Raport w ramach projektu realizowanego przez: Snowdog sp. z o.o. pn. „Opracowanie innowacyjnej technologii oraz realizacja prototypu systemu analizującego obraz utrwalony techniką cyfrową przy pomocy smartfonu”

w ramach
Wielkopolskiego Regionalnego Programu Operacyjnego na lata 2014-2020,
Oś Priorytetowa 1. Innowacyjna i konkurencyjna gospodarka,
Działanie 1.2. „Wzmocnienie potencjału innowacyjnego przedsiębiorstw Wielkopolski”.

Nr zadania	3
Poziom TRL	TRL in: 6 TRL out: 9
Numer projektu	RPWP.01.02.00-30-0175/17
Autorzy raportu	Radosław Zaworski, Bartłomiej Kwiatkowski
Czas pracy i zadania wykonywane przez pracowników na rzecz projektu	3.1. Rozwój aplikacji mobilnej (lipiec – listopad 2019) 3.2. Prace nad dalszą optymalizacją i strojeniem modułu analizy obrazów (lipiec – październik 2019) 3.3. Prace nad modułem rekomendacyjnym (lipiec – listopad 2019) 3.4. Przygotowanie procedur i systemu aktualizacji bazy wiedzy (lipiec – listopad 2019) 3.5. Przygotowanie dokumentacji technicznej (lipiec – listopad 2019)
Kamienie milowe:	Działający i przetestowany moduł rekomendujący Działający kompletny system gotowy do pracy w rzeczywistych warunkach. Wskaźnik rezultatu: Średnia trafność rozpoznawania: przynajmniej 86% top 2 Wskaźnik rezultatu: Średni czas rozpoznawania obiektu: poniżej 1,5 sekundy.
Zgodność z Planem w zakresie prac B+R (TAK/NIE)	TAK

1 Przedmiot badania

1.1 Moduł analizy obrazu

Podczas etapu 3 prace skupiły się między innymi na dostrojeniu modułu analizy obrazu do wykorzystania w module rekomendacyjnym.
Aby moduł rekomendacyjny mógł działać szybko, wynik działania modułu analizy obrazu musi być zoptymalizowany pod kątem porównywania go do innych, obliczonych wcześniej, wektorów cech produktów. Dodatkowo metryka służąca do porównywania powinna pozwolić na utworzenie struktury danych umożliwiającej wydajne przeszukiwanie przestrzeni produktów.
Kolejnym wyzwaniem jest rodzaj danych wejściowych. Przygotowanie zbioru danych treningowych o wystarczającej liczbie próbek używając znaczącej metryki określającej podobieństwo produktów może okazać się trudne lub wręcze niemożliwe. Realnym rozwiązaniem jest przygotowanie danych wejściowych w postaci tzw. par pozytywnych i negatywnych. W przypadku tego zadania oznacza to, że zdjęcia które przedstawiają ten sam produkt łączone są w pary pozytywne. W celu zbalansowania procesu nauki sieci potrzebne są również pary negatywne, które mogą zostać utworzone na podstawie losowych zdjęć, które przedstawiają dwa różne produkty.
Te wymogi ograniczają zbiór technik które można zastosować do takiego zadania. Po pierwsze techniki nie tworzące funkcji odwzorowującej wejście do nowej przestrzeni, takie jak *Locally Linear Embedding*, *Laplacian Eigenmap*, *Hessian LLE*. Dwie ostatnie metody rozwiązują problem sensownej metryki na wejściu, wystarczającym jest lista najbliższych sąsiadów dla danej próbki, jednak nadal nie umożliwiają porównywania nowych, nieznaych danych bez uprzedniego treningu z ich wykorzystaniem.
Kolejnym ograniczeniem wielu istniejących technik jest tendencja do skupiania w grupach danych w przestrzeni wyjściowej, czasami prowadząc do wyników które można uznać za zdegenerowane.
Wydaje się, że metodą, która może sprostać powyższym wymaganiom jest *Dimensionality Reduction by Learning an Invariant Mapping* (redukcja wymiarowości przez naukę niezmiennego odwzorowania). Redukcja wymiarowości obejmuje odwzorowanie wielowymiarowej przestrzeni wejściowej do niskowymiarowej przestrzeni wyjściowej w taki sposób by podobne do siebie próbki znajdowały się blisko siebie w przestrzeni wyjściowej. Autorzy wymieniają następujące zalety tej metody:

- Wymaga jedynie relacji z sąsiadami pomiędzy próbkami danych treningowych.
- Może nauczyć funkcji które są niezmienne dla skomplikowanych nieliniowych przekształceń danych wejściowych takich jak zmiany oświetlenia czy zniekształcenia geometryczne.
- Nauczona funkcja może być użyta do odwzorowania nowych próbek nie widzianych podczas treningu, bez wcześniejszej wiedzy o nich.
- Odwzorowanie jest w pewnym sensie "gładkie" i spójne w przestrzeni wyjściowej.

Funkcja kosztu nazwana *contrastive loss* jest wykorzystana do nauki parametrów W funkcji G_W w taki sposób, że pary pozytywnych próbek są przyciągane do siebie, a negatywne odpychane.
Ta metoda wykorzystuje model oparty na energii, który używając relacji z sąsiednimi próbkami do wyuczenia funkcji odwzorowującej. Dla rodziny funkcji G sparametryzowanych za pomocą W, celem jest znaleźć taką wartość W która odwzorowuje wysokowymiarową przestrzeń wejściową do przestrzeni wyjściowej takiej że odległość euklidesowa między punktami $D_W(\vec{X_1}, \vec{X_2}) = \|G_W(\vec{X_1}) - G_W(\vec{X_2})\|_2$ aproksymuje "semantyczne podobieństwo" danych w przestrzeni wejściowej.
To wszystko oznacza, że wykorzystując powyższą motodę możemy uzyskać model mapujący obrazy w pewien sposób przekształcone (zdjęcie w nieoptymalnych warunkach oświetleniowych, zrobione pod różnymi kątami) do przestrzeni wyjściowej w której powinny znaleźć się w sąsiedztwie podobnych obrazów – zdjęć produktów w sklepie. Dzięki temu, że wykorzystywana jest odległość euklidesowa, zdjęcia produktów w sklepie mogą być wcześniej odwzorowane w przestrzeni wyjściowej i przechowywane w strukturze danych takiej jak np. drzewa k-wymiarowe, którą będzie można potem wydajnie przeszukiwać w module rekomendacyjnym.

1.2 Rekomendacje behawioralne

Rekomendacje behawioralne maja za zadanie spersonalizowanie polecanych użytkownikowi przedmiotów na stronie sklepu tak by trafiły w jego gusta i upodobania. Klasyczne podejście do spersonalizowanych rekomendacji opiera się na jawnych [25] informacjach które dostarcza użytkownik, jak oceny piosenek w serwisie muzycznym czy historia obejrzanych filmów na platformie VOD. Tak

