura

Modele latające, mimo swojej różnorodności, w większości przypadków sterowane są za pomocą aparatury RC (ang. *radio/remote control*) składającej się z nadajnika radiowego, dwóch drążków, zestawu przycisków, przełączników i potencjometrów oraz dodatkowych elementów, takich jak wyświetlacze LCD lub ekrany dotykowe. Obowiązuje zasada: im droższe urządzenie, tym więcej funkcji oferuje. Zaawansowana aparatura powinna teoretycznie obsłużyć niemal wszystkie możliwe modele latające, począwszy od styropianowych szybowców, skończywszy na modelach odrzutowców.

Prócz wysyłania komend, możliwe jest również odbieranie danych telemetrycznych, takich jak wysokość nad poziomem morza, ciśnienie czy temperatura otoczenia. Może to być także obraz z kamery bądź dane ze skanerów i czujników odległości. W zależności od wyposażenia urządzenia oraz jego oprogramowania, odczyty z różnych dodatkowych czujników są automatycznie wykorzystywane do korekcji lotu, bądź traktowane jedynie jako informacja dla użytkownika. Bardziej doświadczony pilot w większości przypadków zdecyduje samodzielnie nawigować maszyną, choć niekiedy jest to sporym wyzwaniem. Mimo, że dla większości konstrukcji możliwe jest przypisanie elementom aparatury odpowiednich komend sterujących, często potrzeba wielu godzin treningu, aby nabrać właściwych nawyków i umiejętnie pilotować.

Najczęściej spotykany układ w jakim zrealizowane jest sterowanie za pomocą aparatury ogranicza się do przypisania lewemu drążkowi regulacji mocy (ang. *throttle*) i *yaw*, a prawemu *pitch* i *roll* (rys. 3.1). Zadawanie odpowiednich



Rys. 3.1: Podstawowy układ sterowań na aparaturze

kątów RPY prowadzi do zmiany orientacji kwadrokoptera wokół osi X, Y i Z, co skutkuje przemieszczaniem się maszyny w wybranym kierunku.

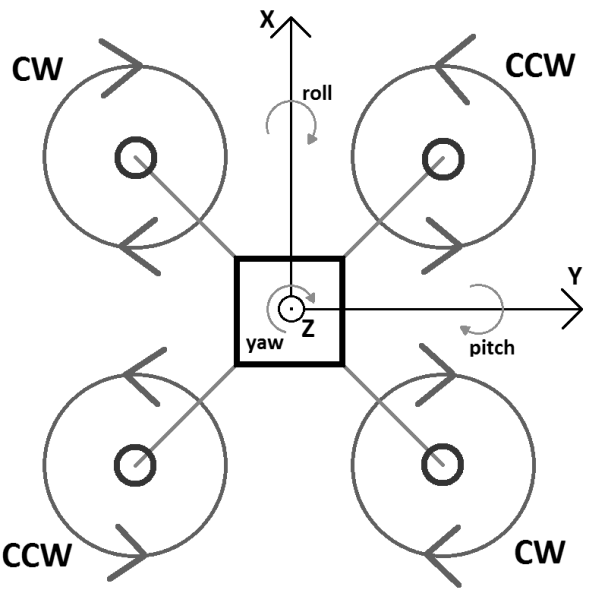
W Internecie można znaleźć wiele obszernych poradników, które doskonale wyjaśniają omawiane aspekty sterowania [1, 2].

3.2.2. Konstrukcja kwadrokoptera

Do zrozumienia podstaw działania kwadrokoptera, potrzebna jest wiedza o tym jak urządzenie jest zbudowane. Podstawowym elementem jest rama, na której montowane są podzespoły, takie jak:

- silniki BLDC (ang. *BrushLess Direct-Current motor*),
- regulatory ESC (ang. *Electronic Speed Control*),
- kontroler lotu,
- odbiornik radiowy i/lub nadajnik (często zintegrowane z kontrolerem lotu),
- bateria litowo-polimerowa.

Ważną kwestią dotyczącą sterowania jest umiejscowienie osi przyjętego układu współrzędnych względem ramion kwadrokoptera, które może być różne w zależności od sposobu montażu kontrolera lotu i znajdujących się na nim czujników. Jeżeli osie X i Y są równoległe do ramion to określamy takie ustawienie jako konfigurację „+”. Inną opcją jest konfiguracja „×” przedstawiona na rysunku 3.2. W tym wariancie osie X i Y są przesunięte względem ramion o 45°. Oznacza to konieczność modyfikacji prędkości wszystkich czterech silników w celu rotacji *roll* lub *pitch*. Wybór konfiguracji nie ma wpływu na parametry lotu kwadrokoptera, a jedynie na programową realizację sterowania.



Rys. 3.2: Schemat poglądowy kwadrokoptera w konfiguracji „×”

W każdym wielowirnikowcu, istotną kwestią jest odpowiedni montaż silników z uwzględnieniem kierunku obrotów. W kwadrokopterze, aby dron nie wykonywał niepożądanych rotacji wokół własnej osi, dwa silniki powinny obracać się w stronę przeciwną do pozostałych (równoważenie momentów sił). Oznacza to konieczność zastosowania dwóch rodzajów śmigieł:

- CW (ang. *clockwise*) – dla silników obracających się zgodnie z kierunkiem wskazówek zegara,
- CCW (ang. *counter-clockwise*) – dla silników obracających się przeciwnie do kierunku wskazówek zegara.

Coraz częściej producenci oferują silniki dedykowane dla wielowirnikowców i występujące również w dwóch wariantach w zależności od kierunku obrotów. Różnią się wtedy gwintowaniem wału na który przykręcana jest piasta, co zapobiega odkręcaniu się śmigieł podczas lotu. Przykładowo jeśli silnik obraca się zgodnie z kierunkiem wskazówek zegara to piasta mocująca śmigło przykręcana jest w stronę przeciwną.

3.3. Zastosowania laserowych czujników odległości

Podstawowe czujniki, w które wyposażony jest prawie każdy kwadrokopter to przede wszystkim akcelerometr, żyroskop i magnetometr. Dostarczają one informacji o orientacji modelu, jednak nie przekazują żadnych danych o otoczeniu, w którym urządzenie znajduje się podczas lotu. W tym przypadku użyteczne może się okazać wykorzystanie kamery przekazującej obraz użytkownikowi, który na jego podstawie może podjąć stosowne akcje. Jednak w rozsądnym przedziale cenowym takie rozwiązanie rzadko kiedy się sprawdza. Ponadto zależne jest od szybkości z jaką zareaguje pilot, gdyż kontrolery lotu najczęściej nie mają dostatecznej mocy obliczeniowej, aby móc przeprowadzić analizę obrazu.

Inną opcją jest wykorzystanie ultradźwiękowego czujnika odległości, jednak jego niewielka dokładność i czas trwania pomiaru wyklucza większość ciekawszych przypadków użycia.

Dość rozsądne zatem wydaje się wykorzystanie dalmierzy laserowych, które mogą mieć szereg zastosowań w nawigacji kwadrokopterem:

Ochrona przed kolizją – informacje pochodzące z dalmierzy laserowych mogą posłużyć do ostrzegania użytkownika o potencjalnie niebezpiecznych manewrach, mogących zakończyć się rozbiciem kwadrokoptera o przeszkodę. Ze względu na szybszy czas reakcji komputera pokładowego, maszyna może również samodzielnie modyfikować swoje położenie i orientację w celu uniknięcia kolizji [3].

Kalibracja czujników – czujniki znajdujące się na kontrolerze lotu cechuje pewna niedokładność, która jest źródłem błędu zwanego dryfem. Dotyczy to głównie akcelerometrów i żyroskopów, które należy co jakiś czas kalibrować przed startem, bądź w takcie lotu wykorzystując wiedzę o bezwzględnym położeniu pochodzącą przykładowo z nadajnika GPS. Niestety w zamkniętych pomieszczeniach taka wiedza jest zwykle niedostępna i kwadrokopter będący

w trakcie lotu, przy braku kontroli użytkownika, dość szybko uderzy w przeszkodę, najczęściej ulegając uszkodzeniu. W tej sytuacji bardzo pomocna w minimalizacji dryfu może okazać się informacja pochodząca z dalmierzy laserowych. Wystarczy aby maszyna wykryła przeszkodę (na przykład ścianę) i za pomocą odpowiedniego sterowania tak korygowała swoje położenie, aby odległość od obiektu była stała (a raczej zmienna w jak najwęższym zakresie). Kąt rotacji powinien w takiej konfiguracji wynosić 0° (dla osi stowarzyszonej z danym dalmierzem), a wszelkie różnice stanowią tak zwany *offset*, który należy zapamiętać jako wartość korygującą odczyty z czujników inercyjnych.

Autonomiczne lądowanie – wykorzystując czujniki odległości możliwe jest wykonanie przez kwadrokopter samodzielnego, bezpiecznego lądowania. W takim przypadku najkorzystniej jest, gdy urządzenie posiada 4 czujniki odległości skierowane w dół, zamontowane możliwie blisko nóżek, tak aby mieć pewność uzyskania styku z podłożem.

