Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-182905-74349

Bc. Andrej Vítek

Pravdepodobnostné prístupy pre analýzu vykonávanej úlohy na základe pohybu očí

Diplomová práca

Študijný program:	Inteligentné softvérové systémy
Študijný odbor:	9.2.5 Softvérové inžinierstvo (hlavný), 9.2.8 Umelá inteligencia (vedľajší)
Miesto vypracovania:	Ústav informatiky, informačných systémov a softvérového inžinierstva
Vedúci práce:	Mgr. Jozef Tvarožek, PhD.

Apríl, 2019

STU FIIT
SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Zadanie diplomovej práce

Meno študenta: **Bc. Andrej Vítek**

Študijný program: Inteligentné softvérové systémy

Študijný odbor: Softvérové inžinierstvo – hlavný študijný odbor
Umelá inteligencia – vedľajší študijný odbor

Názov práce: **Pravdepodobnostné prístupy pre analýzu vykonávanej úlohy na základe pohybu očí**

Samostatnou výskumnou a vývojovou činnosťou v rámci predmetov Diplomový projekt I, II, III vypracujte diplomovú prácu na tému, vyjadrenú vyššie uvedeným názvom tak, aby ste dosiahli tieto ciele:

Všeobecný cieľ:

Vypracovaním diplomovej práce preukážte, ako ste si osvojili metódy a postupy riešenia relatívne rozsiahlych projektov, schopnosť samostatne a tvorivo riešiť zložité úlohy aj výskumného charakteru v súlade so súčasnými metódami a postupmi študovaného odboru využívanými v príslušnej oblasti a schopnosť samostatne, tvorivo a kriticky pristupovať k analýze možných riešení a k tvorbe modelov.

Špecifický cieľ:

Vytvorte riešenie zodpovedajúce návrhu textu zadania, ktorý je prílohou tohto zadania. Návrh bližšie opisuje tému vyjadrenú názvom. Tento opis je záväzný, má však rámcový charakter, aby vznikol dostatočný priestor pre Vašu tvorivosť.

Riaďte sa pokynmi Vášho vedúceho.

Pokiaľ v priebehu riešenia, opierajúc sa o hlbšie poznanie súčasného stavu v príslušnej oblasti, alebo o priebežné výsledky Vášho riešenia, alebo o iné závažné skutočnosti, dospejete spoločne s Vaším vedúcim k presvedčeniu, že niečo v texte zadania a/alebo v názve by sa malo zmeniť, navrhnite zmenu. Zmena je spravidla možná len pri dosiahnutí kontrolného bodu.

Miesto vypracovania: Ústav informatiky, informačných systémov a softvérového inžinierstva, FIIT STU v Bratislave

Vedúci práce: **Mgr. Jozef Tvarožek, PhD.**

Termíny odovzdania:

Podľa harmonogramu štúdia platného pre semester, v ktorom máte príslušný predmet (Diplomový projekt I, II, III) absolvovať podľa Vášho študijného plánu

Predmety odovzdania:

V každom predmete dokument podľa pokynov na www.fiit.stuba.sk v časti:
home > Informácie o > štúdiu > harmonogram štúdia > diplomový projekt.

V Bratislave dňa 12. 2. 2018

SLOVENSKÁ TECHNICKÁ UNIVERZITA
V BRATISLAVE
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4
1

prof. Ing. Pavol Návrat, PhD.
riaditeľ Ústavu informatiky, informačných systémov
a softvérového inžinierstva

STU FIIT
SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Návrh zadania diplomovej práce

Finálna verzia do diplomovej práce [1]

Študent:

Meno, priezvisko, tituly: Andrej Vítek, Bc.
Študijný program: Inteligentné softvérové systémy
Kontakt: andrej2024@gmail.com

Výskumník:

Meno, priezvisko, tituly: Jozef Tvarožek, Mgr. PhD.

Projekt:

Názov:	Pravdepodobnostné prístupy pre analýzu vykonávanej úlohy na základe pohybu očí
Názov v angličtine:	Probabilistic models for analysis of performed task based on eye movement
Miesto vypracovania:	Ústav informatiky, informačných systémov a softvérového inžinierstva, FIIT STU, Bratislava
Oblasť problematiky:	sledovanie pohľadu, analýza dát

Text návrhu zadania[2]

Využitie dynamických webových aplikácií na riešenie pracovných alebo osobných úloh sa stáva každodennou súčasťou nášho života. Spôsob ako sa ľudia pozerajú, odráža nielen vykonávanú úlohu a úroveň ich schopností v príslušnej doméne, ale aj informácie o vlastnostiach pozorovateľa. Výskum v oblasti analýzy postupností pohybov očí sa pokúša nájsť nové metódy analýzy pre efektívnejšie určenie vykonávanej úlohy alebo charakteristík pozorovateľa použitím informácií z pohľadu. Pravdepodobnostné metódy umožňujú modelovať vzory pohybov očí pri vykonávaní úlohy ako prechody medzi význačnými oblasťami vo vizuálnom stimule, čo umožňuje vhodne zachytiť stochastický charakter pohybov očí.

Analyzujte možnosti pravdepodobnostných metód na modelovanie a analýzu pohybov očí. Navrhnite novú metódu, prispôsobenú na analýzu vykonávanej úlohy vo vybranej doméne (napr. online nakupovanie, výučba). Navrhnutú metódu overte vykonaním používateľského experimentu. Pri overení využite fakultné výskumné centrum používateľského zážitku a interakcií.

---

[1] Vytlačiť obojstranne na jeden list papiera

[2] 150-200 slov (1200-1700 znakov), ktoré opisujú výskumný problém v kontexte súčasného stavu vrátane motivácie a smerov riešenia

Literatúra[3]

- Coutrot, Antoine, Janet H. Hsiao, and Antoni B. Chan., "Scanpath modeling and classification with hidden Markov models.", Springer, Behavior Research Methods: 1-18., 2017.
- Coutrot, A., Binetti, N., Harrison, C., Mareschal, I., & Johnston, A., "Face exploration dynamics differentiate men and women.", The Association for Research in Vision and Ophthalmology, Journal of vision, 16(14), 16-16, 2016.

Vyššie je uvedený návrh diplomového projektu, ktorý vypracoval(a) Bc. Andrej Vítek, konzultoval(a) a osvojil(a) si ho Mgr. Jozef Tvarožek, PhD. a súhlasí, že bude takýto projekt viesť v prípade, že bude pridelený tomuto študentovi.

V Bratislave dňa 9.1.2018

Podpis študenta

Podpis výskumníka

Vyjadrenie garanta predmetov Diplomový projekt I, II, III

Návrh zadania schválený: áno / ~~nie~~[4]

Dňa: 12. 2. 2018

Podpis garanta predmetov

[3] 2 vedecké zdroje, každý v samostatnej rubrike a s údajmi zodpovedajúcimi bibliografickým odkazom podľa normy STN ISO 690, ktoré sa viažu k téme zadania a preukazujú výskumnú povahu problému a jeho aktuálnosť (uveďte všetky potrebné údaje na identifikáciu zdroja, pričom uprednostnite vedecké príspevky v časopisoch a medzinárodných konferenciách)

[4] Nehodiace sa prečiarknite

ČESTNÉ PREHLÁSENIE

Čestne vyhlasujem, že som túto prácu vypracoval(a) samostatne, na základe konzultácií a s použitím uvedenej literatúry.

V Bratislave, 30. apríla 2019 .....

Andrej Vítek

Anotácia

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Študijný program:	Inteligentné softvérové systémy
Autor:	Bc. Andrej Vítek
Diplomová práca:	Pravdepodobnostné prístupy pre analýzu vykonávanej úlohy na základe pohybu očí
Vedúci diplomovej práce:	Mgr. Jozef Tvarožek, PhD.
Apríl 2019	

Pohľad pozorovateľa na vizuálny stimul a jeho vzory sú ovplyvňované 3 zložkami. Prvá zložka (ovplyvnenie zdola nahor) súvisí s vizuálnym stimulom a jeho vlastnosťami. Druhá zložka (ovplyvnenie zhora nadol) súvisí s vlastnosťami pozorovateľa, ako je riešená úloha, skúsenosť, nálada alebo jeho pohlavie. Tretia zložka súvisí s pohybovými vlastnosťami očí. V dôsledku týchto zložiek obsahuje pohľad pozorovateľa v sebe veľké množstvo informácií o používateľovi ako aj o tom na čo sa pozerajú.

Modely, ktoré tento pohľad používajú, sú často založené na vyhodnocovaní agregovaných čŕt pohľadu, čím sa stráca informácia o postupnosti vizuálneho prehľadávania používateľa v čase. V rámci tejto práce sa zameriavame na vytvorenie nových čŕt pomocou použitia Skrytých Markovovych modelov, ktoré by dokázali abstrahovať časovú zložku z dát a skúmame, či pridanie týchto čŕt dokáže zlepšiť úspešnosť modelov, ktoré obsahujú len agregované črty.

Na vyhodnotenie našej metódy používame dátové sady pohybov očí z domény čítania štruktúrovaného textu a porovnávame klasifikátory, ktoré použili navrhnuté črty s tými, ktoré ich nepoužili. Na základe vyhodnotenia sme našli najväčšie zlepšenie (5-7%) u klasifikátora sklonom posilňovaných stromov.

Annotation

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

Degree course: Intelligent software systems

Author: Bc. Andrej Vítek
Master's Thesis: Probabilistic models for analysis of performed task based on eye movement
Supervisor: Dr. Jozef Tvarožek
2019, April

Gaze and visual patterns of observer while looking on visual stimuli are affected by 3 factors. First factor (bottom-up) is related to visual stimuli and its properties. Second factor (top-down) is related to personal characteristics of observer like task at hand, experience, mood or gender. Third factor is related to characteristics of oculomotor system. As a result of these factors, the observer's gaze contains a great amount of information about the user as well as about what they are looking at.

Models, which are using information about gaze, are often based only on evaluation of aggregated metrics of gaze, thereby losing temporal information about visual exploration of observer. In this work, we focus on the creation of features by using Hidden Markov models, which could partially abstract temporal information from data. We investigate, if adding these features to models which only use aggregated features can improve their results.

To evaluate our method, we used eye movement data from domain of structured text reading and compared classifiers which used designed features with other classfiers which did not use them. Based on evaluation, we have found greatest improvement (5-7%) with classifier gradient boosted trees.

Obsah

1 Úvod

Okolité prostredie obsahuje príliš veľa vizuálnych informácií na to, aby sme ich boli schopní všetky spracovať v tom istom momente. Z tohto dôvodu si človek vyvinul selektívnu pozornosť, kedy sa sústredí len na 1 miesto (fixácia) a jeho blízke okolie a rýchlymi pohybmi (sakády) sa premiestňuje medzi dôležitými miestami. Sekvenčná postupnosť fixácií a sakád sa nazýva pohľad alebo postupnosť pohľadu.

Pohľad a jeho vzory sú ovplyvňované rôznymi zložkami Kollmorgen et al. [2010], ktoré súvisia s vlastnosťami okolitého prostredia a jeho výraznosťou ako aj s vlastnosťami používateľa a vykonávanej úlohy. V článku Coutrot et al. [2017] je ukázané, že pravdepodobnostné modely (v článku je použitý Skrytý Markovov model) dokážu zohľadniť tieto zložky pri vytváraní modelu. Výhodou pravdepodobnostných modelov ako sú Markovove modely je, že sú to dátovo založené metódy a dokážu sa naučiť väčšinu interných parametrov na základe dát. Ďalšou dôležitou vlastnosťou je, že modelujú dáta ako časový rad namiesto agregovaných štatistík, čo lepšie reprezentuje pohľad Coutrot et al. [2017].