przedstawiony problem jest klasycznie rozwiązywany za pomocą collaborative filtering[16] czy też content-based filtering[1]. Collaborative filtering opiera się na założeniu że użytkownicy którzy zgadzali się w przeszłości będą mieć podobne zdanie w przyszłości, na podstawie tego założenia użytkownicy są porównywani a rekomendacje generowane są na bazie najbliższych, najpodobniejszych, użytkownikowi sąsiadów. Występuje tu problem tak zwanego 'zimnego startu'[8], który polega na zbyt malej ilości informacji które dostarczył użytkownik by algorytm mógł skutecznie znaleźć najbardziej mu podobnych sąsiadów. Innym problemem jest skalowalność takiego rozwiązania. Collaborative filtering tworzy macierz o wymiarze UxP, gdzie U to liczba użytkowników a P to liczba przedmiotów. Zatem im więcej przedmiotów oferuje dany sklep i im więcej ma użytkowników tym więcej mocy potrzeba na wyliczenie rekomendacji. Z kolei content-based filtering bazuje na danych o produktach, stosowany w sytuacjach gdy wiadome są cechy produktów i można wyliczyć między nimi podobieństwa. Profil użytkownika jest następnie budowany w oparciu o przedmioty jakie użytkownik ocenił, polubił czy tez odwiedził.
Niestety w rozważanym przypadku system nie posiada jawnej wiedzy o użytkowniku, nie ma również dostępu do historii jego działań. Jedynym źródłem wiedzy o preferencjach użytkownika jest jego aktywna sesja w sklepie. Sesją nazywamy zbiór wszystkich aktywności jakie użytkownik dokonuje w czasie odwiedzin strony sklepu. Sesja jest również przedstawiona jako sekwencja zdarzeń, uporządkowana względem czasu. Problem rekomendacji behawioralnej w oparciu o sesję użytkownika można sprowadzić do problemu predykcji następnego kroku użytkownika, co sprowadza się do problemu klasyfikacji sekwencji. Do rozwiązania tego problemu użyte zostały algorytmy uczenia z nadzorem[18]. Do oceny jakości działania systemu w czasie fazy eksperymentów użyta została, powszechna w problemach klasyfikacji metryka, celność. Celność wyliczana na osobnym zbiorze testowym stanowi procent udanych predykcji. Jednak pożądaną cechą systemu rekomendacji jest jest zróżnicowanie wyników, aby nie serwował użytkownikowi tych samych, najpopularniejszych przedmiotów, lecz zapewnił użytkownikowi możliwie jak najróżnorodniejsze ale wciąż adekwatne rekomendacje. Równomierna sprzedaż całego katalogu produktów jest istotna dla sklepów internetowych ze względu na ograniczoną powierzchnię magazynową. Zalegające produkty ograniczają możliwość rotacji i wprowadzania do sprzedaży nowych produktów. Nie bez znaczenia jest czas potrzebny na dokonanie przez model predykcji i serwowanie rekomendacji użytkownikowi, najlepiej by odbywało się to w czasie rzeczywistym podczas przeładowywania strony, tak by użytkownik nie musiał czekać.

1.2.1 Dane

Zestaw danych użyty podczas eksperymentów składa się z dwóch części: sesji użytkowników, zawierającej aktywności w sklepie takie jak wejście na stronę produktu, klikniecie w galerie, dodanie do koszyka; oraz szczegółowych informacji o produktach jak nazwa, opis, nazwa producenta czy kolor. W najprostszej, domyślnie używanej podczas eksperymentów, formie sesje użytkowników zostały sprowadzone do sekwencji odwiedzonych kolejno produktów, wszelkie sesje z liczbą zdarzeń równą jeden zostały odrzucone. System potrzebuje choćby dwóch produktów by przeprowadzić trening, jeden który użytkownik właśnie odwiedza i jeden, ten który odwiedzi następny, jako cel systemu.

1.2.2 Reprezentacja danych

Odpowiednia reprezentacja danych wejściowych wpływa na sprawność działania algorytmu uczenia maszynowego, zarówno jakość osiąganych wyników jak i szybkość działania całego systemu. Od skutecznej reprezentacji danych oczekujemy by była możliwie kompaktowa jednak zachowując wystarczająco dużo informacji o cechach produktu do procesu nauki. Reprezentację danych należy tak zdefiniować, by z każdym nowym by z każdym nowym produktem w sklepie nie przeliczać całego algorytmu uczącego od początku tylko douczyć na nowych danych. Wymaga to więc by dane przedstawione były w jednej, spójnej przestrzeni reprezentacji, a funkcja je tworząca na tyle uniwersalna by mogła bez problemu poradzić sobie z nowymi przedmiotami. Reprezentacje można podzielić na dwie kategorie, jedna to takie które wymagają zmiany formatu danych a druga wymaga nauki do przedstawienie danych w odpowiedni sposób.
Najbardziej klasycznym sposobem reprezentacji danych, opartym na zmianie formatu danych, jest kodowanie '1 z n'[9]. Jest to prosty sposób polegający na stworzeniu wektora w którym wszystkie wartości poza jedną to zera. Te pojedyncze wartości oznaczane są jako jedynki, oznaczają one występowanie danego zdarzenia lub przedmiotu w danych. Tak więc wektor z jedynka na pierwszym

miejscu oznacza jeden przedmiot, z jedynka na drugim miejscu drugi i tak dalej. Jest to prosta i dość skuteczna metoda. Niestety nie sprawdza się w przypadku rekomendacji bazujących na sesji użytkownika. Gdy tylko zostanie dodany nowy przedmiot, czyli będzie to już reprezentacja '1 z n+1' cały system należy ponownie uczyć dla nowych wartości. Drugim problemem jest wielkość samej reprezentacji. W przypadku gdy na sklepie znajduje się ok 40 tysięcy przedmiotów, a sesje użytkownika mogą mieć długość po 30 to wejście do systemu rośnie do rozmiarów 40 tysięcy na 30. Może to wpłynąć na czas predykcji modelu jak i na ilość zasobów potrzebnych do funkcjonowania modelu.
Kolejna testowaną metodą jest użycie specjalnej warstwy w sztucznej sieci neuronowej jak Embedding layer[15], co stanowi podejście hybrydowe, raz mieniany jest sposób formatowania danych ale reprezentacje otrzymywane są w wyniku nauki. Jest to niezwykle wygodne rozwiązanie, wymaga dostarczenia do sieci danych wejściowych przedstawionych w postaci kolejnych liczb całkowitych, a wyliczeniem reprezentacji zajmie się sama sieć podczas nauki. Jest to znaczące zmniejszenie wielkości danych wejściowych względem metody '1 z N', nieco odbija się na wielkości samej sieci gdyż musi ona w tym momencie posiadać dodatkową warstwę wielkości odpowiadającej liczbie produktów na wielkość reprezentacji. Takie rozwiązanie jest również wygodne, sieć sama uczy się potrzebnych sobie reprezentacji i to przy okazji rozwiązywania docelowego problemu. Niestety występuje tu ten sam problem jak w przypadku '1 z N', po dodaniu nowego przedmiotu sieć trzeba uczyć od nowa.
Inne podejście do tworzenia reprezentacji produktów przedstawia MRNet-Product2Vec[4]. Jest to sieć oparta na warstwach rekurencyjnych[23] której zadaniem jest nauka reprezentacji produktu w oparciu o jego nazwę na wejściu sieci i cechy produktu jako cel nauki. Reprezentację produktów stanowi przedostatnia warstwa sieci, która skupiać powinna wszystkie cechy zależne od nazwy. Wybór cech produktu polegał na znalezieniu takich cech które raz występowały w większości produktów, tak by można je porównać między sobą, a dwa miały sens w oczach typowego użytkownika. Sposób ten daje możliwość tworzenia reprezentacji o stałych wymiarach, w jednej przestrzeni oraz oferuje możliwość dodania nowych przedmiotów bez rozpoczynania procesu nauki od zera. Niestety po sprowadzeniu reprezentacji z docelowej 100 wymiarowej przestrzeni do przestrzeni dwuwymiarowej w celu jej zobrazowania przy pomocy narzędzia TSNE[21] okazało się że proponowane rozwiązanie bardziej niż na cechach przedmiotów skupia się na ich nazwach, co podzieliło przestrzeń produktów na klastry producentów. Nie jest to może najgorsze wyjście, jednak prowadzić może do polecania przez system przedmiotów tylko od jednego producenta. Nie jest to spójne z założeniem o jak największej różnorodności polecanych przedmiotów.
Kolejnym podejściem do tworzenia reprezentacji jest użycie autoencoderów[13]. Są to sztuczne sieci neuronowe których zadaniem jest jak najwierniejsze odtworzenie danych wejściowych. Ta z pozoru nieprzydatna funkcja daje jednak dużo możliwości, jedną z nich jest zmniejszenie wymiarowości danych, inną tworzenie reprezentacji danych. Umożliwia to budowa sieci, złożonej z dwóch części: enkodera i dekodera, która kształtem przypomina klepsydrę. Zadaniem enkodera jest przetworzenie danych wejściowych do pośredniego wektora o wymiarach znacznie mniejszych od wymiarów na wejściu, a zadaniem dekodera jest odtworzenie danych wejściowych na podstawie reprezentacji zawartej w tym pośrednim wektorze. W założeniu sieć będzie szukać takich wartości wektora pośredniego by zawarł on wystarczająco dużo informacji by możliwe było pełne odtworzenie danych wejściowych przez dekoder. Gdy sieć osiągnie odpowiednie wyniki w rekonstrukcji, wektor pośredni można traktować jako reprezentację danych w skompresowanej formie. Do budowy reprezentacji produktów użyto opisanych wcześniej cech produktów. Tu pojawił się pierwszy problem, klasyczna reprezentacja cech, metoda '1 z N' nie sprawdziła się w praktyce. Z jednej strony tak zakodowane cechy zajmowały bardzo dużo miejsca, wektory cech potrafiły mieć wymiary idące w tysiące co skutecznie przedłużało naukę, z drugiej cierpi na ten sam problem co zwyczajne kodowanie '1 z N'. Gdy pojawi się produkt z zupełnie nowa cechą, sieć należy przeliczyć ponownie, stracona zostaje uniwersalność takiego zadania.
Z uwagi na to że cechy produktów są opisane słownie, możliwe jest użycie technik z dziedziny rozpoznawania języka naturalnego (NLP)[12], polegające na przypisaniu słowom wektora liczbowego który je reprezentuje. Do tego celu użyta została biblioteka fasttext[5], którą dotrenowano na korpusie składającym się z opisów i cech przedmiotów. Fasttext umożliwia również tworzenie reprezentacji całych zdań. Zdania te zbudowane były z cech przedmiotów, dzięki czemu otrzymano reprezentacje wektorową każdego przedmiotu. Co ważne taka metoda jest uniwersalna, raz wytrenowany model fasttext bez problemu radzi sobie ze słowami spoza korpusu uczącego, dzięki czemu nowe przedmioty nie stanowią problemu.
Testowana została również metoda hybrydowa, łącząca autoencodery z reprezentacjami cech pozyskanymi z modelu fasttext. Reprezentację tę użyto jako wejście do modelu autoencodera,