Proste mapowanie otoczenia – zapamiętując informacje o położeniu różnych obiektów, kwadrokopter może generować uproszczoną mapę otoczenia (niestety dokładność będzie zdecydowanie niższa niż z wykorzystaniem skanerów, takich jak w [4]). Urządzenie może wykorzystać uzyskana wiedzę w kolejnych przelotach i tak modyfikować swoje położenie, aby być w jak największej odległości od przeszkód.

Wyszukiwanie drogi w labiryncie – kilka czujników (co najmniej 6) wystarcza, aby dron z powodzeniem wyszukiwał wyjścia z kompleksów pomieszczeń lub tuneli. Analogią jest w tym przypadku pokonywanie labiryntu przez roboty typu micromouse przeniesione do przestrzeni trójwymiarowej. Zagadnienie jest dość skomplikowane, jednak w najprostszym przypadku wystarczy, aby urządzenie cały czas podążało wzdłuż ścian znajdujących się po wybranej stronie.

3.4. Testy

W badaniach zdecydowano się sprawdzić możliwość kalibracji czujników położenia, w szczególności akcelerometru dostarczającego informacji o przyspieszeniach w trzech osiach. Metoda została częściowo opisana w rozdziale 3.2. Opiera się na utrzymywaniu możliwie stałej odległości od przeszkód (najłatwiej za pomocą regulatorów PID) w celu wyznaczenia offsetów dla osi X i Y (kąty *pitch* i *roll*).

3.4.1. Kalibracja czujników inercyjnych

Kalibracja jest nieodłączną części pracy z modelami wyposażonymi w czujniki inercyjne. Na błąd odczytów ma wpływ wiele czynników takich jak sposób montażu, zakłócenia elektromagnetyczne, temperatura lub ciśnienie. Powodują one, że wektor przyśpieszenia, wyznaczany z odczytów na osiach X, Y i Z, rysuje w przestrzeni trójwymiarowej przesuniętą elipsoidę. W idealnym przypadku chcielibyśmy, aby zbiór pomiarów tworzył kulę o środku w punkcie (0, 0, 0). Oznaczałoby to, że zadanie z aparatury zerowych kątów *pitch* i *roll* skutkuje stałą pozycją względem ziemi (zakładając stałą wysokość). Niestety w praktyce, dość

szybko urządzenie zacznie niekontrolowanie zbaczać w losowym kierunku. Znajdując się w zamkniętym pomieszczeniu, kolizja jest niemal gwarantowana.

Sytuacja zmienia się kiedy mamy na pokładzie dalmierze laserowe, które mogą ostrzec o napotkanych obiektach. W takim przypadku, kontroler lotu może podjąć stosowne akcje w celu uniknięcia kolizji, czyli w najprostszym przypadku, wyznaczyć nowy kurs w stronę przeciwną do przeszkody. Krokiem bardziej zaawansowanym jest sprawdzenie o ile należałyby zmodyfikować kąt rotacji wokół osi stowarzyszonej z dalmierzem wykrywającym obiekt, aby odległość do tego obiektu była stała. Wyznaczoną w ten sposób wartość należy dodać do przyszłych odczytów orientacji.

3.4.2. Platforma testowa

Przedstawioną metodę zdecydowano się przetestować na rzeczywistym kwadrokopterze. Badania przeprowadzono z wykorzystaniem 6 czujników Sharp GP2Y0A02 pracujących w zakresie 20-150 cm (rys. 3.3).

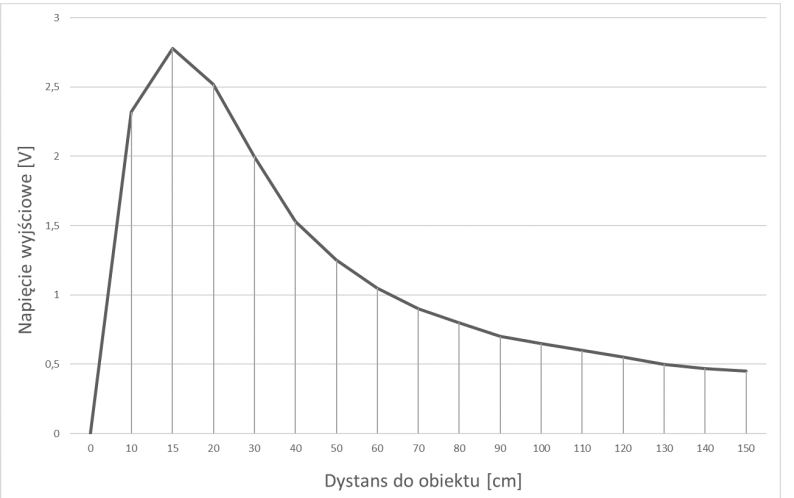
Rys. 3.3: Dalmierz laserowy Sharp GP2Y0A02

Wybrane czujniki generują nowy odczyt co 5 ms. Wyjściem dalmierzy jest analogowy sygnał napięciowy, zależny od wyznaczanej odległości jak pokazano na rysunku 3.4. Rozmieszczenie czujników odległości na ramie kwadrokoptera przedstawiono na rysunku 3.5.

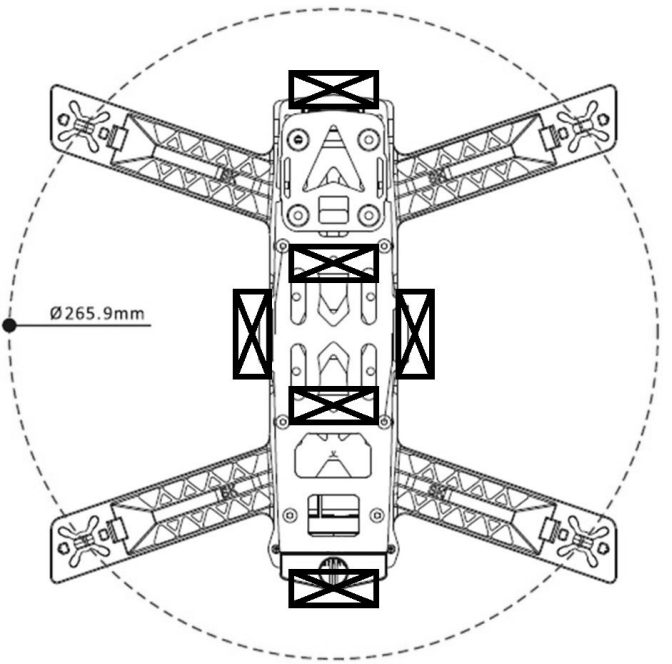
Konstrukcja, na której zamontowano czujniki, jest częścią autorskiego projektu, w ramach którego zbudowano kontroler lotu przystosowany do odczytu wartości napięcia z 6 kanałów przewrotnika analogowo-cyfrowego ADC (ang. *analog to digital converter*). Fizyczny model zaprojektowanego kwadrokoptera przedstawiono na rysunku 3.6).

Prawidłowe przeprowadzenie kalibracji za pomocą dalmierzy wymagało przyjęcia następujących założeń:

- pełna znajomość orientacji kwadrokoptera wokół osi Z (taką informację najczęściej pozyskuje się z danych magnetometru),
- stała pozycja obiektów wykrywanych przez dalmierze,
- zadawane kąty *pitch* i *roll* wynoszące 0°.



Rys. 3.4: Zależność napięcia od odległości dla czujnika GP2Y0A02 [5]



Rys. 3.5: Rozmieszczenie dalmierzy laserowych na kwadrokopterze

Ze względu na dużą wrażliwość wybranych dalmierzy na zakłócenia pochodzące od pracujących silników (prototypowy charakter konstrukcji), kalibrację wykonano w warunkach statycznych, w których kwadrokopter ustawiono w orientacji imitującej błąd położenia kątowego, z przeszkodami w zasięgu czujników odległości.

Rys. 3.6: Testowa konstrukcja

3.4.3. Wyniki

Dla lepszego zrozumienia uzyskanych wyników, warto znać ogólny schemat zaimplementowanej pętli sterowania kwadrokopterem:

- pobranie danych z czujników inercyjnych,
- wyznaczenie kątów RPY,
- odbiór danych z aparatury (na potrzeby testów, zadawane jest zerowe położenie kątowe),
- pobranie danych z dalmierzy laserowych,
- w razie wykrycia przeszkody modyfikacja zadawanej orientacji (celem jest utrzymanie odczytywanej wartości z dalmierzy na stałym poziomie),
- dostosowanie wartości korygujących,
- regulacja mocy silników (PID).

Uzyskane wyniki są wartościami korygującymi dla czujników i były rozpatrywane w dwóch wariantach. W pierwszym offset stanowiła różnica między 16-bitowymi danymi pobieranymi bezpośrednio z akcelerometru, a wartościami oczekiwanymi dla zadawanej orientacji. Wyznaczone w ten sposób wartości należy dodać do surowych danych z akcelerometru na etapie pobierania danych z czujników inercyjnych.