Na to aby sme boli schopný efektívne použiť určité algoritmy strojového učenia, ako rozhodovacie stromy alebo logistickú regresiu, je dôležité aby sme dáta z pohľadu používateľa predspracovali do čŕt. Použitím len agregovaných čŕt strácajú modely informáciu o postupnosti pohľadu v čase.

V tejto práci sa sústredíme na vytvorenie nových čŕt, ktoré by boli schopné abstrahovať časovú informáciu z postupnosti pohľadu pomocou pravdepodobnostných modelov. Našou hypotézou je, že pridaním čŕt, ktoré obsahujú temporálnu informáciu pohľadu do modelov, ktoré používajú na trénovanie len agregované črty, sme schopní zvýšiť ich úspešnosť. Ako skúmanú doménu sme sa rozhodli zvoliť doménu čítania zdrojového kódu, kvôli dostupnosti

dátových sád.

Články spomínané v podobných prácach používajú rôzne vstupné črty na trénovanie pravdepodobnostných modelov. Ďalším cieľom tejto práce je zistiť, aká kombinácia vstupných čŕt dokáže najlepšie reprezentovať dáta z vybranej domény a spraviť analýzu týchto čŕt.

V rámci tejto práce skúmam nasledovné výskumné otázky:

- Ktorý z vybraných pravdepodobnostných modelov a s akými parametrami a vlastnosťami dokáže najlepšie reprezentovať dáta z vybranej domény?
- Dokáže použitie pravdepodobnostných modelov významne zlepšiť predikciu vykonávanej úlohy, charakteristík používateľa alebo ďalších vlastností oproti modelom založených na vyhodnotení agregovaných údajov?

2 Analýza

V tejto kapitole sa zaoberám analýzou oblastí, ktoré súvisia s modelovaním pohľadu pomocou štandardných a pravdepodobnostných modelov a s čítaním štrukturovaného textu.

2.1 Pohľad

Keďže svet okolo človeka obsahuje priveľa informácií na spracovanie v jednom momente, tak človek si vyvinul spôsob spracovania na základe selektívnej pozornosti. Pohľad človeka spolu so spracovaním je sústredený len na miesto pohľadu a jeho blízke okolie Jacobs [1979]. Preto sú pohyby očí dôležitým prvkom pri spracovaní vizuálnych informácií.

Počas čítania textu, pozerania sa na scénu a iných činnostiach súvisiacich s pohľadom, vykonávajú naše oči postupnosť pohybov a zastavení. Pohyby sú nazývané *sakády* a slúžia na zmenu miesta pozornosti a sú charakteristické veľkou rýchlosťou (až 500° za sekundu). Kvôli vysokej rýchlosti nezískavame počas sakád nové informácie, pretože by sme videli namiesto informácie iba škvrnu Rayner [1998]. Prestávky medzi sakádami sa nazývajú *fixácie* a slúžia na získavanie informácií. Fixácie sú považované za dobu, kedy sú naše oči nehybné, alebo sa pohybujú len veľmi malé vzdialenosti okolo bodu pozornosti. Dĺžka fixácie závisí od vykonávanej úlohy a pri čítaní textu (angličtina) sa priemerná dĺžka fixácie pohybuje okolo 200-250 milisekúnd Rayner and Duffy [1986].

Existujú ešte aj iné pohyby očí, ktoré sú odlišné od štandardných sakád. Prvým z týchto pohybov sú prenasledovania (angl. *smooth pursuit*), ktoré nastávajú, keď pomocou pohybov očí prenasledujeme pohybujúci sa cieľ. Druhým pohybom je konvergencia očí (angl. *vergence*), ktorý nastáva, keď sa sústredíme na objekt, ktorý je blízko pri nás. Posledným sú vestibulárne

pohyby očí, ktoré súvisia s otáčaním očí, ako kompenzácia za pohyb hlavy alebo tela Rayner [1998].

Počas štandardných pohybov nastávajú aj menšie pohyby, ktoré sú nazývané mikropohyby. Medzi mikropohyby očí patria napríklad mikrosakády a nystagmus. Nystagmus je veľmi malý ale konštantný pohyb očí, ktorý sa vyskytuje aj počas fixácií. Mikrosakády slúžia na vrátenie očí na pôvodné miesto v prípade posunutia kvôli nedokonalému ovládaniu očí Rayner [1998].

Správanie pohľadu a jeho vzory sú ovplyvňované 3-mi rôznymi zložkami Kollmorgen et al. [2010], Le Meur and Liu [2015]:

- Zdola nahor (angl. *bottom-up*) - súvisí s vlastnosťami vizuálneho stimulu ako je pohyb, farba, osvetlenie, rozloženie v priestore a či obsahuje tváre alebo iné objekty. Výraznosť vizuálneho stimulu sa modeluje pomocou mapy výraznosti (angl. *saliency map*).

- Zhora nadol (angl. *top-down*) - je prepojená s používateľom. Závisí od používateľa, jeho zdravotného stavu, kultúry, predchádzajúcich skúseností s vizuálnym stimulom, motivácie a ďalších personálnych a kognitívnych vlastností a taktiež od vykonávanej úlohy a jej vlastností.

- Priestorové ovplyvnenie (angl. *spatial viewing biases*) - je nezávislé od vykonávanej úlohy a vizualného stimulu. Patrí sem napríklad preferencia ľavej strany na začiatku skúmania Ossandon et al. [2014].

Postupnosť pohľadu obsahuje množstvo informácií o vykonávanej úlohe Yarbus [1967], danej doméne a o charakteristikách používateľa Coutrot et al. [2016]. Existuje veľa teórií a modelov, ktoré sa snažia modelovať pohľad používateľa a vlastnosti scény na ktorú sa pozerá Itti et al. [1998]. Prístupy v tejto oblasti trpia viacerými problémami, ako je použitie len jednoduchých reprezentácií ako výraznosť vizuálnej informácie a tiež málo modelov zohľadňuje časti zložky zhora nadol ako je vykonávaná úloha, personálne a kognitívne charakteristiky subjektu, alebo pohlavie, ktoré tiež ovplyvňujú vlastnosti pohľadu Coutrot et al. [2016].

2.2 Čítanie prirodzeného textu

Počas čítania prirodzeného textu sa oči pohybujú približne 4 krát za sekundu s priemernou dobou fixácie okolo 200 až 250 milisekúnd a dĺžkou sakád okolo 7 až 9 písmen (približne rovnaké množstvo písmen aj pre rôzne vzdialenosti čitateľa od textu), ktorých cieľom je dostať novú časť textu do foveálneho videnia (oblasť, na ktorú sa priamo pozeráme a dokážeme najlepšie vnímať) Rayner [1998].

Samotná dĺžka fixácie sa pohybuje od 100 po 500 milisekúnd v závislosti od rôznych vlastností slov, ako je ich dĺžka. Ak je slovo ľahko predikovateľné z okolitého kontextu, tak fixácia naň trvá kratší čas ako na slová, ktoré nie sú predikovateľné. Podobnú vlastnosť majú aj slová, ktoré sú v jazyku menej často používané oproti často používaným a taktiež slová, ktoré majú nesamozrejmý význam z daného kontextu alebo nie sú ľahko pochopiteľné (napríklad kvôli jazykovým chybám v slove) Rayner and Duffy [1986].

Väčšina sakád sa pri čítaní prirodzeného textu pohybuje zľava doprava a postupne sleduje riadky textu (platí pre jazyky s čítaním zľava doprava). Okolo 10% až 15% sakád sú regresie - sakády, ktoré sa vracajú opačným smerom na predchádzajúce slovo v texte. Kratšie regresie v rámci jedného slova indikujú problémy so spracovaním aktuálneho slova. Dlhšie regresie (viac než 10 znakov) indikujú problémy s pochopením časti textu. Niektoré kratšie regresie, dlhé len zopár písmen, vznikajú z dôsledku príliš dlhej sakády a následného vrátenia, aby oči mohli pokračovať v spracovaní textu Rayner [1998].

Rýchlosť čítania prirodzeného textu závisí od čitateľa a od cieľov, ktoré chcú čítaním dosiahnuť. V knihe Carver [1990] sú rýchlosti čítania rozdelené do viacerých kategórií v závislosti od metriky *slov za minútu* (angl. *WPM - word per minute*) pre angličtinu:

- Skenovanie (angl. *scanning*, rýchlosť okolo 600 slov za minútu) - je zvyčajne použité, ak čitateľ vyhľadáva konkrétne slovo alebo frázu v

texte.

- Kĺzanie (angl. *skimming*, rýchlosť okolo 450 slov za minútu) - nastáva, ak čitateľ chce rýchlo získať základný prehľad o čítanom texte.
- Čítanie (angl. *Rauding*, rýchlosť okolo 300 slov za minútu) - predstavuje normálne čítanie, kde čitateľ postupne prechádza slová v texte.
- Učenie (angl. *learning*, rýchlosť okolo 200 slov za minútu) - pri nadobúdaní znalostí z textu.
- Memorovanie (angl. *Memorizing*, rýchlosť okolo 130 slov za minútu) - pri dôslednom čítaní a postupnom testovaní čitateľom s cieľom zapamätať si text.

Čitatelia medzi týmito procesmi preskakujú v závislosti od aktuálnej úlohy.

2.3 Čítanie štruktúrovaného textu

Štruktúrovaný text sa odlišuje od prirodzeného textu vo viacerých ohľadoch. Na rozdiel od prirodzeného textu je často štruktúrovaný text vyjadrený pomocou 2 jazykov:

- Kľúčové slová - slúžia na ohraničovanie a určenie štruktúry alebo na zabezpečenie špecifickej funkcionality v prípade kódu. Klúčových slov býva väčšinou len veľmi malé množstvo oproti počtu slov v prirodzených jazykoch. Sú predpísané daným štruktúrovaným jazykom.
- Slová z prirodzeného jazyka - sú použité v rámci štruktúrovaného jazyka na rôzne účely ako sú hodnoty alebo identifikátory.

Ďalším rozdielom oproti prirodzenému textu je špecifická štruktúra. Štrukturované texty majú na základe použitých klúčových slov presne definovanú štruktúru a často taktiež obsahujú rôzne druhy horizontálneho aj vertikálneho odsadenia na zabezpečenie prehľadnosti alebo ako časť štruktúry (v programovacom jazyku Python sa odsadenia používajú na ohraničenie blokov vykonávania) Busjahn et al. [2015].

2.3.1 Čítanie zdrojového kódu

Zdrojový kód sa od prirodzeného jazyka líši ešte aj na sémantickej úrovni. Na pochopenie prirodzeného jazyka je potrebné byť schopný prečítať text (znaky) a na základe jeho domény a kontextu (význam) mu porozumieť. Kód je rozšírený ešte aj o ďalšiu úroveň, ktorú čitateľ musí pochopiť a ňou je jeho vykonanie (angl. *execution*) Busjahn et al. [2015]. Vykonanie je určené postupnosťou volania príkazov spolu so stavom programu počas jeho vykonávania.