a następnie wytrenowano ten model. Tak pozyskana reprezentacja poszczególnych produktów została następnie użyta jako wejście do docelowego systemu rekomendacji. Uzyskane w ten sposób reprezentacje spełniają wymagania zarówno co do wspólnej przestrzeni jak i uniwersalności. Przy każdym nowym produkcie jego cechy zostają zakodowane najpierw przy pomocy fasttext'u a następnie użyte jako wejście do enkodera tworząc reprezentację nowego produktu w tej samej przestrzeni co wcześniejsze. W przebadanych reprezentacjach widać wpływ reprezentacji słów na powiązania między reprezentacjami produktów, jednak jest on znacznie mniejszy niż w przypadku MRNet-Product2Vec.

2 Wynik działania

2.1 Moduł analizy obrazu

Na podstawie powyższego oraz wcześniejszych badań został zaimplementowany proces strojenia modelu oraz moduł rekomendacyjny działający w następujących krokach:

1. Trening wag modelu ResNet50 w architekturze syjamskiej metodą *Dimensionality Reduction by Learning an Invariant Mapping*.
2. Obliczenie wektorów cech produktów znajdujących się w sklepie za pomocą modelu wytrenowanego w poprzednim kroku.
3. Utworzenie drzewa k-wymiarowego złożonego z wektorów cech produktów.
4. Zlokalizowanie obiektu (zegarka) na zdjęciu dostarczonym przez użytkownika używając modelu YOLO wytrenowanego na klasie *watch* zbioru Open Images Dataset.
5. Obliczenie wektora cech zlokalizowanego obiektu za pomocą modelu wytrenowanego metodą *Dimensionality Reduction by Learning an Invariant Mapping*.
6. Wyszukanie w drzewie k-wymiarowym najbliższych sąsiadów używając odległości euklidesowej jako metryki.

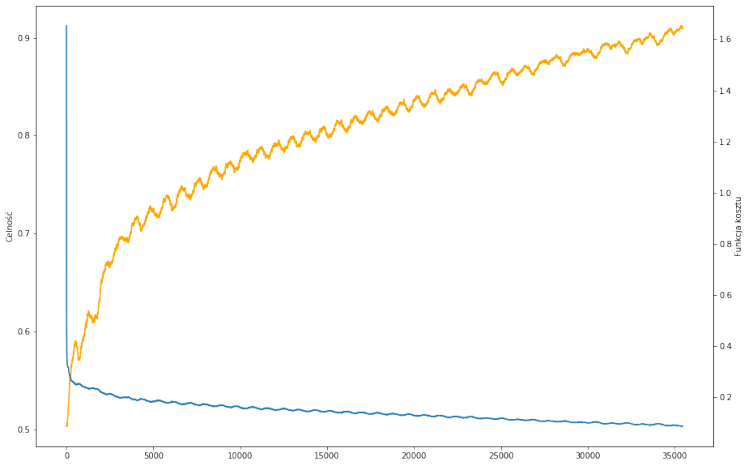
Architektura sieci syjamskiej, złożona z dwóch modeli ResNet50 dzielących ze sobą wagi została zaimplementowana z wykorzystaniem frameworka Keras. Wagi wytrenowane na zbiorze ImageNet zostały wykorzystane jako inicjalizacja tak by maksymalnie poszerzyć źródła danych i poprawić generalizację. Modele ResNet50 łączą się w warstwie Lambda obliczającej odległość L1 między ich wyjściami, na końcu sieci znajduję się pojedyńczy neuron używający sigmoidalnej funkcji aktywacji. Funkcja kosztu *contrastive loss* również została zaimplementowana w tym samym frameworku tak jak w publikacji, z jedną różnicą, polegającą na tym, że pary pozytywne posiadają wyjście 1, a negatywne 0 (odwrotnie niż w publikacji).
Przygotwana została metryka pomocnicza – celność, pomagająca śledzić postęp nauki, oraz klasa testująca model w sposób "produkcyjny".
Ponadto przygotowane zostały metody do generowania zbalansowanego zbioru par obrazów na podstawie datasetu wejściowego, oraz generator danych na potrzeby zasilania sieci syjamskich danymi wejściowymi.
Ze względu na ograniczoną ilość danych wejściowych, 2500 par zdjęć złożonych z zdjęcia produktowego oraz dodatkowego zdjęcia w innych warunkach np. na nadgarstku, zdecydowano się zastosować tzw. *data augmentation* powiększając zbiór zdjęć 10-krotnie. Wykorzystano następujące przekształcenia:

- odbicia poziome,
- przycięcia,
- normalizacja kontrastu
- mnożenie wartości kanałów RGB (manipulacja kolorem),
- skalowanie,
- przesunięcia,
- obroty,
- zmiana perspektywy,
- rozmycia (gaussowskie, średnie, medianowe),

- wyostrzanie,
- zmiana odcienia i saturacji,
- zmiana jasności,
- pochylanie,
- dodawanie szumu

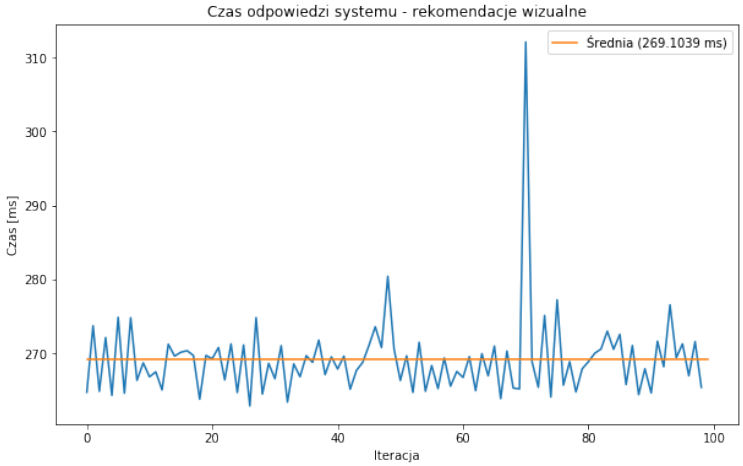
Proces nauki został zobrazowany na 1

Rysunek 1: Proces nauki sieci syjamskiej.