W drugim zdecydowano się wyznaczać wartości korekcyjne w stopniach i dodawać je na dalszym etapie przetwarzania danych, gdzie z surowych danych z akcelerometru i żyroskopu uzyskiwane są kąty *roll* i *pitch* zgodnie ze wzorami:

$$acc\_roll = \text{atan2}(Y, Z)\frac{180}{\pi}$$

$$acc\_pitch = \text{atan2}(-X, \sqrt{Y^2 + Z^2})\frac{180}{\pi}$$

$$roll = 0.98 \cdot gyr\_roll + 0.02 \cdot acc\_roll$$

$$pitch = 0.98 \cdot gyr\_pitch + 0.02 \cdot acc\_pitch$$

Przedrostek *acc* oznacza kąt wyznaczony z danych samego akcelerometru, a *gyr* z żyroskopu. Mnożenie przez współczynniki 0.98 i 0.02 jest najprostszym przykładem fuzji danych z czujników inercyjnych, określanym jako *filtr kompensacyjny*.

Dzięki testom naziemnym wartości można było odczytywać bezpośrednio z programatora J-Link EDU, podłączonego do pracującego mikrokontrolera znajdującego się na kontrolerze lotu. Zatem możliwe było sprawdzenie poprawności działania programu. Przykładowo, jeżeli zasymulowano 3-stopniowy błąd kąta *roll*, przy stałej odległości od przeszkód, kontroler lotu uznawał, że skoro przy danym wychyleniu pozycja się nie zmienia, to te 3° muszą być błędem. Potwierdzały to kolejne serie obliczeń, na których kwadrokopter dodawał do wyznaczanego kąta *roll* offset o wartości -3°.

3.5. Podsumowanie

Laserowe czujniki odległości stanowią dobrą alternatywę dla znacznie droższych kamer i skanerów. Mają szereg interesujących zastosowań w platformach wielowirnikowych. Przede wszystkim, pozwalają zminimalizować liczbę wypadków, zwłaszcza podczas nawigacji w zamkniętych, niedoświetlonych pomieszczeniach. Ponadto dane wyjściowe z dalmierzy stosunkowo łatwo się odczytuje i przetwarza bez zbytniego obciążania kontrolera lotu.

W bardziej zaawansowanych zastosowaniach, takich jak kalibracja czujników, należy zwrócić szczególną uwagę na prawidłowy montaż czujników, minimalizujący wpływ zakłóceń związanych z pracą silników. Dzięki temu osiągamy dokładność, która jest kluczowa dla poprawy jakości sterowania.

Literatura

[1] 4śmigła. Quadrocoptery – Pierwszy start. 2017. http://4smigla.pl/quadrocoptery-drony-eksploatacja/, [dostęp dnia 30.05.2017].

[2] UAV Coach. How to Fly a Drone. 2017. https://uavcoach.com/how-to-fly-a-quadcopter-guide/#Controls, [dostęp dnia 10.06.2017].

[3] CyrilAnthony777. Quadcopter Collision Avoidance Using Sharp IR Sensors. 2017. http://www.instructables.com/id/Quadcopter-Collision-Avoidance-Using-Sharp-IR-Sens/, [dostęp dnia 24.04.2017].

[4] Vijay Kumar. The Future of Flying Robots. 2017. https://www.youtube.com/watch?v=ge3--1hOm1s, [dostęp dnia 24.04.2017].

[5] SHARP. GP2Y0A02YK0F - Distance Measuring Sensor Unit. 2006. https://www.sparkfun.com/datasheets/Sensors/Infrared/gp2y0a02yk_e.pdf, [dostęp dnia 30.05.2017].

ROZDZIAŁ 4

SELEKCJA ZAWODNIKÓW

K. Kwieciński

W rozdziale omówiono zagadnienia związane z problemem optymalnego doboru zawodników podczas rywalizacji drużynowej na przykładzie gry Dota 2. Opisano wpływ jednostek na wyniki ogółu oraz wskazano na kluczowe aspekty występujące podczas definiowania składu zespołu w perspektywie osiągnięcia końcowego sukcesu. Szczególną uwagę poświęcono przypadkowi selekcji ostatniego zawodnika.

4.1. Wstęp

W grach zespołowych na sukces drużyny składa się wiele czynników. Spośród nich kluczowym jest odpowiedni dobór składu drużyny poprzez selekcję zawodników o odpowiednich umiejętnościach. Zespoły, których skład został odpowiednio dobrany zawsze osiągają lepsze wyniki od zespołów składających z indywidualistów i to nawet wybitnych [1]. Trudno bowiem oczekiwać, by najlepsi zawodnicy od razu stworzyli od razu zgrany kolektyw. Proces budowania dobrej drużyny wymaga czasu potrzebnego na zdobycie doświadczenia we współdziałaniu. Ponadto by osiągnąć szczyt zespołowych możliwości niezbędne jest pokonywanie stawianych przed drużyną wyzwań [2].

W obecnych czasach świat rzeczywisty i wirtualny przenikają się. Coraz częściej współpracuje się nie tylko twarzą w twarz, ale także korzystając z dróg komunikacji zapewnianych przez najnowocześniejsze technologie. D'Souza i Colarelli w artykule [3] porównują kompletowanie zespołów rzeczywistych i wirtualnych. Na podstawie ich badań można stwierdzić, że dobierając drużynę głównym czynnikiem wpływającym na wybór osoby są posiadane przez nią umiejętności, jednak w świecie rzeczywistym niebagatelną wagę odgrywa także jej charakter.

Odpowiednia selekcja zawodników jest problemem złożonym i trudnym. Jedna z popularnych metod wykorzystywanych przy tworzeniu zespołów opiera się na podejściu funkcyjnym [4], czyli takim dobieraniu członków drużyny, żeby dobrze realizowali oni powierzone im zadania. Czytelnie ilustruje to przykład selekcji zawodników drużyny piłkarskiej. Drużyna piłkarska odnosząca sukcesy składa się z zawodników, którzy dobrze wykonują powierzone im zadania. Rolą

trenera jest odpowiedni wybór piłkarzy. Zazwyczaj ma on do dyspozycji więcej graczy niż może desygnować do wyjścia na boisko. Przy kompletowaniu składu selekcjoner kieruje się pozycją, na jakiej występuje zawodnik. Z reguły piłkarz wykazuje naturalne predyspozycje do pełnienia określonej roli, a także posiada preferencje co do pozycji, na których mógłby oraz nie chciałby występować. Ponadto wybór zawodników zależy od przeciwnika, z którym zespół mierzy się w konkretnym spotkaniu. Przykładowo, wiedza że w drużynie rywala występują wysocy, ale powolni obrońcy, może prowadzić do wystawienia niskich, ale szybkich napastników. Konsekwencją takiego wyboru jest większa lub mniejsza przydatność zawodników w zależności od charakteru rozgrywki, mianowicie od proporcji stałych fragmentów gry do zwykłego konstruowania akcji.

4.2. Dota 2

Dota 2 to jedna z najpopularniejszych gier komputerowych typu *multiplayer online*. Została stworzona i wydana w lipcu 2013 roku przez Valve Corporation. Od tego momentu rozegrano w niej ponad dwa miliardy meczów, a dziennie aktywnych jest średnio około 500 tys. użytkowników [5].

4.2.1. Zasady gry

Rozgrywka Rozgrywka opiera się na rozgrywaniu meczów w trybie wieloosobowym, w których dwie pięcioosobowe drużyny mają za cel zniszczenie ufortyfikowanych twierdz przeciwnika. Gracze przed pojedynkiem wybierają:

- zespół (*radiant* lub *dire*), po którego stronie będą walczyć,
- bohatera, którym sterują w trakcie meczu.

Zespoły Zawodnicy tworzą pięcioosobowe drużyny, które mogą być tworzone spontanicznie przed meczem lub składać się ze stałych członków.

Tryby gry Po uformowaniu zespołów konieczne jest wyselekcjonowanie bohaterów, którymi gracze będą sterować podczas meczu. Sposób wyboru postaci różni się w zależności od trybu rozgrywki, spośród których dwa najpopularniejsze to:

- *Captain's Mode* — kapitanowie dwóch drużyn naprzemiennie podejmują decyzję *pick*/*ban*, czyli wybierają bohatera do swojego zespołu bądź odrzucają go z listy dostępnych postaci dla danego meczu.
- *All Pick* — tryb analogiczny do poprzedniego z tym wyjątkiem, że to kolejni zawodnicy, a nie wyłącznie kapitanowie, dokonują selekcji zawodników.

Bohaterowie W grze dostępnych jest 113 bohaterów, spośród których wybierać mogą gracze. Każda postać ma inne umiejętności i statystyki. W trakcie meczu można ulepszyć wojownika wykorzystując zebrane złoto i doświadczenie. Bohaterów dzieli się ze względu na ich główny atrybut: siłę, zręczność, lub inteligencję.