Články, ktoré sa zaoberajú analýzou pohľadu pri čítaní zdrojového kódu, častokrát riešia jednu alebo viacero úloh súvisiacich s prácou so zdrojovým kódom. Medzi časté úlohy patria:

- hľadanie chýb v kóde alebo jeho debugovanie Bednarik [2012], Turner et al. [2014].
- pochopenie cudzieho zdrojového kódu Turner et al. [2014] - táto úloha býva rozšírená o zistenie správneho výstupu pre daný zdrojový kód alebo v niektorých článkoch o test, kde treba doplniť chýbajúce slovo na vyznačené miesto (angl. *Cloze test*) Crosby and Stelovsky [1990].

Viacero článkov v tejto oblasti analyzuje rozdiely v pohľade začiatočníka v programovaní od experta Wiedenbeck et al. [1993], Busjahn et al. [2015]. V článku Bednarik [2012] sa zameriava nad nájdením rozdielov medzi začiatočníkmi a expertmi pri debugovaní kódu - používatelia sa mohli pozerať na samotný kód, jeho výstup alebo vizualizáciu programu, ktoré boli na rovnakej obrazovke. Na základe štúdie zistil, že experti trávia pri debugovaní viacej času pozeraním sa na kód a jeho výstup ako začiatočníci, ktorí strávili viacej času pozeraním sa na vizualizáciu. V článku Hansen et al. [2013] bolo zistené, že väčšia skúsenosť môže v určitých prípadoch byť aj záporná, najmä pri neštandardných situáciach. Príkladom môže byť metóda na vykonanie operácie, ktorá nevracia hodnotu.

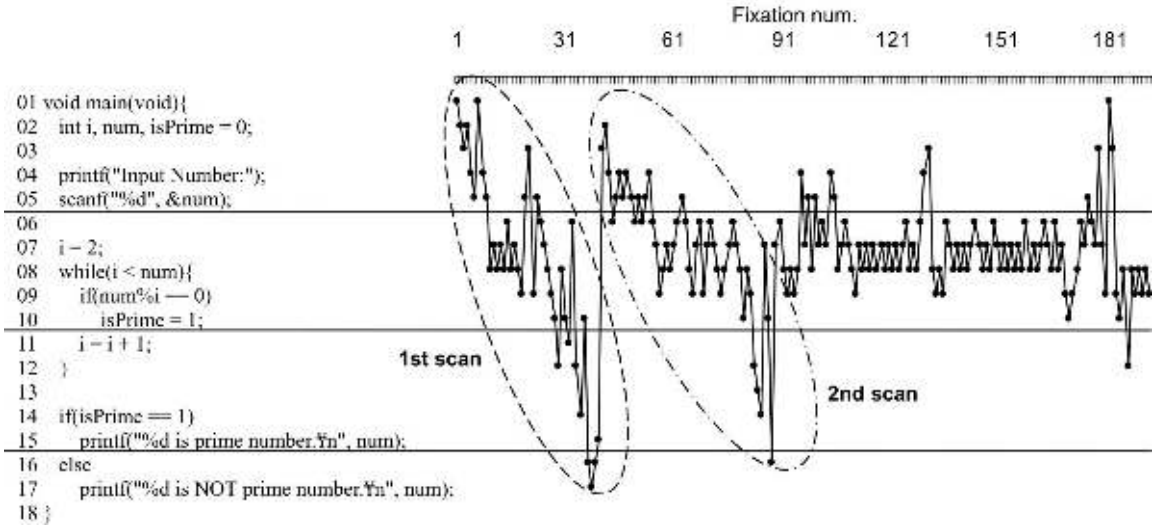
Kým pri čítaní prirodzeného jazyka je len 10% až 15% zo sakád zároveň aj regresiou, pri čítaní zdrojového kódu je ich oveľa viac (33% až 41%) Crosby

and Stelovsky [1990], Busjahn et al. [2011]. Taktiež pri čítaní zdrojového kódu je priemerná dĺžka fixácie dlhšia (300 - 400 milisekúnd oproti 200 až 250 milisekúnd pri čítaní prirodzeného textu v angličtine) Busjahn et al. [2011]. Toto naznačuje, že komplexita pri čítaní zdrojového kódu vzniká hlavne z pochopenia ako program funguje a interakcii v ňom, než zo samotného čítania Wiedenbeck et al. [1993].

Aby bol zdrojový kód zrozumiteľný nie len pre počítač, ktorý ho vykonáva ale aj pre programátora, ktorý ho číta a píše, tak má častokrát špecifickú formu podobnú prirodzenému textu (kľúčové slová bývajú vyjadrené pomocou slov z anglického jazyka, ktoré opisujú funkciu daného kľúčového slova). V článku Busjahn et al. [2015] testujú, či pohľad programátora pri čítaní zdrojového kódu má lineárny vzor alebo vzor postupnosti vykonania (angl. *execution order*). Lineárna postupnosť čítania reprezentuje čítanie podobné prirodzenému textu, kde čitateľ postupne sleduje text riadok po riadku zľava doprava. Postupnosť vykonávania je určená tým, ako by cez zdrojový kód prechádzal počítač. Ďalším cieľom bolo zistiť, či je niektorý z týchto vzorov čítania viac preferovaný pri porovnaní začínajúcich programátorov s expertmi. Na základe štúdie bolo zistené, že model postupnosti vykonávania programu lepšie vysvetľuje pohľad experta ako začiatočníka.

Pri čítaní zdrojového kódu existujú rôzne stratégie a vzory, ktoré sa vyskytujú medzi programátormi. V rámci článku Uwano et al. [2006] bolo zistené, že 72.8% riadkov kódu je prejdených v prvých 30% času. Tento vzor bol nazvaný *Scan*, ktorého pravdepodobnou úlohou je pochopenie celkovej programovej štruktúry a identifikácií dôležitých častí. Tento vzor je vidieť na obrázku 2.1. Medzi ďalšie nájdené vzory patria *Vrátenie sa k deklarácii* (angl. *Retrace Declaration*) a *Vrátenie sa k referencii* (angl. *Retrace Reference*), ktoré súvisia s vrátením sa k predchádzajúcemu výskytu premennej, na ktorú sa aktuálne čitateľ pozerá.

Dĺžka fixácií pri čítaní zdrojového kódu súvisí s druhom elementu, na ktorý sa čitateľ aktuálne pozerá. V článku Busjahn et al. [2014] je rozoberaný



Obr. 2.1: *Ukážka vzoru Scan pri čítaní zdrojového kódu (obrázok z Uwano et al. [2006]).*

vplyv rôznych druhov elementov nachádzajúcich sa v zdrojovom kóde na dĺžku zostatia na ňom (angl. *dwell time*, od dĺžky fixácie sa líši v tom, že vyjadruje celkový čas zostatia naprieč všetkými fixáciami na element). Elementy sú závislé od programovacieho jazyka, pričom v článku sú skúmané nasledovné elementy z jazyka Java:

- Identifikátory - postupnosti písmen a číslic, ktoré slúžia na pomenovanie premenných, funkcií, tried a iných konštrukcií v jazyku.
- Kľúčové slová - predpísané slová, ktoré nemôžu byť použité ako identifikátory a slúžia na zabezpečenie hlavnej funkcionality v jazyku.
- Literály - kódom reprezentované hodnoty. Príkladom môžu byt číslice, boolovské hodnoty alebo reťazce znakov.
- Separátory - patria sem zátvorky ((), , []) a interpunkcia (., „ ;).
- Operátory - slúžia na priraďovanie hodnôt a na vykonávanie operácií nad 1 alebo viacerými elementami. Zvyčajne obsahujú 1 až 2 znaky ako = alebo ++.

Štatisticky významný rozdiel v dĺžke zostatia mali iba separátory, ktoré mali významne nižší čas zostatia oproti ostatným druhom elementov aj po normalizácii na základe počtu písmen v slove (podobne ako pri čítaní prirodzeného textu má dĺžka slov vplyv na dĺžku fixácie).

2.4 Spracovanie pohľadu

Na zachytenie miesta, kam sa používateľ pozerá pri vykonávaní úlohy na počítačovej obrazovke a ďalších vlastností ako je velkosť zreničky a vzdialenosť hlavy používateľa sa používa zariadenie na sledovanie pohľadu (angl. *eye-tracker*), ktoré sa štandardne nachádza pri displeji alebo je umiestnené pri alebo na hlave používateľa. Kým v minulosti musela byť hlava používateľa fixovaná na jednom mieste, alebo zariadenie namontované na hlave používateľa, teraz je možné, aby sa používateľ do určitej miery pohyboval voľne a taktiež tieto zariadenia dosahujú dobrú presnosť (do 1 stupňa odchýlky a lepšie) Böhme et al. [2006]. Na to, aby zariadenie vedelo presne rozoznať, na ktoré miesto sa používateľ pozerá, používa väčšina zariadení určitý spôsob kalibrácie (štandardne ide o 9 bodovú kalibráciu, kde sa používateľ pozrie do stredu a na 8 bodov na kraji obrazovky). Použitie podobného postupu ako pri kalibrácií má zmysel taktiež aj na validáciu presnosti nameraných dát alebo ich opravu.

Zariadenia na snímanie pohľadu majú väčšinou určitú snímaciu frekvenciu, s ktorou zaznamenávajú vlastnosti oka a pozíciu miesta pohľadu na obrazovke. Týmto snímaním vznikajú surové dáta pohľadu. Na získanie postupnosti pohľadu používateľa (angl. *scanpath*), ktorá sa skladá z fixácií a sakád, sa používa fixačný filter. Metódy na základe ktorých fixačné filtre fungujú, sa rozdeľujú do 2 kategórií Kasneci et al. [2014]:

- Metódy na základe hranice (angl. *threshold-based*): rozdelenie surových dát na fixácie a sakády je založené na prekročení hranice atribútov ako je rozptyl, rýchlosť alebo zrýchlenie.
- Pravdepodobnostné metódy: rozdeľujú surové dáta na fixácie a sakády

na základe pravdepodobnosti, ktorá je prispôsobená pozorovaným dátam.

Pri niektorých úlohách, ktoré sú opísané v časti 2.4.1, hrajú dôležitú úlohu oblasti záujmu (angl. *Areas of interest = AOI*). Oblasti záujmu sú dôležité alebo skúmané miesta na obrazovke alebo v rozhraní, na ktoré sa používateľ pozerá. Tieto miesta sú častokrát vytvárané automaticky (vytvorenie mriežky s rovnako veľkými oblasťami) alebo manuálne na základe obsahu. Oblasti záujmu hrajú dôležitú rolu pri niektorých algoritmoch na porovnanie postupnosti pohľadu (napr. metóda porovnania editačných vzdialeností) alebo na agregáciu fixácií a vytvorenie ďalších čŕt Le Meur and Baccino [2013].

2.4.1 Štandardné úlohy

Vplyv na pohľad má aj vykonávaná úloha, ako bolo ukázané v článku od Yarbus [1967] kde na základe vykonávanej úlohy boli sústredené iné časti obrazu a v inej postupnosti, čím ukázal, že vykonávaná úloha a kognitívne procesy používateľa majú vplyv na pozornosť.