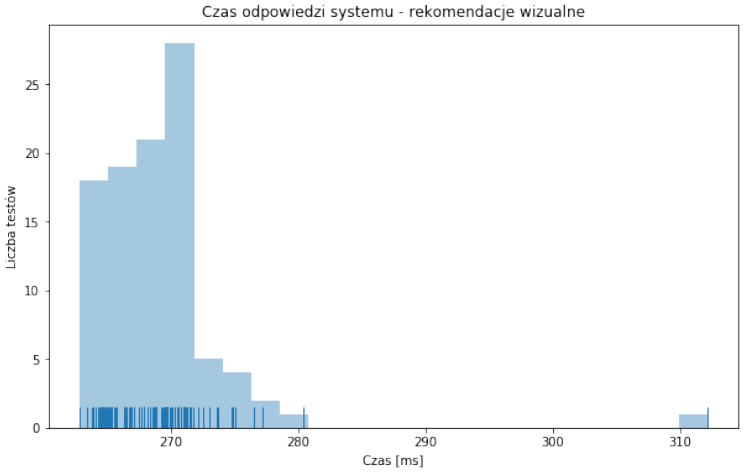


Po przeprowadzonym treningu model został sprawdzony za pomocą klasy testującej **osiągając wynik 88,2%**.
By wygodnie używać wytrenowanego modelu z sieci syjamskiej współdzielone wagi ResNet50 zostały z niej wyciągnięte i zastosowane w nowym modelu skladającym się z pojedynczej sieci ResNet50 uzyskując w ten esktraktor cech. Używając przygotowany ekstraktor obliczono wektory cech dla wszystkich zdjęć *produktowych* ze zbioru danych. Do umieszczenia ich w strukturze danych wykorzystano implementacje drzewa k-wymiarowego pochodzącego z pakietu scikit-learn.
Następnym krokiem jest analiza zdjęcia uzyskanego od użytkownika. W pierwszej kolejności takie zdjęcie zostaje przetworzone przez model YOLOv3 wytrenowanym wcześniej do zadania detekcji zegarka. Sieć zwraca wierzchołki prostokątów wewnątrz których wykryło interesujący nas obiekt oraz "prawdopobieńswa". Moduł rekomendacyjny wybiera prostokąt dla którego wartość "prawdopobieństwa" jest najwyższa i wycina ten obszar z całego zdjęcia przekazując go do dalszej analizy. Wycinek zostaje wprowadzony na wejście tego samego modelu, przez który zostały przetworzone zdjęcia "produktowe", a wektor wejściowy wprowadzony jako zapytanie do drzewa k-wymiarowego. W efekcie uzyskujemy posortowaną listę najbliżej znajdujących się punktów wraz ich indeksami, dzięki czemu możemy uzyskać indentyfikatory podobnych produktów.
Testowanie modelu wykazało, że celność w rozpoznawaniu produktu na zdjęciu wynosi ponad 88% top2, pod warunkiem, że zdjęcie wejściowe od użytkownika miało taki sam charakter jak zdjęcia na których model został wytrenowany, oraz że produkty na zdjęciach testowych znajdowały się w sklepie.
Przygotowany moduł analizy obrazu został zbadany pod względem wydajności, wyniki zostały przedstawione na wykresach 2 3.

Rysunek 2: Czas odpowiedzi systemu rekomendacji wizualnych



Rysunek 3: Histogram czasów odpowiedzi systemu rekomendacji wizualnych

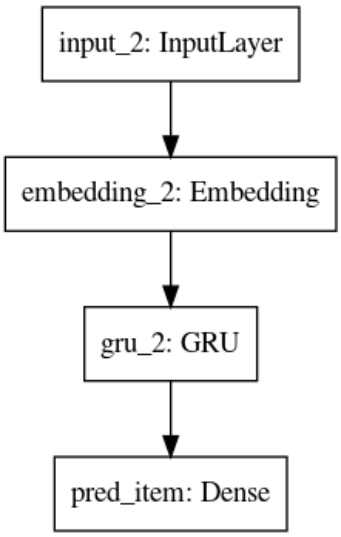


2.2 Rekomendacje behawioralne

Z racji ujęcia problemu rekomendacji bazujących na sesji użytkownika jako problemu klasyfikacji następnego kroku w sekwencji, naturalnym wyborem wydaja się rozwiązania oparte na warstwach rekurencyjnych. Jednak w oparciu o badania przedstawione w [3] na wstępnym etapie badań sprawdzone zostały również rozwiązania oparte o warstwy konwolucyjne[14]. Wstępnie przebadane zostały architektury oparte o jednowymiarowe konwolucje[24], oparte o trójwymiarowe konwolucje[20] oraz rozwiązania oparte na warstwach rekurencyjnych[11, 10, 7]. Sieci oparte o warstwy rekurencyjne i jednowymiarowe konwolucje posiadały same dane na wejściu, to jest reprezentacje przedmiotów w postaci uporządkowanej względem czasu listy. Inne podejście zostało wykorzystane w przypadku sieci opartych na trójwymiarowych konwolucjach. W tym przypadku głównym zamysłem jest wykorzystanie jak największej ilości informacji, zatem dane wejściowe zostały poszerzone, tak by nie składały się tylko z produktów odwiedzonych przez użytkownika ale również z pozostałych zdarzeń mających miejsce podczas danej sesji.

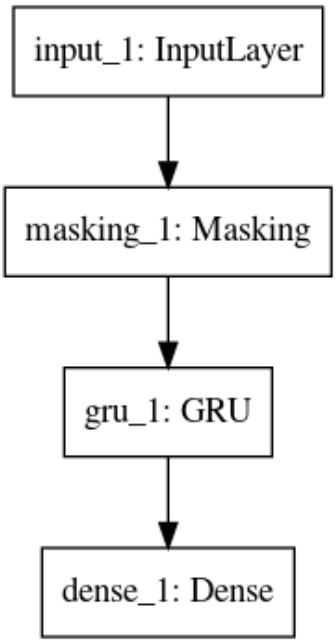
Wstępne badania nad architekturami modeli rekomendacyjnych zostało przeprowadzone na małej części dostępnego zestawu danych, dzięki czemu możliwe było relatywnie szybkie wytrenowanie sieci i skupienie badań na najlepiej rokującej metodzie. I tak, mimo obiecujących badań nad użyciem sieci konwolucyjnych do klasyfikacji sekwencji, przebadane modele oparte o jedno i trój wymiarowe konwolucje sprawdziły się znacznie gorzej od rozwiązań rekurencyjnych, nie przekraczając jednoprocentowego progu celności. Nie stanowi to oczywiście dowodu na uniwersalną wyższość rozwiązań rekurencyjnych, możliwe że przedstawiony problem jest na tyle specyficzny, że rozwiązania rekurencyjny są dla niego najodpowiedniejsze. Z uwagi na powyższe, główne badania skupione zostały na sztucznych sieciach neuronowych opartych na warstwach rekurencyjnych.

Rysunek 4: Model A



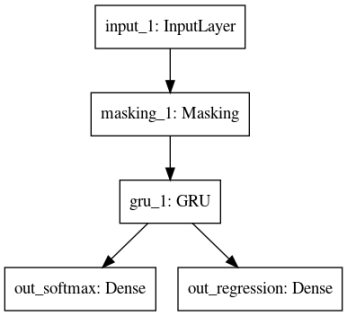
Testowane modele oparte były głównie o architekturę „GRU4REC”[11], z modyfikacjami dotyczącymi danych wejściowych i często również samej sieci. Model A jest implementacja rozwiązania „GRU4REC”, jak przedstawia obraz 4 model składa się z warstwy wejściowej, podążającej za nią, opisanej wcześniej przy okazji reprezentacji danych, warstwy Embedding której zadaniem jest tworzenie reprezentacji produktów, następnie warstwy GRU[6] której zadaniem jest rekurencyjne analizowanie sesji użytkownika, następnie warstwy wyjściowej, dokonującej kategoryzacji sesji i przypisującej prawdopodobieństwo bycia następnym przedmiotem w sesji wszystkim produktom w katalogu sklepu. Udoskonaloną wersję modelu A przedstawia model B, który w odróżnieniu od modelu A, wyposażony został w mechanizm regulacji[22], co pozwoliło osiągnąć lepsze wyniki, mechanizm ten został użyty również w pozostałych modelach.