4.2.2. Przykładowy mecz

Przebiegi wszystkich meczów są zapisywane w bazie danych dostępnej na stronie internetowej [6]. Szczegółowe informacje dotyczące rozgrywki, w formacie `.json`, można uzyskać przy pomocy wystawionego API (`https://docs.opendota.com/`). Na listingu 4.1 przedstawiono przykład zapisu rozgrywki, który został uproszczony by pokazać tylko najważniejsze atrybuty opisujące mecz:

- `match_id` — id meczu,
- `game_mode` — tryb meczu,
- `duration` — czas trwania meczu,
- `[dire|radiant]_score` — wynik uzyskany przez zespół,
- `radiant_win` — zwycięzca meczu,
- `picks_bans` — selekcja bohaterów:
  - `is_pick` — wybór bądź odrzucenie bohatera,
  - `hero_id` — id bohatera,
  - `team` — zespół,
  - `order` — numer kolejki, w której dokonano selekcji danego bohatera,
- `players` — tablica zawierająca zawodników:
  - `account_id` — id gracza,
  - `hero_id` — id bohatera, którym gracz sterował,
  - `isRadiant` — przynależność do zespołu,
  - `benchmarks` — statystyki gracza (złoto, doświadczenie, zabicia na minutę).

Listing 4.1: Przykładowy zapis meczu

```
{
  "match_id": 3037172147,
  "game_mode": 2,
  "duration": 1681,
  "dire_score": 6,
  "radiant_score": 25,
  "radiant_win": true,
  "picks_bans": [
    {
      "is_pick": false,
      "hero_id": 107,
      "team": 1,
      "order": 0
    },
  ],
  "players": [
    {
      "account_id": 41288955,
      "hero_id": 50,
      "isRadiant": true,
      "benchmarks": {
        "gold_per_min": { },
        "xp_per_min": { },
        "kills_per_min": { }
      }
    },
  ]
}
```

4.3. Selekcja zawodników na przykładzie gry Dota 2

Zagadnienie selekcji zawodników w grze Dota 2 jest bardzo rozbudowane. Można tutaj wyróżnić dwa problemy:

- selekcję zawodników rzeczywistych,
- selekcję zawodników wirtualnych.

Selekcja zawodników rzeczywistych

W grze rywalizują ze sobą drużyny złożone z różnych osób o zróżnicowanych umiejętnościach. Jedni ludzie wykazują większy talent do gry, podczas gdy inni są tylko przeciętni. Skutkuje to pojawieniem się w rozgrywce „czynnika ludzkiego". Zazwyczaj wraz z większą ilością czasu poświęconej na grę, wyniki osiągane przez daną osobę stają się coraz lepsze.

Gracze, sterując tymi samymi bohaterami, mogą kierować nimi w odmienny sposób. Każdy człowiek posiada pewne preferencje dotyczące używania i rozwijania umiejętności sterowanej przez siebie postaci. W konsekwencji wykorzystanie zdolności bohatera podczas rozgrywki różni się, co skutkuje *de facto* niepowtarzalnością gry tym samym wojownikiem w zależności od zawodnika.

Ponadto każdy człowiek ma pewien charakterystyczny styl gry. Niektórzy preferują intensywny start, a inni motywują się do lepszej gry dopiero pod koniec pojedynku. Część osób męczy się podczas rozgrywki i ich efektywność spada wraz z czasem trwania meczu, a bywają osoby, które potrzebują chwili na rozgrzanie się i osiągają szczyt możliwości w dalszych etapach rywalizacji.

Także zgranie zawodników wchodzących w skład zespołu ma wpływ na osiągane przez niego wyniki. Im lepiej dogadują się oni ze sobą, tym lepsze odnosi on rezultaty. Jedni ludzie wykazują większą chęć do gry drużynowej, podczas gdy pozostali preferują brawurowe, indywidualne szarże na przeciwnika.

Wszystkie wymienione czynniki pokazują ogromną wagę odpowiedniego wyboru członków zespołu, pozwalającego osiągać najlepsze wyniki. Dobra drużyna powinna składać się z zawodników, którzy uzupełniając się tworzą kolektyw.

Selekcja zawodników wirtualnych

Po utworzeniu zespołu zawodników należy dokonać wyboru postaci, którymi będą oni sterować (takie postacie można nazwać avatarami, zawodnikami wirtualnymi czy też wojownikami), kierując się kilkoma czynnikami.

Niektórzy zawodnicy mogą wykazywać większe lub mniejsze zdolności do gry konkretnymi postaciami. Co więcej, predyspozycje zawodników mogą się czasem krzyżować. Dlatego podczas selekcji należy zadbać, by budowana drużyna osiągnęła jak największą siłę przy jednoczesnym osłabieniu siły drużyny przeciwnej. Kompromis jest trudny do osiągnięcia, ale możliwy.

Każda z dostępnych postaci jest unikatowa, a jej efektywność gry przeciwko innym postaciom może być różna. Ponieważ zespół składa się z pięciu zawodników, istotne jest, żeby wybrane postaci tworzyły drużynę o oczekiwanej charakterystyce – wojownicy powinny uzupełniać się swoimi zdolnościami. Ponieważ wojownicy spełniają podczas rozgrywki różne role i selekcja takich bohaterów, których funkcje dopełniają się, potrafi przeważyć szalę zwycięstwa.

Dodatkowo bohaterowie są wyposażeni w różnego rodzaju przedmioty, które lepiej sprawdzają się dla niektórych z nich, podczas gdy dla innych okazują się prawie bezużyteczne. Biorąc pod uwagę preferencje zawodników oraz możliwości kupna i zastosowania pewnych przedmiotów wybór postaci wcale nie musi być oczywisty.

4.4. Wybór ostatniego bohatera w sposób maksymalizujący szansę na zwycięstwo

Rozważana jest gra w trybie *Captain's Mode*. Sposób prowadzenia rozgrywki, czyli wybieranie bohaterów z listy 113 dostępnych postaci naprzemiennie przez kapitanów rywalizujących drużyn, powoduje, że wiedza o dotychczasowym składzie własnej drużyny jest bardzo istotna.

Interesującym problemem, rozważanym w niniejszej pracy, jest zagadnienie selekcji postaci w ostatniej kolejce kompletowania zespołów. W takiej sytuacji znanych jest już czterech bohaterów wchodzących w skład drużyny i pozostaje dobranie tylko jednego, który w najlepszy sposób uzupełniałby stworzony zespół.

Wiedza na temat składów drużyn i wyników jakie rezultaty one osiągały, pozwoliłaby pomóc w wyborze optymalnego składu drużyny, który miałby największe szanse na zwycięstwo. Posiadając dane dotyczące rozegranych gier można na ich podstawie wnioskować czy i w jaki sposób wybierani bohaterowie mają wpływ na wynik meczu.

4.4.1. Dane statystyczne

Potrzebne do analizy dane dotyczące meczów zostały pobrane ze strony internetowej [6]. W tym celu skorzystano z możliwości odpytywania bazy danych przy pomocy zapytań SQL. Pozyskano tylko niezbędne dane: id meczu oraz id bohatera wraz z informacją o jego zwycięstwie bądź porażce. Zdecydowano się pobrać 50 tys. rekordów, czyli wyniki 5 tys. meczów. Rezultaty kwerendy posortowano według id meczu, natomiast wewnątrz pojedynczego pojedynku bohaterowie automatycznie posortowani są według przynależności do drużyny oraz kolejności selekcji. Treść zapytania SQL przedstawiona jest na listingu 4.2.

Listing 4.2: Zapytanie SQL pozyskujące wyniki meczów

```
SELECT matches.match_id,
     ((player_matches.player_slot < 128) = matches.radiant_win) win,
       player_matches.hero_id
FROM matches
JOIN player_matches USING(match_id)
JOIN heroes ON player_matches.hero_id = heroes.id
ORDER BY matches.match_id
LIMIT 50000
```

Zdecydowano się na pobranie danych w formacie `.csv`. Umożliwiło to ich późniejszą obróbkę w arkuszu kalkulacyjnym. Sprowadzała się ona do pogrupowania bohaterów w drużyny, które reprezentowali w danym meczu. Dodatkowo zmieniono id postaci na ich główne atrybuty. W ten sposób wyróżniono trzy cechy, według których później dokonywano klasyfikacji. Ostatecznie rekord analizowanych danych składał się z:

- głównych atrybutów każdego z bohaterów drużyny (z zachowaniem kolejności selekcji postaci do zespołu),
- binarnej informacji na temat wyniku spotkania.

4.4.2. Uczenie maszynowe

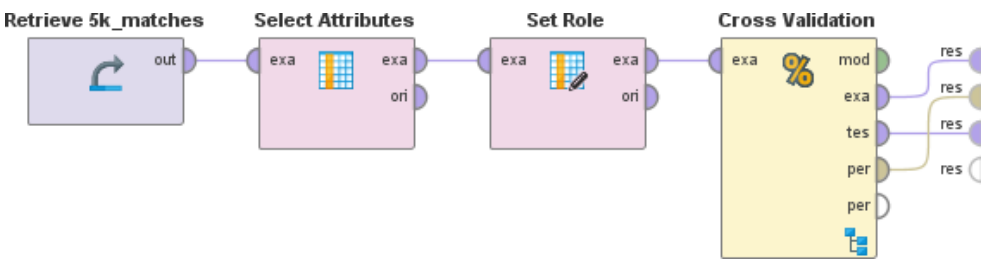
Uczenie maszynowe jest dziedziną wchodzącą w skład nauk zajmujących się sztuczną inteligencją. Z jego pomocą możliwe jest rozwiązywanie problemów przez komputer, bez bezpośredniego zaprogramowania go przez człowieka. Metoda działania polega na dostarczaniu do systemu zewnętrznych danych empirycznych, na podstawie których poznaje on prawa rządzące konkretnym procesem. Posiadając odpowiednią liczbę danych, system powinien być w stanie poznać badane zjawisko na tyle, żeby móc rozwiązać przedstawiony mu problem.