Úlohy, ktoré participanti vykonávajú závisia od cieľov štúdie ale taktiež od druhu vizuálneho stimulu, na ktorý sa budú participanti štúdie pozerať. Medzi často používané druhy vizuálnych stimulov patria:

- Obrázok - ide o grafický prvok, ktorý môže obsahovať viacero objektov. Dôležitý rozdiel pri pozeraní na obrázok môže spôsobiť to, či sa na ňom nachádzajú ľudia a ich tváre Le Meur and Baccino [2013].
- Text - môže sa líšiť v tom, či ide o prirodzený jazyk alebo umelý a taktiež akým spôsobom je text štrukturovaný (odseky, odrážky, riadkovanie, ...). Určité druhy textu ako zdrojový kód majú špecifickú štruktúru 2.3.1.
- Dynamická scéna alebo video - ide o vizuálny stimul, ktorý sa mení postupom času. Môže ísť o video na obrazovke, alebo o samotný pohľad participanta v priestore s nakrúcaním prostredia (napr. prechádzka v parku) počas nejakej špecifickej úlohy Kit and Sullivan [2016].

- Rozhranie - je špecificky štrukturovaný prvok, ktorý sa môže skladať z rôznych iných častí a umožňuje určité interakcie. Interakcie s rozhraním môžu zmeniť jeho obsah a rozmiestnenie.

Pri pozeraní sa na obrázok patria medzi časté úlohy voľné pozeranie (angl. *free-viewing*) Le Meur and Liu [2015], nájdenie objektu (angl. *find target*) Coutrot et al. [2017] alebo zapamätanie si objektov na obrázku Chuk et al. [2014] a ich obmeny na základe cieľov štúdie. Obrázky sa často používajú na vyhodnotenie nových metód a ich schopnosti sa naučiť rozlišovať vykonávanú úlohu Le Meur and Baccino [2013], Anderson et al. [2015]. Podobné úlohy sú taktiež používané pri pozeraní sa na video.

Medzi časté úlohy pri pozeraní sa na text patrí voľné čítanie, hľadanie slova alebo odpovede na otázku a pochopenie textu Simola et al. [2008]. Pri čítaní zdrojového kódu je pochopenie textu rozšírené o pochopenie vykonávania programu a časté úlohy súvisia s hľadaním chýb v kóde alebo jeho pochopením a na jeho základe zistiť, aký bude výpis programu 2.3.1.

Pri rozhraní hrá dôležitú rolu typ rozhrania, na ktorý sa používateľ pozerá. V prípade vývojového prostredia (angl. *integrated development environment*) sú riešené podobné úlohy, ako len pri pozeraní na zdrojový kód Bednarik [2012]. Často používaným rozhraním je prehliadač s internetovými stránkami, kde bývajú testované úlohy založené na navigácii, vyhľadávaní a testovaní zručnosti používateľa Ehmke and Wilson [2007].

2.5 Druhy čŕt pre pohľad

Na modelovanie pohľadu sa používajú rôzne druhy čŕt, ktoré súvisia s určitými algoritmami a dajú sa rozdeliť do týchto kategórií Coutrot et al. [2017]:

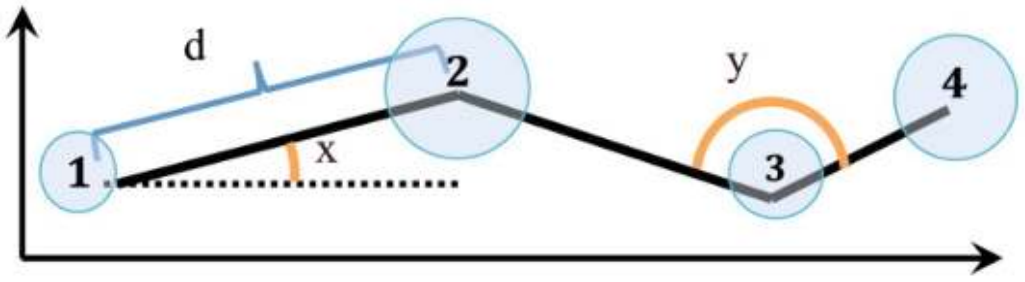
- Základné parametre pohľadu a agregácie - sem patria hlavné parametre pohľadu a ich agregácie vytvorené cez štatistické charakteristiky ako *suma*, *priemer* alebo *rozptyl*.

- Priestorové rozdelenia pozícií očí - súvisí s priestorovými rozdeleniami vytvorenými pomocou základných parametrov. Medzi známe priestorové rozdelenia patrí tepelná mapa (angl. *heatmap*) alebo mapa výraznosti Itti et al. [1998].

- Reťazcové a geometrické porovnávania - črty, ktoré súvisia s porovnávaním 2 postupností pohľadu.

- Pravdepodobnostné prístupy - medzi známe črty patrí *matica prechodov* (angl. *transition matrix*), ktorá vzniká napr. pri Markovovych procesoch.

Prvá kategória sa zaoberá hlavnými parametrami pohľadu a ich agregáciami. Pri fixáciách patria medzi používané črty *dĺžka fixácie* a *priemerná dĺžka fixácie*, *celkový počet fixácií* a ich štatistické vlastnosti ako je ich rozptyl. Pri sakádach sa často používajú črty súvisiace s ich dĺžkou, rýchlosťou a smerom Kardan et al. [2015]. Medzi ďalšie zaujímave črty patrí veľkosť zreničky, ktorá súvisí s kognitívnou zátažou používateľa Klingner et al. [2008]. Taktiež sem patrí množstvo a dĺžka žmurknutí a vzdialenosť hlavy od displeja a jej pozícia, ktorá z časti súvisí s postojom. Črty *dĺžka sakády* (d na obrázku 2.2), *relatívny uhol sakády* (y na obrázku 2.2) a *absolútny uhol sakády* (x na obrázku 2.2) umožňujú modelovať trendy v pohľade používateľa ako je špecifická postupnosť pohľadu Steichen et al. [2014].



Obr. 2.2: *Ukážka fixácií, sakád a niektorých ich vlastností (obrázok z Steichen et al. [2014]).*

Medzi rôzne kategórie patria črty vytvorené z oblastí záujmu, ktoré majú dôležitú úlohu pri porovnávaní postupností pohľadu (angl. *scanpath*) ale

taktiež ako agregácie základných parametrov pohľadu. Pri reťazcových porovnávaniach slúžia na transformáciu postupnosti pohľadu do reťazca, kde sú jednotlivé oblasti záujmu označené písmenom, ktoré je priradené všetkým obsiahnutým fixáciám Kübler et al. [2017], Anderson et al. [2015]. Oblasti záujmu sa taktiež dajú použiť na tvorbu agregovaných čŕt ako je *počet fixácií v AOI*, *strávený čas v AOI* a ďalšie črty súvisiace s prvou fixáciou na oblasť záujmu a jej vlastnosťami ako je doba strávenia. Pri porovnávaní času strávenom v rôznych oblastiach záujmu medzi inými používateľmi má zmysel normalizovať čas na percentuálne strávenú dĺžku. Táto normalizácia vyvažuje používateľov, ktorí majú viacero fixácií alebo strávia na oblastiach záujmu dlhší čas Jbara and Feitelson [2017].

Dôležitú úlohu má pri pohľade aj vizuálny stimul, na ktorý sa používateľ pozerá, ktorý súvisí s ovplyvnením zdola nahor Kollmorgen et al. [2010]. Pri dynamickom vizuálnom stimule hrá dôležitú vlastnosť pohyb a všeobecne má na pohľad vplyv farba, osvetlenie, rozloženie v priestore. Tieto vlastnosti sú modelované na základe mapy výraznosti Itti et al. [1998].

2.5.1 Črty pri čítaní

Pri čítaní prirodzeného textu môže mať zmysel modelovať *počet fixácií*, *priemernú dĺžku sakád* a *počet regresií* v určitých častiach textu. Väčší počet fixácií, dlhšie sakády a väčšie množstvo regresií môže indikovať problémy pri spracovaní alebo pochopení textu Rayner [1998]. V článku Simola et al. [2008] používajú črty súvisiace s trvaním fixácií, dĺžkou a smerom sakád na určovanie kognitívnych stavov čitateľa.

Pri čítaní zdrojového kódu môže mať vplyv na vlastnosti pohľadu druh elementu, na ktorý sa čitateľ pozerá Busjahn et al. [2014], Crosby and Stelovsky [1990]. Tento problém sa dá riešiť pomocou anotovania fixácií na základe toho, na ktorý element v kóde dopadá (môže byť problém namapovať fixáciu na časť kódu, ak je zariadenie na sledovanie pohľadu nepresné, alebo fixácie sú príliš veľké oproti zdrojovému kódu) Orlov [2015]. Po tomto spracovaní

sa dajú druhy elementov spracovať a vytvoriť črty podobné, ako pre oblasti záujmu (oblasť záujmu bude tvorená skupinou elementov rovnakého druhu).

2.6 Štandardné metódy na vyhodnotenie a modelovanie pohľadu

Vhodnosť použitej metódy na spracovanie a vyhodnotenie pohľadu závisí od riešenej úlohy a cieľov daného článku. Pri vyhodnocovaní závisí od toho, či sú vyhodnocované dáta alebo vytvorený model nad dátami. Na vyhodnotenie dát sa používajú neparametrické Turner et al. [2014] a parametrické Crosby and Stelovsky [1990], Rayner and Duffy [1986] štatistické testy (častým príkladom je test ANOVA), ktorých použitie závisí od vlastností dát. Spôsoby vyhodnocovania modelov ako je výpočet metrík vyhodnotenia a porovnávanie sú opísane v časti 2.8.

Pri modelovaní pohľadu pomocou strojového učenia sa používajú rôzne druhy modelov v závislosti od druhu úlohy. Medzi časté úlohy patrí:

- Regresná analýza - slúži na modelovanie závislej premennej (skúmaná vlastnosť) na základe vzťahov s inými nezávislými premennými. Regresná analýza vysvetľuje ako sa závislá premenná mení pri zmenách nezávislých premenných. Medzi časté druhy regresnej analýzy patrí lineárna regresia Hansen et al. [2013]. Príklad článku skúmajúceho pohľad používateľa v ktorom bola použitá regresná analýza je Kollmorgen et al. [2010].

- Klasifikácia - zaoberá sa identifikáciou, do ktorej triedy z vopred stanovených tried prislúcha daná dátová vzorka na základe iných nezávislých premenných. Triedy sú stanovené na základe riešenej úlohy Steichen et al. [2014], Hansen et al. [2013]. Ukážku klasifikácie na základe pohľadu je vidieť v článku Coutrot et al. [2017].

- Zhlukovanie - slúži na zoskupenie dát do skupín na základe podobnosti vlastností. Použitie zhlukovania na analýzu pohľadu je vidieť napríklad

v článku Coutrot et al. [2016].

V strojovom učení patrí regresná analýza a klasifikácia do skupiny metód *učenia s učiteľom* (angl. *supervised learning*). Metódy z tejto skupiny potrebujú na naučenie a testovanie dáta, ktoré už majú závislú premennú určenú (hodnota pre regresiu alebo trieda pre klasifikáciu). Táto hodnota by mala predstavovať preukázateľnú pravdu (angl. *ground truth*) a jej nepresnosti korelujú s nepresnosťami pri vyhodnocovaní. Zhlukovanie patrí do skupiny metód *učenia bez učiteľa*, kde nie je potrebné dopredu vedieť správnu triedu, ale ak ju poznáme, tak môže slúžiť na vyhodnotenie úspešnosti zhlukovania.

V niektorých špecifických úlohách nemá zmysel brať do úvahy celú postupnosť pohľadu (napríklad priebežná klasifikácia počas vykonávania úlohy). V týchto prípadoch sa nad dátami používa pohyblivé okno (angl. *moving window*), ktoré určuje do akej vzdialenosti od aktuálnej pozície sa majú brať vzorky na strojové učenie. Vzdialenosť môže byť určená rôznymi metrikami ako je časová vzdialenosť Kasneci et al. [2014] alebo počet fixácií.