Rysunek 5: Model C



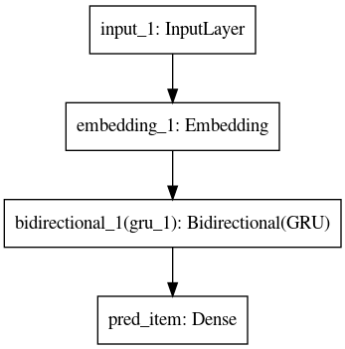
Z kolei model C, przedstawiony na rysunku 5 oparty na podobnej architekturze, różnił się od dwóch poprzednich danymi wejściowymi. Tak jak model A i B same uczyły się reprezentacji produktów dzięki Embedding layer, tak model C korzystał z wyuczonych wcześniej reprezentacji za pomocą MRNet-Product2Vec. Zauważyć można również warstwę „Masking", jako drugą w kolejności, dzięki niej model może przyjmować sesje o różnych długościach poprzez maskowanie braków gdy sesja jest krótsza niż maksymalna jaką może przyjąć sieć.

Rysunek 6: Model D



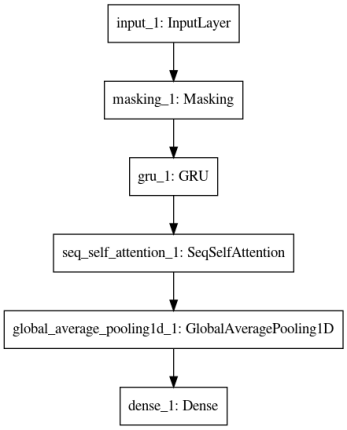
Jeszcze inne modyfikacje przedstawia model D, który zainspirowany uczeniem wielozadaniowym[17] uczył się dwóch rzeczy naraz. Wejściem do tego modelu również są reprezentacje pozyskane przez MRNet-Product2Vec. Jak widać na rysunku 6 model ten posiada dwa wyjścia. Pierwsze, „out_softmax" jest identyczne z poprzednimi modelami, dokonuje klasyfikacji następnego produktu, drugie, nazwane „out_regression", jest zadaniem regresyjnym, w którym sieć musi odtworzyć reprezentacje następnego produktu. Drugie wyjście z sieci miało za zadanie stabilizacje i regularyzacje nauki głównego zadania.

Rysunek 7: Model E



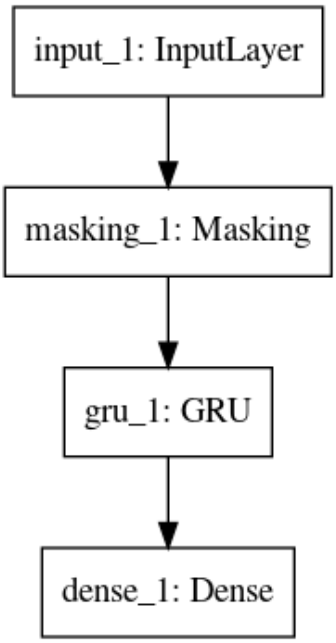
Z kolei model E, rysunek 7, zainspirowany został architekturami używanymi w dziedzinie przetwarzania języka naturalnego[19], składa się z niejako dwóch warstw GRU, jednej sprawdzającej sesje od prawej do lewej i drugiej od lewej do prawej. Główną motywacja jest zapominanie przez warstwy rekurencyjne współcześniejszych kroków, stąd druga warstwa patrząca odwrotnie na sekwencje ma tę pamięć nieco odświeżyć. Ma to pomóc lepiej reprezentować sekwencje a co za tym idzie, polepszyć wyniki predykcji sieci. Model ten stanowi rozwinięcie modelu A, również korzysta z warstwy Embedding.

Rysunek 8: Model F



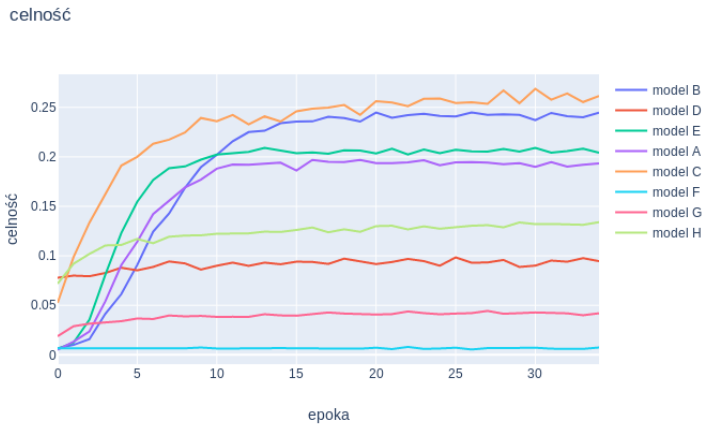
Model F, przedstawiony na rysunku 8, został również zainspirowany metodami wywodzącymi się z dziedziny przetwarzania języka naturalnego, w odróżnieniu od modelu E, ten korzysta z mechanizmu atencji[2]. Mechanizm ten powinien pomóc sieci skupić się na najważniejszych częściach sesji użytkownika, poprzez przypisywaniu wag poszczególnym krokom w sekwencji, przez co dostarczać warstwie klasyfikującej wyższej jakości sygnału.

Rysunek 9: Model G

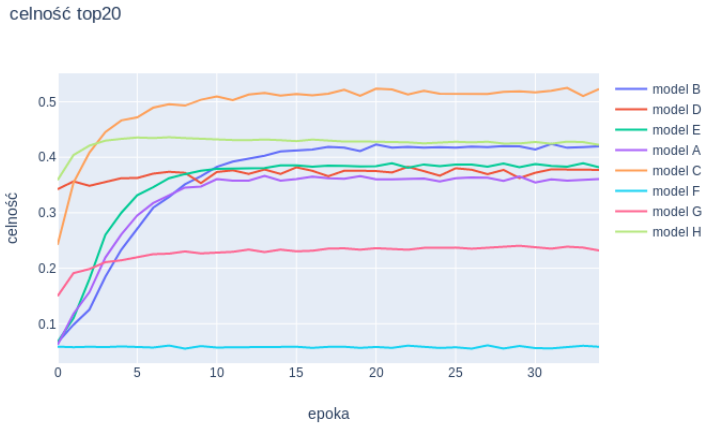


Model G 9, architekturą nie rożni się od modelu C, główna różnica między tymi modelami jest sposób reprezentacji wejścia. W przypadku modelu G dane wejściowe reprezentowane są poprzez wektor utworzony za pomocą algorytmu fasttext. Z kolei model H, z budowy identyczny do modelu G, jako wejścia używa wektorów pozyskanych z hybrydowego połączenie autoencodera z wektorami fasttext.