Ucząc system trzeba pamiętać, że zbiór uczący nie powinien być ani zbyt mały, ani zbyt duży, ponieważ w takich przypadkach mogą wystąpić odpowiednio zjawiska niedouczenia lub przeuczenia. Ponadto rozważany proces należy scharakteryzować za pomocą dobrze dobranej liczby atrybutów. Jest to ważna kwestia, ponieważ nieoptymalna liczba cech prowadzi do złego opisu problemu, co skutkuje błędnym poznaniem zjawiska przez maszynę.

RapidMiner Studio

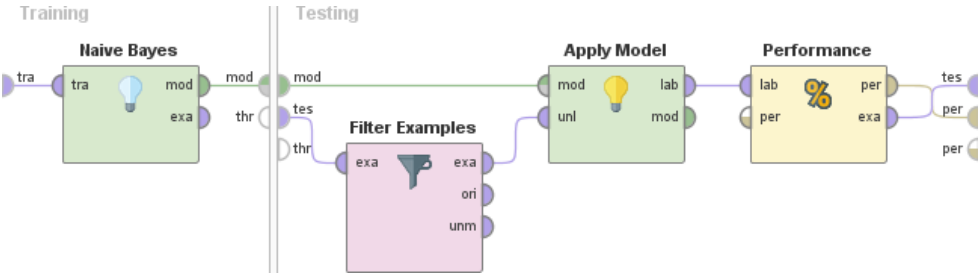
RapidMiner Studio (`https://rapidminer.com/products/studio/`) to program służący do analizy danych. Dostarcza środowisko do uczenia maszynowego. W darmowej wersji jest rozprowadzany na zasadach licencji AGPL i pozwala na przetwarzanie do 10 tys. rekordów.

Procesy w programie opisywane są w sposób graficzny poprzez łączenie bloków funkcjonalnych. Na rysunku 4.1 przedstawiono schemat pozwalający na analizę selekcji zawodników. System na wejściu dostaje dane opisujące 5 tys. meczów. W następnych krokach wybierane są tylko niezbędne do analizy atrybuty, a także określane są ich role. W rozważanym zagadnieniu przedmiotem zainteresowania jest selekcja ostatniego zawodnika, dlatego wybrany jest główny atrybut piątego bohatera. Ostatni etap jest najważniejszy i odpowiada za walidację krzyżową.



Rys. 4.1: Program analizujący dane opisujące mecze

Blok walidacji krzyżowej składa się z zagnieżdżonych w nim bloków. Jego struktura przedstawiona jest na rysunku 4.2. Walidacja krzyżowa to metoda statystyczna polegająca na podziale próby statystycznej na podzbiory, które następnie są analizowane. Typową liczbą podzbiorów jest 10 i tyle też wybrano w programie.



Rys. 4.2: Blok walidacji krzyżowej

Pierwszy etap walidacji krzyżowej polega na uczeniu programu zbiorem uczącym, natomiast w drugim kroku testowana jest wiarygodność otrzymanych wyników zbiorem testowym. W rozważanym przykładzie jako metodę uczenia wybrano naiwny klasyfikator Bayesa. Mimo swojej prostoty daje ona bardzo dobre wyniki [7]. Uczenie zachodzi przy użyciu danych dotyczących wszystkich wyników, natomiast walidacja stworzonego modelu zachodzi tylko dla zwycięskich drużyn.

Ocena zaproponowanego modelu zachodzi w ostatnim bloku. Przykładowy wynik, w postaci macierzy błędów, dla uczenia przy użyciu naiwnego klasyfikatora Bayesa pokazano na rysunku 4.3. Dokładność klasyfikatora wynosi $45,76\% \pm 2,66\%$. Jest to skuteczność z jaką komputer przewiduje wybór ostatniego zawodnika w zwycięskiej drużynie.

accuracy: 45.76% +/- 2.66% (mikro: 45.80%)

	true str	true int	true agi	class precision
pred. str	1040	628	657	44.73%
pred. int	392	774	402	49.36%
pred. agi	345	286	476	43.00%
class recall	58.53%	45.85%	31.01%	

Rys. 4.3: Macierz błędów dla naiwnego klasyfikatora Bayesa

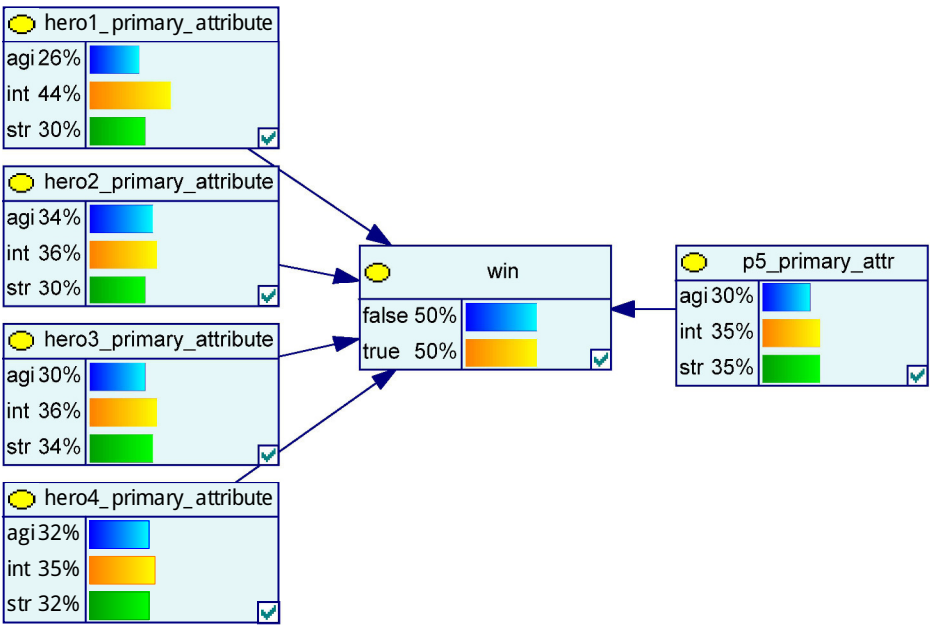
4.4.3. Sieć bayesowska

Sieć bayesowska jest graficznym modelem reprezentującym zmienne losowe i ich warunkowe prawdopodobieństwa jako skierowany graf acykliczny. Węzły sieci to zmienne losowe, natomiast łuki reprezentują zależności między węzłami, czyli bezpośredni wpływ zmiennych losowych na siebie nawzajem. Podstawowym założeniem sieci bayesowskiej jest niezależność zmiennych losowych od innych, nie będących ich przodkami. Zmienne losowe przyjmują wartości dyskretne i każdy węzeł ma stowarzyszoną z nim tablicę prawdopodobieństw warunkowych.

GeNIe

GeNIe (`https://www.bayesfusion.com/genie-modeler`) to graficzny interfejs użytkownika dla silnika wnioskowania SMILE (`https://www.bayesfusion.com/smile-engine`). Umożliwia interaktywne budowanie i badanie sieci bayesowskich. Aby skonstruować sieć bayesowską należy określić zmienne losowe oraz połączenia pomiędzy nimi. Następnie należy wprowadzić dane, na podstawie których określane są prawdopodobieństwa warunkowe i „a priori". Gotową sieć można badać poprzez określanie wartości niektórych zmiennych losowych. W takim przypadku program aktualizuje wiedzę na temat zjawiska i wyznacza prawdopodobieństwa „a posteriori".