Medzi často používané modely pri modelovaní pohľadu a riešení úloh súvisiacich s klasifikáciou patrí *logistická regresia* (angl. *logistic regression*), algoritmy na základe *rozhodovacích stromov* (angl. *decision trees*), *podporný vektorový stroj* (angl. *support vector machine = SVM*) a neurónové siete Steichen et al. [2014, 2013], Lagun et al. [2011].

2.6.1 Porovnávanie postupností pohľadu

Keď používame na modelovanie len agregačné črty pohľadu ako je počet fixácií alebo ich dĺžka, tak nevyužívame dôležité informácie o pohľade, ktoré súvisia s jeho pohybom v priebehu času. Metódy založené na porovnávaní postupnosti pohľadu sa sústredia na porovnávanie tejto postupnosti fixácií, sakád a ich vlastností. Jednou z dôležitých vlastností je spôsob ako sú mapované a porovnávané postupnosti medzi sebou (spôsob kvantizácie), na základe ktorej sa dajú rozdeliť metódy do nasledovných kategórií Anderson et al. [2015]:

- Mriežka - fixácie sú mapované do mriežky, ktorá slúži na transformáciu postupnosti pohľadu na reťazec znakov.
- Dosah (angl. *radius*) - porovnanie, či 2 fixácie sú považované za rovnaké, je na základe prahovej vzdialenosti.
- Priame - na porovnávanie sa používajú skutočné súradnice fixácií.

Metódy, ktoré sú založené na mapovanie fixácií do mriežky, fungujú na princípe porovnávania reťazcov na zistenie, ako sú postupnosti podobné. Reprezentantom tejto skupiny je porovnávanie na základe editačnej vzdialenosti, kde sa počíta počet znakov, ktoré treba pridať, nahradiť alebo zmazať, aby boli reťazce rovnaké Le Meur and Baccino [2013]. Tieto prístupy majú ale viacero problémov. Presnosť porovnania závisí od mriežky a jej manuálne vytváranie je často časovo náročné. Fixácie, ktoré sú veľmi blízko seba, možu byť rozdielne označené ak sú na okraji čiary mriežky Anderson et al. [2015].

2.7 Pravdepodobnostné metódy na modelovanie pohľadu

V rámci pravdepodobnostných prístupov existuje viacero modelov, ale nie všetky sú vhodné na riešenie úloh súvisiacich so spracovaním pohľadu. Počas analýzy stavu problematiky sme identifikovali 2 hlavné skupiny modelov, ktoré sa používajú na modelovanie pohľadu používateľa. Ktorý model je vhodnejší na použitie má veľkú závislosť od riešenej úlohy.

2.7.1 Bayesovský zmiešaný model (angl. *Bayesian mixture model*)

Bayesovský zmiešaný model je pravdepodobnostný model, ktorý vzniká zlúčením N náhodných premenných a ich rozdelení do jedného modelu. Model obsahuje aj ďalšie parametre, ktoré slúžia na opis vlastností jednotlivých náhodných premenných a funkcie, ktorá závisí od toho, aké majú tieto náhodné premenné rozdelenie. Častou verziou tohto modelu je Gaussovský zmiešaný model (angl. *Gaussian mixture model*), ktorého náhodné premenné majú

normálne rozdelenie a pre každú náhodnú premennú obsahuje parametre strednej hodnoty a odchýlky.

2.7.1.1 Parametre a ich nastavenie

Dôležitou vlastnosťou Bayesovských zmiešaných modelov je počet náhodných premenných, ich parametre a do akého rozdelenia patria náhodné premenné. Hlavnou výhodou Bayesovkého zmiešaného modelu a modelov od neho odvodených je taký, že parametre náhodných premenných sú natrénovateľné z dát na základe algoritmov maximalizácie očakávaní (angl. *Expectation-maximalization algorithm*) Coutrot et al. [2017].

Počet náhodných premenných sa dá tiež určiť automaticky na základe vstupných dát pomocou kritéria minimálnej dĺžky správ (angl. *minimum message length criterion*), ktorý sa zbaví náhodnej premennej v prípade, keď nie je dostatok dát, ktoré by ju podporili. Ďalším spôsobom výberu modelu s optimálnym počtom náhodných premenných je použitie Bayesovského informačného kritéria (angl. *Bayesian information criterion*), ktorý penalizuje modely, ktoré obsahujú príliš veľa parametrov Coutrot et al. [2017].

Funkcia rozdelenia náhodných premenných sa dá určiť na základe modelovanej domény a dát. Tu sú príklady niektorých vhodných rozdelení na základe modelovaných dát:

- **Bernoulliho rozdelenie** - keď modelujeme kategorické alebo diskrétne hodnoty
- **Normálne rozdelenie** - vhodné na spojité dáta ako je postupnosť pohľadu
- **Logaritmicko-normálne rozdelenie** - použiteľné na hodnoty, ktoré sú kladné a majú tendenciu rásť exponenciálne

V závislosti od druhu dát môže rozdelenie taktiež byť viac-dimenzionálne. Príkladom môže byť modelovanie dát z postupnosti pohľadu používateľa,

kde sa súradnica fixácie skladá z dvoj-dimenzionálnej hodnoty Kasneci et al. [2014].

2.7.2 Skrytý Markovov model (angl. *Hidden Markov Model*)

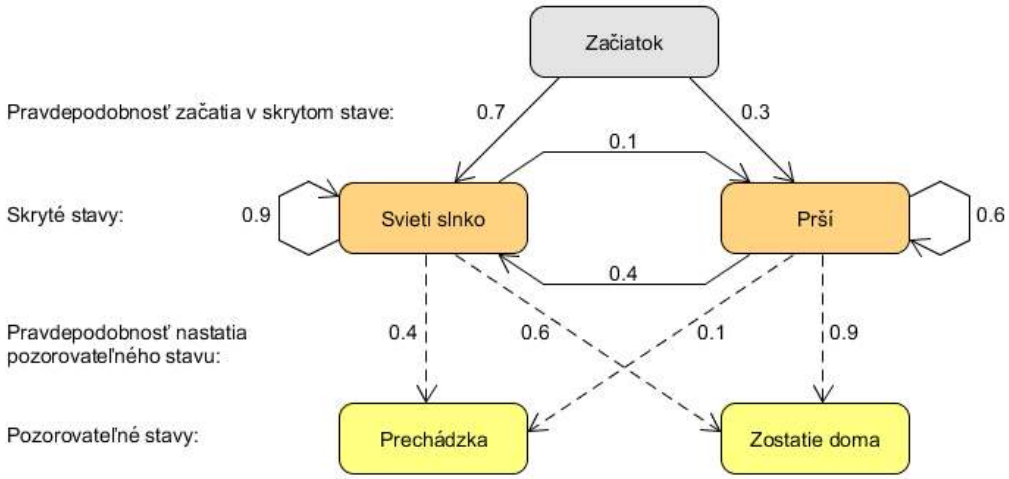
Ide o pravdepodobnostný model, ktorý sa skladá zo skrytých a pozorovateľných stavov. Skryté stavy sú nepoznané a dajú sa určiť len na základe pravdepodobnosti. Poznaná je len hodnota pozorovateľných stavov, ktoré sú reprezentované vstupnými dátami. Skrytý Markovov model obsahuje nasledovné parametre:

- Počet skrytých stavov
- Pravdepodobnosť začatia v skrytom stave (angl. *Priors*)
- Matica prechodov medzi skrytými stavmi (angl. *Transition matrix*)
- Pravdepodobnosť nastatia pozorovateľného stavu zo skrytého stavu (angl. *Emissions*)

Skrytý Markovov model môže obsahovať aj ďalšie parametre v závislosti od reprezentácie skrytých stavov a druhu Markovovho modelu. Vhodné je ho použiť, ak sa štatistické vlastnosti modelovaného radu časom menia. Skrytý Markovov model tieto zmeny vysvetľuje pomocou zmeny skrytého stavu. Dôležitou vlastnosťou všetkých Markovovych modelov je Markovova vlastnosť (angl. *Markov property*), ktorá hovorí, že nasledujúci stav (v tomto prípade skrytý stav) je určený len predchádzajúcim stavom, čo znamená, že sa dá vyjadriť vzorcom P(St | St-1), kde S reprezentuje skrytý stav a t čas. Príklad Skrytého Markovovho modelu je vidieť na obrázku 2.3.

2.7.2.1 Parametre a ich nastavenie

Pravdepodobnosť začatia v skrytom stave, matica prechodov a pravdepodobnosť nastatia pozorovateľých stavov sa dajú zistiť pre konkrétnu úlohu a dáta pomocou algoritmov maximalizácie očakávaní ako je Baum-Welchov algoritmus Bacciu et al. [2015].



Obr. 2.3: *Príklad Skrytého Markovovho modelu aj s opísanými parametrami.*

Parameter počtu skrytých stavov je dôležitý, keďže pri príliš veľkom počte skrytých stavov sa model ľahšie pretrénuje, ale pri príliš malom počte nemusí mať dostatok stavov, aby sa správne naučil vlastnosti dát. Pomocou variačného Bayesovského odvodenia (angl. *Variational Bayesian Inference*), ktorý odhadne parametre modelu a jeho zložitosť, sa dá počet skrytých stavov určiť automaticky na základe vstupných dát McGrory and Titterington [2009].

Dôležitou vlastnosťou je aj reprezentácia skrytého stavu, ktorá má podobnú úlohu ako rozloženia náhodných premenných v časti 2.7.1.1. Reprezentácia je závislá od vstupných dát a riešenej úlohy a vyjadruje akým spôsobom budú modelované pravdepodobnosti nastatia pozorovateľného stavu zo skrytého stavu. Príkladom je reprezentácia skrytého stavu dvoj-dimenzionálnym normálnym rozdelením, ktorý je určený jeho strednou hodnotou a odchýlkou a dá sa použiť na reprezentáciu pozície fixácie na obrazovke (príklad na obrázku 2.4).

2.7.2.2 Použitie a algoritmy

Skryté Markovove modely a prístupy od nich odvodené sú vhodné na modelovanie časových radov ako je sekvencia pohľadu a dokážu opísať pohyby

Obr. 2.4: *Ukážka Skrytého Markovovho modelu, kde sú skryté stavy (na obrázku červená, modrá a zelená elipsa) reprezentované pomocou dvoj-dimenzionálneho normálneho rozdelenia (obrázok z Coutrot et al. [2017]).*

očí lepšie ako agregované štatistiky (napr. priemerná dĺžka fixácie), pretože dokážu modelovať aj zmeny ich štatistických vlastností v čase.

Ich použitie spočíva v ich vlastnostiach a ďalších algoritmoch, ktoré sa dajú nad nimi použiť. Najznámejším algoritmom spojeným so Skrytým Markovovým modelom je Viterbiho algoritmus (Viterbi). Viterbiho algoritmus sa používa pre už natrénovaný model a slúži na nájdenie najpravdepodobnejších postupností skrytých stavov pre dané dáta.