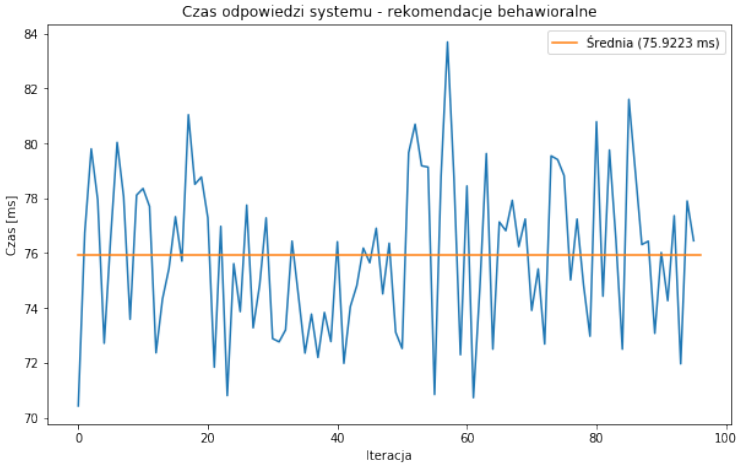
Rysunek 10: Celność TOP1



Rysunek 11: Celność TOP20

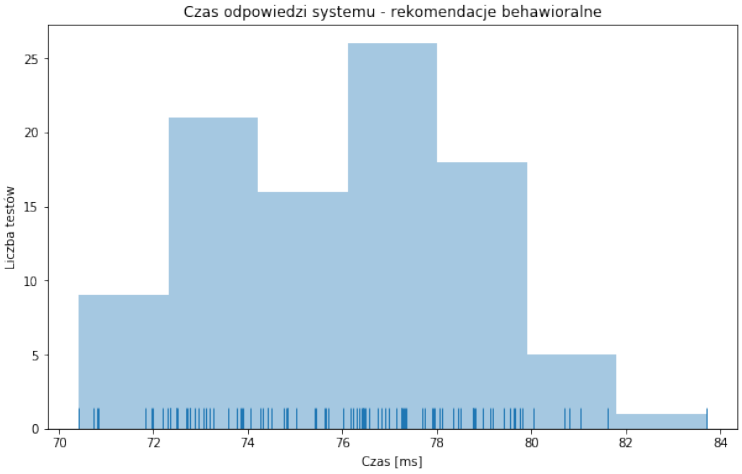


Wyniki ukazane na wykresach 10 11 przedstawiają celność modelu, procent trafnie przewidzianych następnych kroków użytkownika, w odniesieniu do epok nauki modelu, dzięki czemu obserwować można nie tylko wynik jaki osiąga model ale również porównać trening różnych architektur. Wykresy przedstawiają celność modelu w top X zwróconych wyników, to jest top 1 oznacza trafność modelu wśród pierwszego zwróconego wyniku, top 10 oznacza trafność modelu wśród 10 pierwszych zwróconych przez niego wyników i tak dalej. I tak, patrząc na wykres 10 widać że rozwiązania bazujące na mechanizmie atencji, model F, o ile znajdują zastosowanie w przetwarzaniu języka naturalnego tak w przypadku tego problemu zawiodły. Podobnie jak uczenie wielozadaniowe, model G, jak i reprezentowanie danych wejściowych przy pomocy wektorów fasttext również nie dało dobrych rezultatów. Można również zwrócić uwagę na kształt wykresu tych trzech modeli, widać dosyć płaską krzywą wyników względem epok nauki. Oznaczać to może że model na początku trafił w pewne lokalne minimum z którego nie potrafił już wyjść, przez to wynik tych modeli nie polepszał się z czasem. Prym za to wiodą rozwiązania bazujące na reprezentacjach MRNet-Product2Vec, model C, jak i Embedding layer z regularyzacją, model B. Widać tutaj również siłę działania tak prostego zabiegu jak regularyzacja warstw zastosowanego w modelu B względem modelu A. Jeśli przyjrzeć się wynikom bazując na top 10 wynikach, wykres 11 najlepszym modelem pozostaje model C, jednak dosyć mocno wybija się model H, bazujący na hybrydowej reprezentacji danych wejściowych, nieznacznie tylko gorszy od modelu B. Różnica to w ogóle zanika gdy pod uwagę wzięte zostanie top 20 wyników, gdzie model H ustępuje już tylko modelowi C. Przewaga którego jest jednak dość znaczna, prawie 10 punktów procentowych.
Przygotowany moduł rekomendacji został zbadany pod względem wydajności, wyniki zostały przedstawione na wykresach 12 13.



Rysunek 12: Czas odpowiedzi systemu rekomendacji behawioralnych

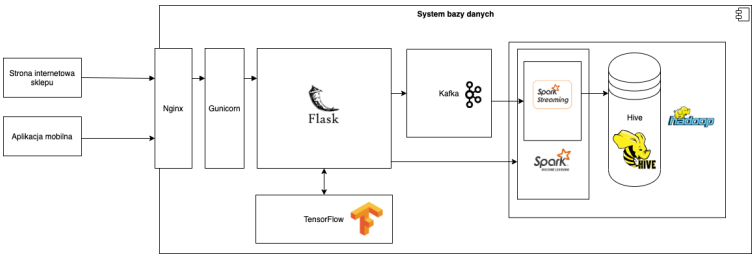
Rysunek 13: Histogram czasów odpowiedzi systemu rekomendacji behawioralnych



2.3 Opis architektury systemu bazy danych

Użytkownicy poruszający się po stronie internetowej sklepu wykonują ogromną liczbę akcji. Dane te mogą zostać użyte w celu polepszenia jakości rekomendacji. Wykorzystanie danych wymaga jednak przechowywania w sposób, który pozwoli na łatwy dostęp i ich przetwarzanie. W tym celu został stworzony system, który na to pozwoli.
System został zaprojektowany w sposób umożliwiający zapis dużych wolumenów danych. Wykorzystano do tego następujące technologie: JavaScript, Flask, Kafka, Spark, Hadoop oraz Hive. JavaScript umożliwił komunikację z serwerem po stronie klienta. Z pomocą frameworku Flask zaimplementowano API w języku Python. Kafka została użyta jako broker wiadomości w systemie rozproszonym. Spark został wykorzystany jako silnik zapytań. Hive został użyty do przetwarzania zapytań do bazy danych w języku HQL.
Po stronie klienta, podczas korzystania ze strony internetowej, użytkownicy wykonują akcje, które są zapisywane w pamięci podręcznej przeglądarki. Skrypty śledzą akcje użytkownika takie jak wyświetlenie strony produktu, kliknięcia w elementy strony, wyszukiwania, przeglądanie katalogu.

Zdarzenia te zostają wysyłane do serwera wraz z informacją o czasie zdarzenia oraz identyfikatorze użytkownika, który je wykonał.
Dane po stronie serwera są odbierane przez Nginxa, czyli serwer WWW, który przekazuje zapytania do Gunicorna odpowiedzialnego za przekazanie zapytań do aplikacji. Zaimplementowany w języku Python, z użyciem frameworku Flask, serwer jest odpowiedzialny za odbiór danych. W aplikacji jest także zaimplementowane wstępne przetworzenie danych, aby ułatwić ich dalsze przetwarzanie w systemie.
Zdarzenia następnie zostają wysłane do Kafki, czyli brokera wiadomości. W ramach projektu wykorzystano aplikację Kafka. Pozwala ona w niezawodny i wydajny sposób na komunikację między elementami systemu. Kafka umożliwia także pracę w systemie rozproszonym, a tym samym pozwala na skalowanie jej wszerz. Kafka zapisuje dane początkowo w pamięci komputera co pozwala na wydajną komunikację między konsumentami wiadomości. Dodatkowo dane zostają też zapisane na dysku, dzięki czemu, nawet jeżeli dalsze elementy systemu zawiodą, to jest możliwe odczytanie wstrzymanych danych i poprawne wznowienie pracy systemu bez utraty żadnych informacji.
Do brokera łączy się aplikacja, która co 1 sekundę pobiera wszystkie nowe wiadomości, a następnie zapisuje je do bazy danych. Aplikacja ta jest napisana w języku Scala w środowisku Spark, który wykorzystuje Hadoopa do zarządzania zasobami. Pozwala to na rozproszenie przetwarzania danych w równomierny sposób na wszystkie maszyny w klastrze.
Na końcu dane zostają zapisywane w bazie danych Hive. Działa ona także na Hadoopie dzięki czemu dane zostają rozproszone i zduplikowane na wile maszyn. Pozwala to na szybsze przetwarzanie danych a także polepsza niezawodność systemu. Tak składowane dane pozwalają na szybkie zapytania do bazy takie jak ostatnio odwiedzone przez użytkownika produkty, ilość zdarzeń zapisanych w systemie, czy znalezienie najczęściej wyświetlanych i sprzedawanych produktów. Hive wykorzystuje do tego specjalną odmianę języka SQL, która nazywa się HQL. Zapytania do bazy danych zostają następnie przetworzone na odpowiadające im zapytania w Sparku, które zostają rozproszone na wiele maszyn, a następnie połączone w wynikowy rezultat.
W celu szybkiego odpytywania systemu, API zostało podłączone za pomocą Sparka do bazy danych. Taka konfiguracja pozwala na wyświetlenie ostatnio odwiedzonych produktów po stronie klienta oraz zagregowanych danych takich jak liczba produktów dodanych do koszyka czy wartość sprzedanych przedmiotów.
API umożliwiło także serwowanie rekomendacji wizualnych, do którego łączy się aplikacja mobilna. Na podstawie przesłanego zdjęcia zostają wybrane podobne produkty z katalogu. Wykorzystany do tego model powstał w Kerasie na podstawie zdjęć pochodzących ze sklepu internetowego. Serwer obsługuje także rekomendacje kontekstowe na podstawie ostatnio odwiedzonych przez użytkownika produktów.