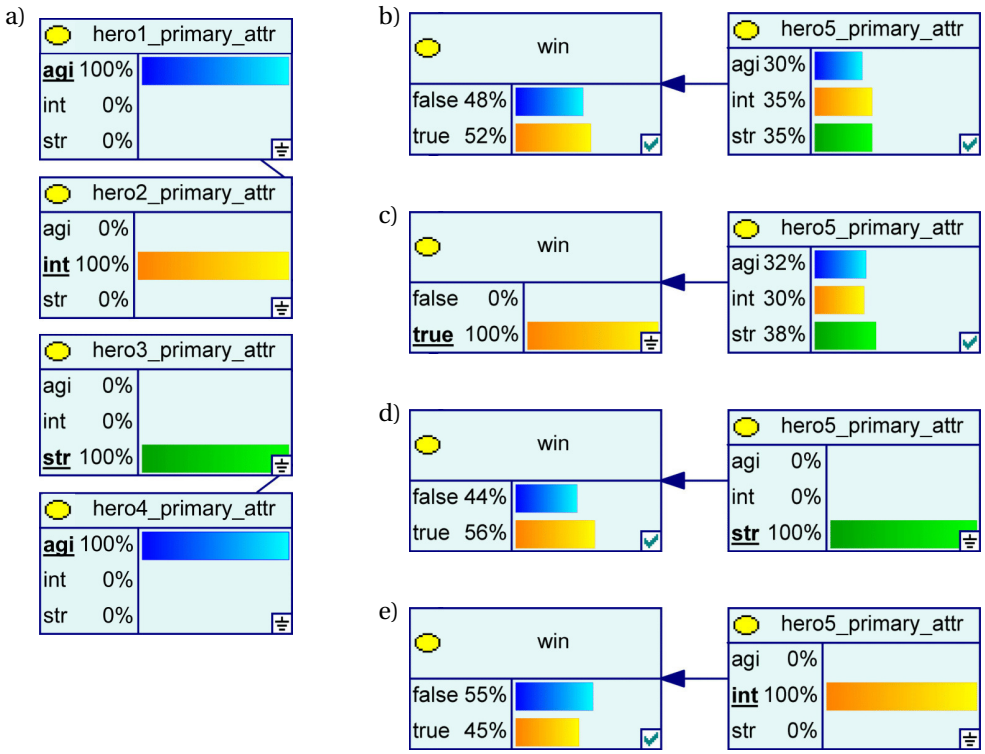
Sieć reprezentująca problem selekcji ostatniego zawodnika zwycięskiej drużyny składa się z sześciu węzłów (rys. 4.4). Pięć z nich odpowiada zmiennym losowym określającym prawdopodobieństwo wyboru w danej turze bohatera o określonym atrybucie głównym (`agi` – zręczność, `int` – inteligencja, `str` – siła), natomiast szósty określa szansę drużyny na zwycięstwo.



Rys. 4.4: Sieć bayesowska dla problemu selekcji ostatniego zawodnika

Wybór zawodników do drużyny może przebiegać w kolejności: 1. zręczność, 2. inteligencja, 3. siła, 4. zręczność (rys. 4.5a). Prawdopodobieństwo wygranej takiej drużyny wynosi 52% (rys. 4.5b). Gdy na cel weźmie się tylko osiągnięcie zwycięstwa (rys. 4.5c), to widać, że wygrywające składy najczęściej posiadały piątego bohatera o głównym atrybucie siły. Rzeczywiście, przy dokonaniu optymalnej selekcji prawdopodobieństwo zwycięstwa wynosi 56% (rys. 4.5d). Natomiast nie postępując według zaleceń sieci i wybierając bohatera np. o głównym atrybucie

inteligencji skompletuje się drużynę, która będzie przegrywać w 55% przypadków (rys. 4.5e), czyli aż o 11 punktów procentowych częściej.



Rys. 4.5: Przykładowa analiza selekcji ostatniego bohatera: a) atrybuty główne czterech bohaterów wybranych przed ostatnią turą; b) prawdopodobieństwo wygranej/porażki drużyny złożonej z dotychczas wybranych czterech postaci; c) określenie przypadku, w którym rozważany jest wybór wojownika tylko do zwycięskiego zespołu i prawdopodobieństwo obecności postaci o danym atrybucie głównym w takiej drużynie; d) prawdopodobieństwo wygranej/porażki przy optymalnym wyborze ostatniego bohatera; e) prawdopodobieństwo wygranej/porażki przy nieoptymalnym wyborze ostatniego bohatera.

4.5. Podsumowanie

Selekcja zawodników jest złożonym problemem, którego analiza może pomóc w osiąganiu sukcesów. W pracy omówiono zagadnienie wyboru ostatniego bohatera do zwycięskiej drużyny przy znajomości pozostałych członków zespołu. Przeprowadzone badania pokazały, że mając do dyspozycji dane dotyczące wystarczająco wielu rozgrywek, możliwe jest wyciągnięcie z nich użytecznych informacji, a następnie wykorzystanie nabytej wiedzy do przeprowadzenia optymalnej selekcji zawodników.

Literatura

[1] J. Katzenbach, D. Smith. *The wisdom of teams: Creating the high-performance organization*. Harvard Business Press, 1993.

[2] B. Kirkman, B. Rosen, C. Gibson, P. Tesluk, S. McPherson. Five challenges to virtual team success: Lessons from Sabre, Inc. *The Academy of Management Executive*, 16(3):67–79, 2002.

[3] G. D'Souza, S. Colarelli. Team member selection decisions for virtual versus face-to-face teams. *Computers in Human Behavior*, 26(4):630–635, 2010.

[4] S. Sauer, P. Arce. Team member selection: A functional-based approach. *Proceedings of the 2004 American Society for Engineering Education Annual Conference & Exposition, American Society for Engineering Education, Salt Lake City, UT*, wolumen 3513, strony 1–6, 2004.

[5] *Dota 2 – Steam Charts*. `www.steamcharts.com/app/570`, [dostęp dnia 4.04.2017].

[6] *OpenDota – Dota 2 Statistics*. `www.opendota.com/`, [dostęp dnia 4.04.2017].

[7] H. Zhang. The Optimality of Naive Bayes. *AAAI Press*, 2004.

ROZDZIAŁ 5

TECHNICZNE WSPARCIE OSÓB NIEWIDOMYCH I NIEDOWIDZĄCYCH W ODCZYTYWANIU WZORÓW MATEMATYCZNYCH

W. Lipieta

W rozdziale przybliżono zagadnienia związane z możliwościami technicznego wsparcia osób niewidomych i słabowidzących w odczytywaniu wzorów matematycznych zamieszczanych w dokumentach elektronicznych oraz na stronach internetowych. Zwrócono uwagę na istniejące technologie zwiększające dostępność takich dokumentów oraz wykorzystywane tam standardy.

5.1. Wstęp

Wzrok jest jednym z najważniejszych zmysłów człowieka, odgrywającym ogromną rolę w wielu obszarach jego funkcjonowania. Umożliwia poznawanie rzeczywistości (np. uczenie się), orientację w przestrzeni oraz dostęp do dóbr kultury i nauki. Szacuje się, że zmysł wzroku dostarcza ok. 75%-80% informacji o świecie. Brak tego zmysłu lub jego uszkodzenie może prowadzić do wykluczenia społecznego. By do tego nie dopuścić opracowano metody wspierające percepcję świata za pomocą innych zmysłów, w tym metody wspierające czytanie i pisanie tekstów zawierających wzory matematyczne.

Mówiąc o osobach z niepełnosprawnością wzroku zwykle rozróżnia się [1]:

- osoby niewidome – osoby, które na skutek znacznego stopnia uszkodzenia narządu wzroku muszą posługiwać się technikami bezwzrokowymi (korzystając z dotyku i pomocy technicznych, które kompensują uszkodzenie wzroku);
- osoby słabowidzące – osoby, które pomimo znacznego uszkodzenia narządu wzroku są w stanie się nim posługiwać, napotykając jednak przy tym określone trudności i ograniczenia (grupa ta jest silnie zróżnicowana pod kątem posiadanych możliwości wzrokowych i umiejętności ich wykorzystywania).

Według klasyfikacji Światowej Organizacji Zdrowia [2] osoby niewidome to: osoby całkowicie niewidome, z poczuciem światła, ze szczątkowym widzeniem, ze znacznym ograniczeniem pola widzenia. Niezależnie jednak od stopnia niepełnosprawności dla osób niewidomych czy słabowidzących odczytywanie wzorów matematycznych bez żadnego wsparcia jest niemożliwe bądź bardzo utrudnione.

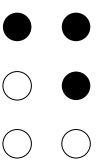
Opracowanie metod wspomagających osoby z niewidomych lub niedowidzących w odczytywaniu i zapisywaniu wzorów matematycznych jest nie lada wyzwaniem. O jego trudności świadczy fakt, iż stosowane we wzorach symbole nie mieszczą się w żadnym znanym alfabecie, a ich znaczenie często wynika z kontekstu położenia względem innych symboli. Celem rozdziału jest przybliżenie sposobów zapisu wzorów zamieszczanych na stronach internetowych oraz zaprezentowanie technologii, które wspomagają ich odczytywanie.

5.2. Zwiększanie dostępności dokumentów

Programy komputerowe mogą wspierać osoby niepełnosprawne w odczytywaniu dokumentów na wiele sposobów. Na przykład umożliwiają zwiększenie rozmiaru wyświetlanego tekstu, poprawę kontrastu, zmianę palety barw itp. Funkcje te są wystarczające, by poprawił się dostęp do dokumentów dla osób o mniejszych wadach wzroku. Jednak wparcie osób niewidomych wymaga zastosowania bardziej zaawansowanych technik.

5.2.1. Alfabet Braille'a

Aby umożliwić osobom niewidomym zapisywanie i odczytywanie tekstów używany jest alfabet opracowany przez Louisa Braille'a. Opierając się na wojskowym systemie do odczytywania rozkazów bez użycia światła zaproponował system zapisu znaków za pomocą kombinacji sześciu wypukłych punktów. Dzięki temu, teksty można odczytywać za pomocą dotyku. Na rysunku 5.1 pokazano przykład zapisu wybranej litery w tym alfabecie.