Pre Skryté Markovove modely existuje viacero prístupov ako ich zhlukovať, ak ich máme viacero. Zhlukovanie Skrytých Markovovych modelov však musí riešiť viacero problémov. Štandardný spôsob zhlukovania súvisí s porovnávaním vektorov v priestore (napríklad algoritmus K-means), čo sa dá použiť na parametre Skrytého Markovovho modelu. Medzi parametrami však nemusí byť lineárny vzťah a taktiež modely s rôznymi parametrami a teda vo vektorovom priestore od seba vzdialené môžu mať rovnaké generatívne vlastnosti (príkladom je dvojica Skrytých Markovovych modelov so zmenenou postupnosťou skrytých stavov). Ďalšou úlohou pri zhlukovaní modelov je nájdenie alebo vytvorenie reprezentatívneho modelu pre zhluky.

Jedno riešenie opísané v článku Jebara et al. [2007] používa spektrálne zhlukovanie (angl. *spectral clustrening*) na vytvorenie zhlukov. Toto riešenie však nevytvára pre zhluky reprezentatívne modely, ale jedným zo spôsobov ako ho získať, je vybrať jeden model z každého zhluku, ktorý je najbližšie k centru. Riešenie v článku Coviello et al. [2014] používa na zhlukovanie *variačný hierarchický algoritmus maximalizácie očakávaní* (angl. *variational hierarchical expectation-maximalization = VHEM*), ktorý dokáže nájsť zhluky a následne modely v danom zhluku spojiť do reprezentatívneho modelu na základe špeciálnej verzie algoritmu maximalizácie očakávaní.

2.7.3 Iné pravdepodobnostné metódy

Existujú aj iné prístupy, ktoré sa zaoberajú spracovaním postupnosti pohľadu používateľa na základe pravdepodobnosti. Jedným z týchto prístupov je spomenutý v článku Kübler et al. [2017], kde sa porovnávajú postupnosti pohľadu (angl. *scanpath*) používateľov pomocou matice prechodov. Jednou z výhod oproti niektorým modelom spomínaným v časti 2.6.1 je, že prechodové pravdepodobnosti môžu byť normalizované pre každú postupnosť pohľadu zvlášť, čo umožňuje porovnávať aj postupnosti s rôznou dĺžkou.

2.8 Spôsoby overenia a vyhodnotenia modelov

Medzi štandardné spôsoby overenia úspešnosti modelu patrí porovnanie metrík na vyhodnotenie so základným (angl. *baseline*) modelom. Tento spôsob slúži na zistenie, či je vôbec možné modelovať dané dáta, alebo na poskytnutie základnej úrovne úspešnosti na zistenie, či má použitý prístup zmysel. Medzi časté základné modely, ktorých úlohou je zistenie, či má zmysel modelovať dáta, patrí náhodný model (angl. *guessing chance*) Kübler et al. [2017], Coutrot et al. [2017], ktorého úspešnosť závisí od počtu predikovaných tried alebo model, ktorý vyberá vždy triedu s najväčšou pravdepodobnosťou Steichen et al. [2014]. Pri testovaní nového prístupu závisí použitý základný model od riešenej úlohy. Príkladom je porovnanie s logistickou regresiou pri použití

Skrytého Markovovho modelu v článku Simola et al. [2008].

Samotné vyhodnocovanie súvisí s druhom modelu. Ak použitý model patrí do skupiny metód *učenia s učiteľom*, tak sa model testuje na testovacej sade, ktorú model nevidel pri učení. Existuje viacero variácií rozdeľovania dát, pričom medzi najpoužívanejšie patrí *krížová validácia* (angl. *cross-validation*), kde sa dáta rozdeľujú do určitého počtu skupín a vždy sa 1 skupina použije ako testovacia množina dát a ostatné ako trénovacie.

Na porovnanie modelov sa používajú metriky na vyhodnotenie, štatistické testy a kritériá. Vhodné vyhodnocovacie metriky na použitie súvisia s druhom strojového učenia. Pri vyhodnotení klasifikačných úloh sa často používa *úspešnosť* (angl. *accuracy*), ktorá vyjadruje počet správne určených zo všetkých Kit and Sullivan [2016], Kübler et al. [2017], Coutrot et al. [2017]. Problémom pri vyhodnotení modelu *úspešnosťou* je, ak sú v dátach nevyvážené triedy, ktoré môžu skresliť výsledky. Na porovnanie regresných úloh sa používa *stredná absolútna chyba* a *stredná kvadratická chyba*, ktoré počítajú spôsobenú chybu pri predikcii a taktiež metrika *R-squared*, ktorá vyjadruje aké množstvo variancie dát je vysvetlenej pomocou zvoleného modelu. Štatistické testy sa používajú na výpočet, či je získaná úspešnosť modelu štatisticky signifikantná oproti zvolenému základnému modelu. Kritériá ako je Bayesovské informačné kritérium sa používajú na vyhodnotenie podobných modelov natrénovaných nad tými istými dátami. Bayesovské informačné kritérium slúži na vybratie lepšieho modelu na základe penalizácie jeho komplexity.

2.9 Podobné práce

V tejto práci sa zaoberám spracovaním a analýzou pohľadu počas čítania štrukturovaného textu, jeho spracovaním pomocou pravdepodobnostných modelov a na základe natrénovaného pravdepodobnostného modelu vytvoriť ďalšie črty, ktoré vylepšujú zvolený štandardný model. Taktiež skúmam, ktorá kombinácia vstupných parametrov do pravdepodobnostných modelov dokáže najlepšie modelovať dáta z pohľadu používateľa pre vybranú doménu. Na

základe týchto úloh som našiel nasledovné podobné práce:

- Simola et al. [2008] - článok sa zaoberá spracovaním pohľadu pri čítaní textu pomocou pravdepodobnostných modelov.
- Coutrot et al. [2016] - použitie pravdepodobnostných modelov a ich zhlukovania na analýzu pohľadu na tvár.
- Coutrot et al. [2017] - použitie pravdepodobnostných modelov pri klasifikácií pohľadu pri sledovaní obrázku a videa do tried.

V článku od Simola et al. [2008] sa zameriavajú na predikciu vykonávanej úlohy z čítaného textu. V článku vykonávajú experiment na 3 typoch úloh a to:

- nájdenie určeného slova
- nájdenie odpovede na položenú otázku
- vybratie najzaujímavejšieho titulu spomedzi viacerých titulov

Na riešenie používajú diskriminačný Skrytý Markovov model, ktorého skrytý stav reprezentuje kognitívny stav pri čítaní textu. Vstupnými hodnotami pre ich model sú dĺžky trvania fixácií, vzdialenosti sakád, smer sakády a či ide o regresiu (či na danom slove už bola predtým fixácia). V článku používajú aj ďalšiu metódu a to logistickú regresiu s ktorou sa porovnávajú a taktiež na zistenie ďalších parametrov ako je velkosť okna fixácií. Dosahujú úspešnosť 60.2% a na záver navrhujú rozšírenie použitím Skrytého Markovvho modelu so vstupom a výstupom.

V rámci diela od Coutrot et al. [2016] sa snažili nájsť, ktoré vlastnosti najviac ovplyvňujú pohľad pozorovateľa pri pozorovaní tváre. Na túto úlohu použili Skrytý Markovov model, kde skryté stavy reprezentujú dvojdimenzionálnym normálnym rozdelením. Na trénovanie použili surové dáta, aby sa zohľadnili aj dĺžky trvania fixácií. Pomocou algoritmu VHEM rozdelili Skryté Markovove modely na niekoľko zhlukov (2 a viac) a skúmali, ktoré

charakteristiky pozorovateľa vplývajú na to, že pohľad používateľa patrí do daného zhluku. Na základe zhlukovania a štatistického testu MANOVA zistili, že vlastnosti Skrytých Markovovych modelov sú najviac ovplyvňované pohlavím pozorovateľa a taktiež pohlavím pozorovanej tváre. Na základe tohto zistenia natrénovali klasifikátor pomocou metódy kvadratickej analýzy diskriminantu (Quadratic Discriminant Analysis), ktorý dokázal s úspešnosťou 73.4% (náhoda = 50%) predikovať pohlavie pozorovateľa a s pravdepodobnosťou 54.3% (náhoda = 25%) predikovať pohlavie pozorovateľa a pohlavie pozorovanej tváre.

V článku Coutrot et al. [2017] je ich cieľom vytvoriť všeobecné riešenie na modelovanie a klasifikáciu pohľadu, ktoré by bolo použiteľné nezávisle od dátovej sady (dataset). Vytvorené riešenie overovali pomocou 2 testovaní, na ktorých sa snažili overiť, či model dokáže zohľadniť mechanizmy zhora-nadol a zdola-nahor, ktoré ovplyvňujú správanie pohľadu:

- Testovanie 1 = Zistenie úlohy z voľne dostupného datasetu z článku Koehler et al. [2014], kde participanti riešili nasledujúce úlohy:
  - Voľné pozorovanie obrázku
  - Zistenie, či sa najvýraznejšia lokalita nachádza vľavo, alebo vpravo
  - Zistenie, či sa daný objekt nachádza na obrázku
- Testovanie 2 = Predikcia, či sa v danom konverzačnom videu nachádza, alebo nenachádza originálna zvuková nahrávka z voľne dostupného datasetu z článku Coutrot and Guyader [2014].

Na riešenie používajú Skrytý Markovov model, ktorého skrytý stav reprezentuje región záujmu a je vyjadrený dvoj-dimenzionálnym normálnym rozdelením. Počet stavov zisťujú automaticky pomocou variačného Bayesovského odvodenia. Na klasifikáciu používajú viacero algoritmov pričom najlepšiu úspešnosť dosahujú, keď použijú metódu analýzy diskriminantu, pričom ako vstupné parametre klasifikátora sú natrénované parametre Skrytého Markovho modelu.

V článku Kit and Sullivan [2016] bol pohľad používateľov nahrávaný v dynamickom prostredí. Z tohto dôvodu sa rozhodol použiť ako vstupné črty do Skrytého Markovovho modelu dĺžku sakády a jej smer.

2.10 Zhrnutie analýzy a ciele

Pohľad používateľa v sebe obsahuje na základe zložiek, ktoré ho ovplyvňujú (opísané v časti 2.1), veľké množstvo informácií o vizuálnom stimule, vykonávanej úlohe a o pozorovateľovi a jeho vlastnostiach. Rôzne úlohy, ako čítanie prirodzeného textu (2.2) alebo štrukturovaného textu (2.3), majú odlišné vlastnosti pohľadu a taktiež vzory čítania a prehľadávania.

Pohľad je zachytávaný pomocou zariadení na sledovanie pohľadu, ktoré zachytávajú informácie o pohľade používateľa, ktoré sa neskôr spracujú na fixácie a sakády na základe rôznych algoritmov (2.4). Na modelovanie pohľadu sa používajú štandardné modely zo strojového učenia (2.6) s rôznymi črtami pohľadu (2.5).

V rámci analýzy sú opísané aj pravdepodobnostné modely (2.7), ktoré sú použité v oblasti spracovania pohľadu. Spôsoby vyhodnocovania modelov sú opísané v časti 2.8.

Na základe vykonanej analýzy v tejto kapitole som si v tejto práci stanovil nasledujúce výskumné otázky:

- Ktorý z vybraných pravdepodobnostných modelov a s akými parametrami a vlastnosťami dokáže najlepšie reprezentovať dáta z vybranej domény?
- Dokáže použitie pravdepodobnostných modelov významne zlepšiť predikciu vykonávanej úlohy, charakteristík používateľa alebo ďalších vlastností oproti modelom založených na vyhodnotení agregovaných údajov?