2.4 Aplikacja mobilna

Aplikacja została wzbogacona o możliwość wykonania zdjęcia. Wykorzystano do tego bibliotekę CameraX, która jest odpowiedzialana za poprawne wyświetlanie podglądu z kamery oraz zapis

metoda, ścieżka	(GET) /count
parametry	Brak
odpowiedź	{ćount": 13}
opis	Zwraca liczbę zapisanych zdarzeń

metoda, ścieżka	(POST) /event
parametry	Dane zdarzenia w ciele zapytania. Przykładowe dane: [[“https:website.plproduct_path”, “cccccccc-aaaa-eeee-bbbb-dddddddddddd”, “”], {“type”:“product”, “timestamp”:1569323308873, “context”:{“sku”:“SKU_OF_PRODUCT”, “price”:“78.00”, “finalPrice”:“78.00”, “currency”:“PLN"}}]
odpowiedź	OK
opis	Pozwala wysłać zdarzenie do API

metoda, ścieżka	(GET) /visited/<uid>
parametry	Parametr ścieżki <uid>: identyfikator użytkownika
odpowiedź	{ "visited": [Śuunto-SS023", Śuunto-SS050", Śuunto-SS024", Śuunto-SS026", Śuunto-SS055"]}
opis	Zwraca ostatnio odwiedzone przez użytkownika produkty

metoda, ścieżka	(GET) /actions/<uid>
parametry	Parametr ścieżki <uid>: identyfikator użytkownika
odpowiedź	{ ąctions": [{ "timestamp": 1565690458006, "type": ćlose"}, { "timestamp": 1565690459189, "type": ópen"}, { "timestamp": 1565690464727, "type": łisting"}, { "timestamp": 1565690490654, "type": ćlose"}]}
opis	Zwraca ostatnio wykonane akcje przez użytkownika

metoda, ścieżka	(GET) /most_visited
parametry	Brak
odpowiedź	{ "most_visited": [{ ćount": 133, śku": "Timex-24"}, { ćount": 123, śku": Śuunto-SS023"}, { ćount": 109, śku": Śuunto-SS050"}]}
opis	Zwraca najczęściej odwiedzane produkty

metoda, ścieżka	(GET) /most_visited
parametry	Brak
odpowiedź	{ "most_visited": [{ ćount": 133, śku": "Timex-24"}, { ćount": 123, śku": Śuunto-SS023"}, { ćount": 109, śku": Śuunto-SS050"}]}
opis	Zwraca najczęściej odwiedzane produkty

metoda, ścieżka	(GET) /asl
parametry	Brak
odpowiedź	{ "data": [{ żating": 1, śku": Śuunto-SS050", "uid": "dddddddd-aaaa-eeee-ddddd-bbbbbbbbbbbb"}, { żating": 1, śku": Ćitizen-BN", "uid": "dddddddd-eeee-aaaa-ddddd-bbbbbbbbbbbb"}, { żating": 1, śku": Źeppelin-76", "uid": "bbbbbbbb-aaaa-eeee-ddddd-dddddddddddd"}]}
opis	Zwraca interakcje/oceny użytkowników z produktami

metoda, ścieżka	(POST) /context_recommendation.json
parametry	Typ: Multipart formPole "imagefile": obraz
odpowiedź	OK
opis	Pozwala uzyskać rekomendacje wizualne na podstawie wysłanego zdjęcia produktu.

metoda, ścieżka	(POST) /similar_products.json
parametry	Lista ostatnio odwiedzonych produktów w ciele zapytania.Przykładowe dane:{"visited": ["Timex-TW", Śuunto-SS022", Ćasio-GW"]}
odpowiedź	OK
opis	Pozwala uzyskać rekomendacje na podstawie odwiedzonych przez użytkownika produktów.

wykonanego zdjęcia. Obraz jest zapisywany w pamięci tymczasowej urządzenia. Zintegrowano także komunikację z API, co pozwala na wysłanie zdjęcia do systemu rekomendacyjnego. Po otrzymaniu odpowiedzi wyświetlana jest lista podobnych zegarków. Znajdują się na niej informacje takie jak nazwa i kod produktu oraz jego zdjęcie. Kliknięcie na produkt umożliwia przejście na stronę produktu.

2.5 Raport z badań UX

2.5.1 Wstęp

Badania User Experience pozwalają poznać rzeczywiste potrzeby oraz zachowania użytkowników. Dostarczają informacji umożliwiających stworzenie użytecznego, prostego w obsłudze i budzącego pozytywne emocje produktu cyfrowego.
W projekcie aplikacji mobilnej służącej do klasyfikacji marki wykrytego zegarka zostały przeprowadzone badania służące weryfikacji projektu oraz zidentyfikowaniu potencjalnych możliwości rozwoju. Pozwoliły one zlokalizować obszary wspierające lub utrudniające realizację celów, jak również elementy wpływające pozytywnie lub negatywnie na odbiór aplikacji.
Na podstawie niniejszych wyników badań i obserwacji zostały wprowadzone zmiany do warstwy wizualnej aplikacji oraz powstała lista rekomendowanych rozwiązań.

2.5.2 Metodologia

Badanie składało się z trzech części: testu klasyfikacji, testu 5 sekund oraz zadania scenariuszowego.

Test klasyfikacji Celem testu klasyfikacji było sprawdzenie, w jaki sposób badani klasyfikują produkty jako podobne.

Test 5 sekund Głównym celem było poznanie pierwszych opinii użytkowników oraz elementów, które przykuły ich uwagę.

Zadanie scenariuszowe Celem zadania scenariuszowego było sprawdzenie, jak użytkownicy radzą sobie z wykonaniem typowego działania w aplikacji.

2.5.3 Uczestnicy badania

W badaniu wzięło udział 5 osób, ze średnim i wyższym wykształceniem, reprezentujących grupy wiekowe 35-44 lata (2 osoby) oraz 45-54 lata (3 osoby). Przebadani respondenci korzystają codziennie z Internetu, najczęściej wybierając do tego celu telefon komórkowy (4 osoby) lub laptop (1 osoba). Badani oceniają swoje umiejętności korzystania z Internetu oraz aplikacji mobilnych jako dobre.

2.5.4 Test klasyfikacji

W przyszłości aplikacja poza rozpoznawaniem marki zegarka ma także sugerować podobne produkty. Dlatego głównym celem przeprowadzenia testu klasyfikacji zegarków było sprawdzenie, w jaki sposób badani dobierają podobne produkty. Jakie cechy produktu są dla nich najważniejsze? Czy zwracają uwagę na markę, kolor, kształt?
Dzięki wynikom z tego badania możliwe będzie dopracowanie algorytmów, aby proponowały produkty, których oczekuje realny użytkownik.

Zadanie składało się z trzech etapów i polegało na dobraniu 4 pasujących zegarków do wskazanego, z prezentowanego poniżej zbioru 20 zegarków. Kolejność dobieranych zegarków została zachowana.