Rys. 5.1: Litera 'D' zapisana przy użyciu Braille'a

Alfabet Braille'a nie jest uniwersalnym alfabetem. W różnych krajach sposób zapisu znaków może być różny [3]. Generuje to trudności w odczytywaniu tekstów napisanych w obcym języku, co zwykle zdarza się przy lekturze prac naukowych. Co więcej, w podstawowej formie alfabet ten nie nadaje się do zapisu wzorów matematycznych. Z tego powodu stworzono wiele standardów do reprezentacji matematycznych i naukowych wyrażeń za pomocą Braille'a [4]. Przykła-

dowo w Polsce używana jest notacja Marburg (nazwana tak od miejsca jej wprowadzenia). W USA oraz innych częściach świata dużą popularność zdobył standard *Nemeth Code for Braille Mathematics*. To właśnie na odmianie Nemeth pracuje większość dostępnych narzędzi do przetwarzania wzorów matematycznych na Braille'a.

Oczywiście oprócz kodowania znaków problemem jest również fizyczna postać generowanego tekstu. Samo wyświetlanie na ekranie czcionek obrazujących kody Braille'a tutaj nie wystarcza. Konieczne jest ich przedstawienie w postaci, którą można odczytywać za pomocą dotyku. W tym celu wykorzystuje się tzw. drukarki brajlowskie (ang. *Braille embosser*), operujące na specjalnym perforowanym papierze. Używa się też terminali brajlowskich, jak np. BraillePen 12 Touch przedstawiony na rysunku 5.2. Takie terminale są urządzeniami posiadającymi linijkę kilkunastu pól, na których powstające wzgórki i wgłębienia odpowiadają znakom Braille'a reprezentującym odczytywany tekst.

Rys. 5.2: Terminal BraillePen 12 Touch z klawiaturą [5]

5.2.2. Czytniki ekranu

Do najbardziej przydatnych programów ułatwiających osobom niewidomym korzystanie z komputera zalicza się czytniki ekranu. Odczytują one na głos tekst znajdujący się na elementach wskazywanych przez użytkownika, np. tekst wyświetlany na stronach internetowych lub w dokumentach; nazwy programów. Programy te generują również komunikaty pomagające zorientować się o aktualnym stanie logiki interfejsu użytkownika. Przykładami takich programów są: `ChromeVox` [6], rozwijany przez Google i dostępny w przeglądarce Chrome oraz systemie operacyjnym ChromeOS; `NVDA` (ang. *NonVisual Desktop Access*) [7], rozwijany przez organizację NV Access typu non-for-profit, działający w systemie Windows. Oba wymienione czytniki są darmowe i wspierają język polski oraz angielski. Ponadto `NVDA` jest oprogramowaniem typu open-source i każdy może zaproponować własne rozszerzenia i poprawki.

5.3. Problem odczytywania wzorów matematycznych

Podczas pisania i odczytywania wzorów matematycznych liczą się nie tylko znaki, z jakich się one składają, ale również ich względne położenie. Lokalizacja znaków w przestrzeni jest niezbędna do zrozumienia znaczenia danego wzoru i jego poprawnego odczytania. Przykładowo, linia pozioma występująca we wzorze 5.1 oznacza dzielenie, wiadomo więc, że część wzoru powyżej linii jest licznikiem, a poniżej – mianownikiem. Do oznaczenia dzielenia we wzorach matematycznych można jednak zamiast linii zastosować ukośnik. W ten sposób powstają wzory o innym wyglądzie graficznym, ale tym samym znaczeniu (wzór 5.2).

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{5.1}$$

$$x = (-b \pm \mathrm{sqrt}(b\hat{}2 - 4ac))/(2a) \tag{5.2}$$

5.4. Sposoby zapisu wzorów na stronach internetowych

Publikowanie wzorów matematycznych na stronach internetowych było swego czasu bardzo trudnym zadaniem. Aby je uprościć starano się zapisywać wszelkie wzory w postaci liniowej (jak we wzorze 5.2) lub graficznej (w formie obrazków).

Pierwszy z wymienionych sposobów pozwalał wyeliminować potrzebę przestrzennego rozmieszczania poszczególnych elementów. Co więcej, choć odczyt wzorów w liniowej postaci może stanowić wyzwanie dla przeciętnego użytkownika, to jednak właśnie taka postać wychodzi naprzeciw wymaganiom użytkowników z wadami wzroku. A dokładniej – liniowo zapisane wzory można dowolnie powiększać oraz przetwarzać na Nemeth Braille, a następnie przesyłać do specjalnych drukarek lub terminali brajlowskich.

Drugi ze sposobów, tj. publikowanie wzorów w formie obrazków, choć daje się zautomatyzować (np. przez kompilację w systemie LaTeX), nie jest jednak zalecany. Bitmapy z grafikami przedstawiającymi wzorami są zwykle generowane w jak najmniejszej rozdzielczości (aby ograniczyć ich rozmiar). Takich obrazków nie dają się dobrze ani skalować, ani konwertować do innej postaci. Co prawa do konwersji obrazków do znakowej postaci można zatrudnić oprogramowania OCR (ang. *Optical Character Recognition*), to jednak w przypadków złożonych matematycznych niezwykle trudno jest osiągnąć zadowalający efekt. Na dodatek w obszarze tym brakuje darmowych rozwiązań.

Komercyjne czytniki ekranu, takie jak JAWS (ang. *Job Access With Speech*) [8] posiadają wbudowane funkcjonalności OCR, jednak nie wspierają odczytywania wzorów matematycznych zapisanych jako obrazy. Są jednak dostępne płatne biblioteki OCR, przeznaczone do odczytywania wzorów z obrazów z zachowaniem ich znaczenia. Przykładem może być Mathpix [9], który pozwala na konwersję wzoru do ustandaryzowanego formatu tekstowego, np. LaTeX. Jest więc możliwe napisanie na ich podstawie programu odczytującego wzory, muszą one jednak być zapisane z odpowiednio wysoką rozdzielczością.

Aby usystematyzować sposób opisu wzorów i symboli matematycznych na stronach internetowych oraz rozwiązać powyższe problemy, W3C opracowało język MathML [10]. Bazuje on na XML i umożliwia jednoznaczne wyrażenie zarówno struktury wzoru, jak i jego znaczenia. Elementy stron internetowych zredagowane w składni MathML mogą być renderowana przez przeglądarkę do formy wizualnej, odpowiedniej dla każdego użytkownika.

W MathML wyróżniono: *Presentation MathML* – część języka służąca do opisu sposobu wyświetlania wzorów na stronie (dającą użytkownikowi pewną dowolność) oraz *Content MathML* – część służącą do określenia semantyki wzorów (do precyzowania matematycznych zależności pomiędzy elementami).

Rozbudowana składnia MathML, opracowana z myślą o łatwym przetwarzaniu przez komputer, może sprawiać kłopoty zwykłemu użytkownikowi redagującemu wzory. Napisanie skomplikowanego wzoru ręcznie może być bardzo trudne, a kod potrzebny do jego opisu – niezwykle obszerny. Również odczytywanie czystego kodu MathML nie jest intuicyjne dla człowieka. Widoczne jest to na listingach 5.1 oraz 5.2, gdzie porównano zapis wzoru 5.1 za pomocą składni MathML oraz LaTeX.

Listing 5.1: Wzór 5.1 zapisany w składni MathML

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <mrow>
    <mi>x</mi> <mo>=</mo>
    <mfrac>
      <mrow>
        <mrow><mo>-</mo><mi>b</mi></mrow>
        <mo>&PlusMinus;</mo>
        <msqrt>
          <mrow>
            <msup><mi>b</mi><mn>2</mn></msup>
            <mo>-</mo>
            <mrow>
              <mn>4</mn><mo>&InvisibleTimes;</mo>
              <mi>a</mi><mo>&InvisibleTimes;</mo>
              <mi>c</mi>
            </mrow>
          </mrow>
        </msqrt>
      </mrow>
      <mrow>
        <mn>2</mn><mo>&InvisibleTimes;</mo>
        <mi>a</mi>
      </mrow>
    </mfrac>
  </mrow>
</math>
```

Listing 5.2: Wzór 5.1 zapisany w składni LaTeX