3 Tvorba čŕt založená na pravdepodobnosti vygenerovania Skrytými Markovovymi modelmi

V tejto kapitole je opísaná navrhnutá metóda. Metóda je založená na Skrytých Markovovych modeloch, podporných algoritmoch a využitia pravdepodobnosti vygenerovania sekvencie = *vierohodnosti*. Natrénované modely sa využijú na tvorbu čŕt, ktoré sa pridajú do pôvodnej dátovej sady a následne sa použijú na klasifikáciu pomocou zvolenej štandardnej metódy. Jednotlivé časti sú opísané v sekciách nižšie.

3.1 Vymedzenie úloh a overenie

Cieľom tejto metódy je vytvoriť črty, ktoré by abstrahovali informáciu o prehľadávaní vizuálneho stimulu a jeho vlastností v čase pre zvolenú úlohu. Zameranie je na klasifikačné úlohy na základe pohľadu používateľa, kde sú predpokladané rozlišné vlastnosti tried (má ich zmysel modelovať). So zvolenou úlohou a doménou súvisí použitý štandardný model na klasifikáciu a taktiež črty použité na trénovanie pravdepodobnostných a štandardných modelov.

Overenie metódy súvisí s danými výskumnými otázkami. Preskúmanie do akej miery zlepší použitie pravdepodobnostných modelov úspešnosť predikcie oproti modelom založeným na vyhodnotení agregovaných údajov, bude slúžiť porovnanie úspešnosti štandardných modelov, ktoré použili a nepoužili vytvorené črty z pravdepodobnostných modelov. Na zistenie, ktorý pravdepodobnostný model a aké vstupné parametre máju vplyv na úspešnosť budú testované a porovnávané rôzne variácie metódy a kombinácie parametrov testovaných pravdepodobnostných modelov.

3.2 Časti metódy

Navrhnutá metóda sa dá rozdeliť do nasledovných častí:

- Príprava dát a vytvorenie agregovaných čŕt
- Trénovanie Skrytých Markovovych modelov
  - rozdelenie na základe vizuálneho stimulu
  - pripravenie dát na trénovanie
  - trénovanie modelov
- Tvorba pravdepodobnostných čŕt
  - optimalizácia na základe pravdepodobnosti
- Klasifikácia novej vzorky

Časť prípravy dát a vytvorenia agregovaných čŕt je závislá na vybranej doméne a dátovej sade a je bližšie opísaná v kapitole overenia 4. Ostatné časti metódy sú opísané v sekciách nižšie.

Celkovú architektúru metódy je vidieť na obrázku 3.3.

3.3 Použitie metódy

Navrhnutá metóda potrebuje k svojej funkčnosti, aby sa použitá dátová sada skladala zo sekvencií fixácií s parametrami pozície fixácie ako súradnice x a y a taktiež parameter trvania fixácie. Dátová sata by mala ďalej obsahovať parametre, na základe ktorých sa dajú sekvencie fixácií rozdeliť (napríklad na základe rôznych participantov alebo úloh) a taktiež aby každá sekvencia obsahovala určitú triedu, ktorú sa snažíme predikovať - vhodné pre úlohy obsahujúce klasifikáciu.

Metóda slúži na obohatenie vytvorených agregovaných čŕt pre sekvencie ďalšími črtami vytvorenými na základe použitia Skrytých Markovovych modelov (ďalej len pravdepodobnostné črty).

3.4 Trénovanie Skrytých Markovovych modelov

V tejto časti je opísaný spôsob vytvárania Skrytých Markovovych modelov. Dôležitými vlastnosťami v tejto časti sú parametre vytváraných Skrytých Markovovych modelov, ktoré sú opísané v časti 2.7.2.1.

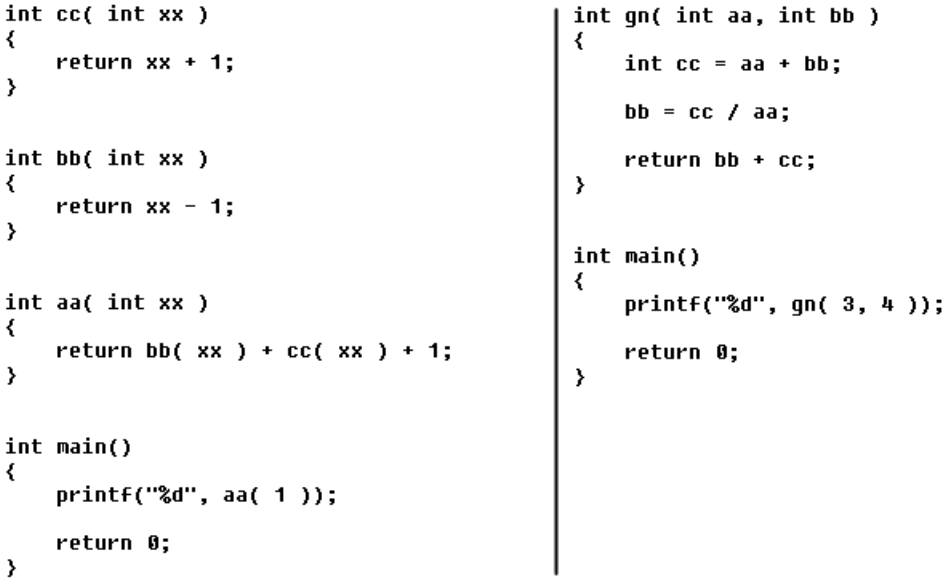
Medzi hlavné parametre Skrytých Markovovych modelov patrí počet skrytých stavov a ich reprezentácia. Počet skrytých stavov pre vytvárané modely spolu s kombináciou vstupných čŕt testujem na zistenie, ktorá kombinácia dokáže najlepšie reprezentovať vstupné dáta. Vhodné je použiť rovnaký počet skrytých stavov pre všetky vytvárané Skryté Markovove modely, aby sa jednoduchšie porovnávali. Skryté stavy v modeloch v tejto metóde reprezentujem pomocou multi-dimenzionálneho normálneho rozdelenia.

3.4.1 Rozdelenie na základe vizuálneho stimulu

Ešte pred prípravou dát na trénovanie Skrytých Markovovych modelov sa môžu rozdeliť dáta na viacero častí. Táto fáza sa používa ak dáta obsahujú úlohy, pri ktorých sa používatelia pozerali na dostatočne rôzne vizuálne stimuly. Dôvod tohto rozdelenia je taký, aby sa každý model učil len na jednom druhu vizuálneho stimulu, aby sa dokázal správne naučiť parametre. Pre každú časť dát sa potom vytvorí vlastný model, ktorý sa bude používať na tvorbu pravdepodobnostných čŕt len nad tou časťou, ktorá má rovnaký vizuálny stimul. Či je potrebné dáta rozdeliť taktiež súvisí s tým, aké črty sa používajú na trénovanie Skrytého Markovovho modelu a taktiež od vizuálnych stimulov, na ktoré sa pozeral participant počas experimentu. Ukážku rozdielnych vizuálnych stimulov je vidieť na obrázku 3.1.

3.4.2 Príprava dát na trénovanie

Pred začiatkom trénovania je potrebné pripraviť dáta do správneho formátu na základe toho, aké črty chceme použiť pri trénovaní pravdepodobnostného modelu. Ako vstupné dáta sa v tejto fáze používajú pôvodné neagregované sekvencie pohybov očí participanta (fixácie a sakády). Počet čŕt, ktoré sa



```
int cc( int xx )
{
    return xx + 1;
}

int bb( int xx )
{
    return xx - 1;
}

int aa( int xx )
{
    return bb( xx ) + cc( xx ) + 1;
}

int main()
{
    printf("%d", aa( 1 ));

    return 0;
}
```

```
int gn( int aa, int bb )
{
    int cc = aa + bb;

    bb = cc / aa;

    return bb + cc;
}

int main()
{
    printf("%d", gn( 3, 4 ));

    return 0;
}
```

Obr. 3.1: *Ukážka rozdielnych vizuálnych stimulov (kódy z analyzovaných dát vo vyhodnotení). Ako je vidieť na obrázku, tak jednotlivé časti sú rozdielne vysoké a dôležité elementy sú umiestnené na iných miestach. Použitie sekvencií z obidvoch vizuálnych stimulov by mohlo ovplyvniť správne naučenie modelu.*

použijú na trénovanie modelu, má vplyv na to, koľkými dimenziami budú reprezentované skryté stavy v modeli.

Na trénovanie modelov vytváram a používam kombináciu nasledovných čŕt, ktoré sú inšpirované črtami použitými v podobných prácach a z článku Steichen et al. [2013]:

- Fixácia X - horizontálna pozícia fixácie participanta na obrazovke.
- Fixácia Y - vertikálna pozícia fixácie participanta na obrazovke.
- Dĺžka sakády - ide o dĺžku sakády, ktorá vychádza z aktuálnej fixácie participanta.
- Absolútny uhol sakády - veľkosť uhla, ktorý zviera vektor vychádzajúcej sakády s vektorom [1, 0].

- Relatívny uhol sakády - veľkosť uhla, ktorý zviera vektor vychádzajúcej sakády s vektorom predchádzajúcej sakády (v prípade prvej fixácie, keď ešte neexistuje predchádzajúca sakáda, tak sa použije vektor [1, 0] na porovanie).
- Dĺžka fixácie - časové trvanie fixácie.

Ukážku, ako funguje absolútny a relatívny uhol sakády je vidieť v kapitole analýza na obrázku 2.2. Keďže na výpočet parametrov dĺžky sakády, absolútneho uhla sakády a relatívneho uhla sakády potrebujem vždy aspoň 2 fixácie, tak poslednú fixáciu zahadzujem.

Aby sa bol Skrytý Markovov model schopný naučiť svoju reprezentáciu z trénovacej sekvencie dát, je vhodné, aby daná sekvencia čŕt obsahovala aspoň toľko prvkov, koľko má daný model skrytých stavov. Z tohto dôvodu zahadzujem všetky sekvencie, ktoré majú menší počet prvkov (počet prvkov = počet fixácií + 1, kvôli zahadzovaniu poslednej fixácie) ako je počet skrytých stavov.

3.4.3 Trénovanie modelov

Vytvorené črty používam na trénovanie pravdepodobnostných modelov. Na naučenie jednotlivých parametrov (prechodová matica, pravdepodobnosti začatia v skrytom stave a nastatia pozorovateľného stavu) modelu používam Baum-Welchov algoritmus, pričom iniciálne hodnoty parametrov sa nastavia na základe K-means zhlukovania nad hodnotami z trénovacích sekvencií čŕt.

Počet vytvorených modelov súvisí s počtom predikovaných tried pri klasifikácii. Vstupné dáta sa ešte pred použitím na trénovanie rozdelia do skupín na základe ich predikovanej triedy. Každý model sa trénuje iba pomocou jednej skupiny.

V prípade ak dáta nie sú vhodné na trénovanie (nastáva v prípade, ak existujú len vzorky, ktoré majú príliš krátke sekvencie na natrénovanie modelu, alebo sú dáta prázdne), použijeme namiesto natrénovaného modelu objekt,

ktorý reprezentuje špeciálnu verziu modelu (použitý návrhový vzor *Null Object*). Tento objekt implementuje funkcie na výpočet pravdepodobnosti vygenerovania sekvencie a po ich zavolaní vracia základné minimálne hodnoty (0% pravdepodobnosť vygenerovania).