Pierwszy etap polegał na dobraniu czterech podobnych zegarków do damskiego zegarka Cluse La Boheme Mesh. Badani w bardzo podobny sposób dopasowali zegarki, zwracając uwagę przede wszystkim na kształt tarczy, kolorystykę oraz podobne wskazówki.

W drugim etapie zadania uczestnicy dobierali produkty do męskiego zegarka Lacoste San Diego. Badani jako kryterium przyjęli m.in. styl (np. sportowa elegancja), markę, bransoletę oraz kolor.

W ostatnim etapie należało dopasować zegarek do męskiego Fossil Grant. Uczestnicy badania w przypadku tego zegarka zwracali uwagę na masywność, skomplikowanie, ale także pasek.

Podsumowując przeprowadzone badanie, użytkownicy dobierając podobne produkty, stosują kilka różnych kryteriów równocześnie. Dla badanych ważniejsze były takie cechy jak kolor, styl czy kształt niż marka. W pierwszym etapie zadania Uczestnik 2 oraz Uczestnik 5 udzielili takich samych odpowiedzi, ale w innej kolejności.

2.5.5 Test 5 sekund

Kolejnym etapem badania z użytkownikami był test 5 sekund. Polega on na tym, że uczestnikowi wyświetlany jest przez kilka sekund pierwszy ekran aplikacji, następnie jest on proszony o opisanie swoich wrażeń.
Dzięki temu możliwe jest odnotowanie, jaki komunikat przekazuje aplikacja swoim użytkownikom, które elementy zostają przez użytkowników zapamiętane, a które pominięte.

Uczestnicy badania zwracali uwagę na to, że aplikacja jest mało atrakcyjna wizualnie. Brakowało im ekranu, który tłumaczyłby, w jakim celu powstała aplikacja oraz w jaki sposób należy z niej korzystać. Trzy osoby zwróciły uwagę na to, że nie potrzebują aż tak precyzyjnych wyników po przecinku. Badanym trudno było rozróżnić, które elementy interfejsu są klikalne, a które nie.

2.5.6 Wnioski i rekomendacje

W drodze analizy danych z badania zidentyfikowano szereg problemów funkcjonalnych projektu. Pomimo ocenienia wyglądu aplikacji jako nieatrakcyjnego badani wyrażali zaciekawienie aplikacją mobilną i chęć korzystania z takiej aplikacji w przyszłości.
Dzięki przeprowadzonym badaniom możliwe będzie ulepszenie ogólnego odbioru aplikacji oraz wyeliminowanie problemów pojawiających się podczas badania.
Wszystkie odnalezione problemy dotyczące użyteczności zostały ocenione pod względem ich znaczenia dla funkcjonowania aplikacji i jej użyteczności.

Wysoki priorytet

1. Dodanie ekranu startowego informującego o celu i sposobie korzystania z aplikacji.

2. Prezentowanie finalnego, jednego wyniku na osobnym ekranie, tak aby użytkownik był pewny, że pomiar się zakończył.

3. Poprawa warstwy graficznej, ograniczenie wyświetlania ilości liczb po przecinku.

Średni priorytet

1. Wyświetlanie nie tylko zidentyfikowanego zegarka, ale także podobnych.

2. Możliwość przekierowania do strony internetowej na której użytkownik może kupić znaleziony zegarek.

3. Historia wyszukiwanych zegarków.

4. Dodanie nawigacji aplikacji: o aplikacji, przycisk skanowania, historia wyszukiwań.

Niski priorytet

1. Zidentyfikowanie nie tylko marki zegarka, ale także jego modelu.

2. Zmiana nazwy aplikacji.

Wprowadzone zmiany Badania wpłynęły na stworzenie, nowego, bardziej atrakcyjnego designu aplikacji prezentowanego poniżej.


4:28
71%
WRPO app
NAUTICA 37.02%
TIMEX 30.85%
SWATCH 13.72%
MORELLATO 13.44%
TOMMY_HILFIGER 3.45%
ATLANTIC 1.11%
CERTINA 0.10%
GUESS 0.10%
ADRIATICA 0.07%
DIESEL 0.04%

3 Wnioski z przeprowadzonego badania

Podczas prac nad trzecim etapem projektu została rozwinięta aplikacja mobilna z szczególnym uwzględnieniem poprawienia tzw. user experience. Zostały wykonane rzeczywiste testy dostarczające

informacji o oczekiwaniach użytkowników.
Moduł analizy obrazu został rozwinięty i dostosowany do pracy w warunkach produkcyjnych. Jego celność wyniosła 88,2% top 2 dla danych testowych przy czasie działania około 270ms. Zrealizowano wdrożenie modułu rekomendacyjnego opierającego się dodatkowo na historycznych wyborach użytkowników. Jej czas działania również został zbadany, a średnia czasu działania wyniosła 76ms.

4 Zgodność realizacji działania z założeniami z Planu B+R

Powyższe badania oraz inne działania spełniły założenia Planu B+R, zrealizowano założenia kamieni milowych:

- Działający i przetestowany moduł rekomendujący
- Działający kompletny system gotowy do pracy w rzeczywistych warunkach.
- Uzupełniona baza danych uczących.
- Wskaźnik rezultatu: Średnia trafność rozpoznawania: przynajmniej 86% top 2
- Wskaźnik rezultatu: Średni czas rozpoznawania obiektu: poniżej 1,5 sekundy

Literatura

[1] Aggarwal and Charu. Recommender systems: The textbook.

[2] Bahdanau, Dzmitry, Kyunghyun Cho, , and Yoshua Bengio. Neural machine translation by jointly learning to align and translate.

[3] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling.

[4] Arijit Biswas, Mukul Bhutani, and Subhajit Sanyal. Mrnet-product2vec: A multi-task recurrent neural network for product embeddings.

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information.

[6] Cho, Kyunghyun; van Merrienboer, Bart; Gulcehre, Caglar; Bahdanau, Dzmitry; Bougares, Fethi; Schwenk, Holger; Bengio, and Yoshua. Learning phrase representations using rnn encoder-decoder for statistical machine translation.

[7] Alexander Dallmann, Alexander Grimm, Christian Pölitz, Daniel Zoller, and Andreas Hoth. Improving session recommendation with recurrent neural networks by exploiting dwell time.

[8] Elahi, Mehdi; Ricci, Francesco; Rubens, and Neil. A survey of active learning in collaborative filtering recommender systems.

[9] Harris, David, Harris, and Sarah. Digital design and computer architecture.

[10] Balázs Hidasi and Alexandros Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations.

[11] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks.

[12] Hutchins and J. The history of machine translation in a nutshell.

[13] Kramer and Mark A. Nonlinear principal component analysis using autoassociative neural networks.

[14] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning aplied to document recognition.

[15] Zhongliang Li, Raymond Kulhanek, Shaojun Wang, Yunxin Zhao, and Shuang Wu. Slim embedding layers for recurrent neural language models.

[16] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. http://www.inf.unibz.it/~ricci/papers/intro-rec-sys-handbook.pdf.

[17] Romera-Paredes, B., Argyriou, A., Bianchi-Berthouze, N., and Pontil. Exploiting unrelated tasks in multi-task learning.

[18] Stuart J. Russell and Peter Norvig. Artificial intelligence: A modern approach.

[19] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learningwith neural networks.

[20] Trinh Xuan Tuan, Hanoi, and Vietnam Tu Minh Phuong. 3d convolutional networks for session-based recommendation with content features.

[21] van der Maaten, L.J.P.; Hinton, and G.E. Visualizing data using t-sne.

[22] Twan van Laarhoven. L2regularization versus batch and weightnormalization.

[23] Williams, Ronald J.; Hinton, Geoffrey E.; Rumelhart, and David E. Learning representations by back-propagating errors.

[24] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. A simple convolutional generative network for next item recommendation.

[25] Markus Zanker and Markus Jessenitschnig. Collaborative feature-combination recommender exploiting explicit and implicituser feedback.

Poznań, 29.11.2019r. .....
(miejscowość, data sporządzenia raportu, podpis autorów)

Poznań, 29.11.2019r. .....
(miejscowość, data sporządzenia raportu, podpis pracodawcy)