```
x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}
```

Na bazie MathML można budować kolejne funkcje oferowane użytkownikowi. I tak np. przeglądarka Internet Explorer daje możliwość zainstalowania dodatku MathPlayer [11] obsługującego MathML, z wbudowanym syntezatorem mowy (do odczytywania wzorów) oraz konwerterem (do składni Nemeth Braille).

5.5. MathJax

Ponieważ natywne wsparcie dla MathML nie występuje we wszystkich przeglądarkach, do obsługi tego języka w ich przypadku potrzeba zewnętrznych systemów i bibliotek.

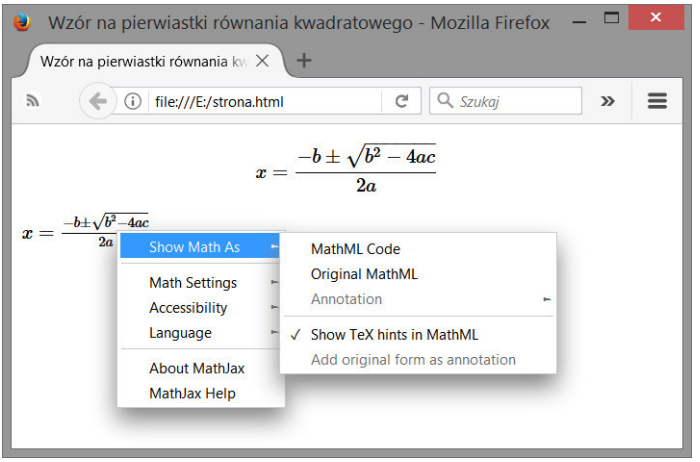
MathJax [12] to open source'owy silnik to wyświetlania wzorów matematycznych napisany przy użyciu języka JavaScript. Pozwala na renderowanie wzorów w oknach wszystkich nowoczesnych przeglądarkach internetowych, również mobilnych. Dzięki temu przeglądarki nie muszą już być wyposażone w zdolność samodzielnego renderowania wzorów zapisanych przy użyciu dedykowanych języków, jak MathML. Ponadto, MathJax umożliwia bezpośrednie zamieszczanie na stronie internetowej wzorów napisanych przy użyciu składni LaTeX, która jest bardziej popularna i łatwiejsza w użyciu. Warto dodać, że renderowane mogą być nie tylko wzory matematyczne, ale również chemiczne i fizyczne.

MathJax jest również narzędziem zwiększającym dostępność wzorów matematycznych. Dzięki jego użyciu użytkownicy mają dostęp do dodatkowego menu kontekstowego, umożliwiającego powiększanie wzorów, renderowanie ich na różne formaty (np. SVG czy HTML-CSS) oraz podglądanie kodu MathML. MathJax współpracuje również z czytnikami ekranu, pozwalając na odczytywanie zaznaczonego wzoru za pomocą modułu *Speech-Rule-Engine*. Dodatkowo możliwe jest włączenie trybu eksploratora wzorów, który pozwala przechodzić pomiędzy elementami składowymi wzoru z zachowaniem jego hierarchii i znaczenia. Dzięki temu użytkownik może odczytywać za pomocą czytnika jedynie wybrany fragment. Co więcej, jak pokazano na listingu 5.3, możliwe jest wyświetlanie tekstu opisującego słownie zawartość danego wzoru. Po włączeniu tej opcji, osoba niewidoma może przetworzyć wzór i przeczytać jak normalny tekst, do którego nie jest potrzebna znajomość żadnej z notacji matematycznych. Niestety, w aktualnej wersji wspierane jest jedynie konwertowanie wzorów na język angielski.

Listing 5.3: Wzór 5.1 przetworzony na tekst za pomocą MathJax

```
'x Equals StartFraction negative b PlusMinus
StartRoot b squared minus 4 a c EndRoot Over 2 a EndFraction'
```

Do zaprezentowania możliwości biblioteki MathJax stworzono szablon prostej strony internetowej (listing 5.4). Na stronie wyświetlany jest wzór 5.1 zapisany zarówno w składni LaTeX, jak i MathML. Na stronie pojawiają się dodatki zwiększające jej dostępność. Aby móc korzystać z nich korzystać użytkownik musi mieć włączoną obsługę JavaScript w przeglądarce oraz aktywować wybrane funkcje w menu kontekstowym MathJax'a.



Rys. 5.3: Elementy pojawiające się na stronie internetowej ze wzorem matematycznym

Podczas używania czytnika ekranu aby uzyskać dźwiękowy efekt wystarczy umieścić w odpowiednim miejscu kursor oraz nacisnąć kombinację klawiszy Shift + Spacja. Po tej operacji cały wzór zostanie odczytany. Do przemieszczania się po elementach składowych wzoru można użyć strzałek, co umożliwia odczytywanie wybranych fragmentów.

Listing 5.4: Przykład publikowania wzoru matematycznego na stronie internetowej

```
<!DOCTYPE html>
<html>
<head>
 <title>Wzór na pierwiastki równania kwadratowego</title>
 <!-- Dołączenie biblioteki MathJax -->
 <script type="text/javascript"
 src="https://cdnjs.cloudflare.com/ajax/libs/mathjax/
2.7.1/MathJax.js?config=TeX-AMS-MML_HTMLorMML">
 </script>
</head>
<body>
<p> <!-- składnia LaTeX -->
$$x = {-b \pm \sqrt{b^2-4ac} \over 2a}$$
</p>
<p> <!-- składnia MathML -->
 <math xmlns="http://www.w3.org/1998/Math/MathML">
  <mrow>
     .....
  </mrow>
 </math>
</p>
</body>
</html>
```

5.6. Podsumowanie

W rozdziale przedstawiono najważniejsze technologie oraz narzędzia wspomagające odczytywanie wzorów matematycznych przez osoby niedowidzące. Skupiono się na sposobach zamieszczania wzorów na stronach internetowych, tak aby zapewnić niepełnosprawnym użytkownikom należyty poziom dostępności. Jak widać, dostępny jest kompleksowy zestaw środków ułatwiających pracę ze wzorami, jednak część z nich wymaga odpowiednich działań ze strony osób tworzących strony internetowe. Narzędzia te pozwalają zarówno na odczytywanie wzorów na głos jak i zapis za pomocą alfabetu Braille'a, bez utraty ich pełnego znaczenia.

Literatura

[1] Książek, M. Niepełnosprawność wzrokowa i wzrokowosłuchowa, jako przesłanki dyskryminacji. http://www.tea.org.pl/userfiles/file/Seminaria/niepelnosprawnosc_wzrokowa_wzrokowo_sluchowa_mksiazek.pdf, [dostęp dnia 11.06.2017].

[2] Światowa Organizacja Zdrowia. *Międzynarodowa Klasyfikacja Funkcjonowania, Niepełnosprawności i Zdrowia (ICF)*. Centrum Systemów Informacyjnych Ochrony Zdrowia, 2001.

[3] UNESCO. *World Braille Usage*. National Library Service for the Blind and Physically Handicapped, Library of Congress, Washington, D.C., USA, 1990.

[4] D. Archambault. Braille Mathematical Notations. http://chezdom.net/mathematicalbraillecodes/#ref1, [dostęp dnia 11.06.2017].

[5] Harpo Sp. z o.o. BraillePen 12 Touch. http://www.harpo.com.pl/sklep/sprzet-dla-niewidomych-i-slabowidzacych/notatniki-i-terminale-brajlowskie/braillepen12-touch-klawiatura/, [dostęp dnia 11.06.2017].

[6] Google. ChromeVox. http://www.chromevox.com/, [dostęp dnia 11.06.2017].

[7] NV Access. NonVisual Desktop Access. https://www.nvda.pl/, [dostęp dnia 11.06.2017].

[8] Freedom Scientific, Inc. JAWS. http://www.freedomscientific.com/Products/Blindness/JAWS, [dostęp dnia 11.06.2017].

[9] Mathpix. Mathpix — strona główna. http://mathpix.com/, [dostęp dnia 11.06.2017].

[10] W3C. MathML. https://www.w3.org/Math/, [dostęp dnia 11.06.2017].

[11] Design Science. MathPlayer. https://www.dessci.com/en/products/mathplayer/, [dostęp dnia 11.06.2017].

[12] MathJax Consortium. MathJax. https://www.mathjax.org/, [dostęp dnia 11.06.2017].

Od redaktora i wydawcy

W niniejszej kolekcji zamieszczono prace autorstwa studentów II roku studiów magisterskich Wydziału Elektroniki Politechniki Wrocławskiej, kierunku Automatyka i robotyka, specjalności Robotyka. Zostały one wykonane w semestrze letnim roku akademickiego 2016/2017 podczas realizacji prowadzonego przeze mnie kursu *Komputerowe przetwarzanie wiedzy*. Opisano w nich zagadnienia związane z szeroko rozumianym wsparciem, jakiego człowiekowi mogą udzielić komputery podczas rozwiązywania różnorodnych problemów naukowych i inżynierskich.

Podobnie do wcześniejszych wydań z serii „Komputerowe przetwarzania wiedzy" rozrzut tematyczny prezentowanych prac jest dość spory. Prace te dotyczą problemów:

- identyfikacji biometrycznej osób (identyfikacja mówcy na podstawie charakterystyki jego głosu);
- wykorzystania metadanych w semantycznym Internecie (zastosowanie arkuszy styli do ekstrakcji metadanych ze stron internetowych);
- nawigacji kwadrokopterem z wykorzystaniem dalmierzy laserowych (projekt platformy, kalibracja czujników, fuzja sygnałów, pętla sterowania);
- doboru składu drużyn biorących udział w rozgrywkach DOTA2 (wskazanie optymalnego składu drużyny do mającego się odbyć meczy na bazie zapisanych historii poprzednich rozgrywek);
- zwiększenia dostępności stron internetowych zawierających wzory matematyczne (narzędzia wpierające osoby niewidome i niedowidzące w odczytywaniu wzorów matematycznych).

Mam nadzieję, że lektura tych opracowań okaże się interesująca dla czytelnika.

Tomasz Kubik
Wrocław, październik 2017

ISBN 978-83-930823-9-1