3.5 Tvorba pravdepodobnostných čŕt

Vytvorené modely z predchádzajúcej časti sa následne využijú na tvorbu pravdepodobnostných čŕt. Pre jednotlivé elementy v agregovanej dátovej sade sa nájdu pôvodné sekvencie, na základe ktorých boli vytvorené a každá sa najprv transformuje na sekvenciu vstupných čŕt do pravdepodobnostného modelu pomocou procesu opísaného v časti 3.4.2.

Pravdepodobnostné črty sa vytvárajú na základe pravdepodobnosti vygenerovania sekvencie daným modelom alebo vierohodnosti sekvencie (angl. *likelihood*) (ďalej len pravdepodobnosť vygenerovania). Použitím pravdepodobnosti vygenerovania z každého natrénovaného pravdepodobnostného modelu vytváram nasledovné črty:

- logaritmus pravdepodobnosti vygenerovania pre každý model
- relatívna pravdepodobnosť vygenerovania pre každý model - normalizované pravdepodobnosti vygenerovania z modelov tak, aby ich súčet bol rovný hodnote 1
- rozdiel medzi pravdepodobnosťami vygenerovania (používa sa iba vtedy, ak sú len 2 predikované triedy) - rozdiel medzi pravdepodobnosťami vygenerovania
- predikovaná trieda - táto črta nadobúda hodnotu 1 pre model, ktorý má najvyššiu pravdepodobnosť vygenerovania pre danú sekvenciu, inak má hodnotu 0

Ukážku pravdepodobnostných čŕt je vidieť na obrázku 3.2.

HMM_0.0	HMM_0.0p	HMM_1.0	HMM_1.0p	HMM_diff	HMM_0.0p_1hot	HMM_1.0p_1hot
-22.467420	0.651866	-23.094672	0.348134	-0.303732	1	0
-24.695910	0.015911	-20.571173	0.984089	0.968179	0	1
-20.183632	0.604181	-20.606549	0.395819	-0.208362	1	0
-20.294861	0.433811	-20.028540	0.566189	0.132379	0	1
-31.673298	0.951124	-34.641650	0.048876	-0.902247	1	0
-19.439109	0.747768	-20.525854	0.252232	-0.495537	1	0
-15.553245	0.513272	-15.606345	0.486728	-0.026544	1	0
-20.158965	0.499596	-20.157349	0.500404	0.000808	0	1

Obr. 3.2: *Ukážka vytvorených pravdepodobnostných čŕt. HMM_0.0 a HMM_1.0 sú logaritmy pravdepodobnosti vygenerovania. HMM_0.0p a HMM_1.0p sú relatívne pravdepodobnosti vygenerovania. HMM_diff je rozdiel medzi pravdepodobnosťami vygenerovania a HMM_0.0p_1hot a HMM_1.0p_1hot reprezentujú predikovanú triedu. Hodnoty 0.0 a 1.0 sú predikované triedy.*

3.5.1 Optimalizácia na základe minimálnej pravdepodobnosti

Cieľom tejto fázy je maximalizovať užitočnosť vygenerovaných pravdepodobnostných čŕt. Na použitie je potrebné vopred rozdeliť vstupné trénovacie dáta na trénovaciu a validačnú časť.

Na základe validačnej vzorky sa zistí hranica pravdepodobnosti, od ktorej má zmysel vytvárať pravdepodobnostné črty. Výpočet hranice pravdepodobnosti je na základe nasledovného vzorca v ktorom je cieľom optimalizovať kvalitu na základe zmeny hodnoty N:

$$kvalita = (uspesnost\_N - nahodna\_uspesnost) * \frac{pocet\_N}{celkovy\_pocet}$$

- uspesnost_N - úspešnosť (reálna trieda = predikovaná trieda) sekvencií vo validačnej vzorke, ktoré majú aspoň pravdepodobnosť N
- nahodna_uspesnost - úspešnosť pri náhodnom modeli (pre 2 triedy je náhodná úspešnosť = 50%)
- pocet_N - počet sekvencií vo validačnej vzorke, ktoré majú aspoň pravdepodobnosť N
- celkovy_pocet - celkový počet sekvencií vo validačnej vzorke

Na základe dosiahnutia pravdepodobnostnej hranice sme navrhli 2 verzie metódy: jednoduchú a striktnú verziu. V jednoduchej verzii sa v prípade nedosiahnutia pravdepodobnostnej hranice modelmi nevyberá pre daný záznam žiadna predikovaná trieda (všetky črty predikovanej triedy budú mať hodnotu 0). V striktnej verzii sa v prípade, ak daná sekvencia nedosiahne pravdepodobnostnú hranicu, upravia aj ostatné vygenerované črty (nastaví sa im základná predvolená hodnota).

3.6 Klasifikácia novej vzorky

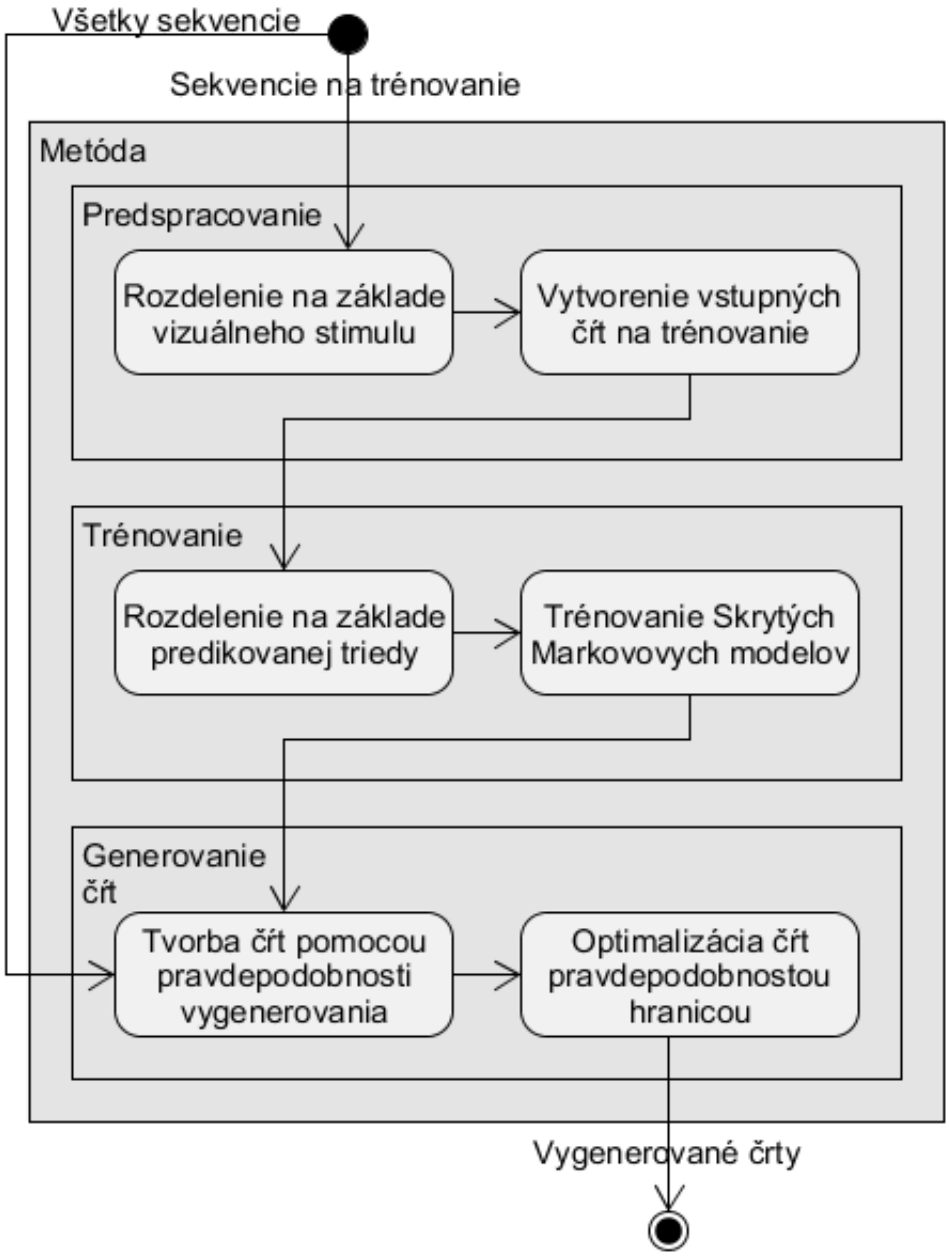
V tejto časti sú opísané potrebné kroky na spracovanie novej sekvencie dát. Na vytvorenie pravdepodobnostných čŕt pre novú sekvenciu je potrebné mať natrénované Skryté Markovove modely, ktoré sú výstupom z trénovania ako je vidieť na obrázku architektúry 3.3. Sekvenciu je najprv potrebné pripraviť do správneho formátu pomocou krokov zo sekcie 3.4.2 na vstupné črty do pravdepodobnostného modelu. Následne sa z vytvorených vstupných čŕt pre model vytvoria pravdepodobnostné črty z časti 3.5 pomocou natrénovaných Skrytých Markovovych modelov.

3.7 Kombinované modely

Ide o verziu metódy, kde je možné použiť pre každé rozdelenie dát na základe vizuálneho stimulu 3.4.1 inú kombináciu vstupných čŕt. V prípade, ak nie je pre dané rozdelenie explicitne zapísaná kombinácia, tak sa použije predvolená používateľom. Má ich zmysel použiť iba vtedy, keď používame rozdelenie na základe vizuálneho stimulu.

Hlavné využitie kombinovaných modelov je na optimalizáciu úspešnosti v prípade, ak sú rozdelenia dát dostatočne odlišné a kombinácie vstupných čŕt majú rozdielne výsledky v závislosti od toho na ktorom rozdelení dát boli použité. Pre každé rozdelenie sa môže použiť taká kombinácia vstupných čŕt, ktorá má pre daný druh dát najlepšiu úspešnosť. Nevýhodou pri kombinačných modeloch je, že je na správne nastavenie potrebné otestovať a vyhodnotiť viacero kombinácií vstupných čŕt na optimalizáciu, čo je časovo viac náročné.

Kombinované modely súvisia s prípravou dát na trénovanie a klasifikovaním novej vzorky, kedy pomocou náhradnej kombinácie predpisujú, do akých vstupných čŕt sa transformujú dáta.



Obr. 3.3: *Architektúra navrhovanej metódy.*

4 Vyhodnotenie

V tejto kapitole sú opísané kroky použité pri vyhodnocovaní metódy, výsledky pri rôznych nastaveniach, realizácia riešenia a použité dátové sady.

4.1 Dostupné vzorky dát a úlohy

V tejto časti sú opísané dostupné vzorky dát, ktoré som použil v rámci tejto práce.

4.1.1 Menší experiment so začínajúcimi a mierne pokročilými programátormi

Tento experiment obsahoval 27 participantov z prvej skupiny, ktorí sú začínajúci programátori a 28 participantov z druhej skupiny, ktorí sú mierne pokročilí programátori. Úlohou experimentu bolo pochopiť zdrojový kód programov v programovacom jazyku C a napísať výstup po ich vykonaní. Experiment obsahoval nasledovných 6 programov, pričom posledné 2 programy dostalo len malé množstvo účastníkov:

- Program 1 (code-loop_array) - funkcia na filtrovanie deliteľov bezo zvyšku z daného pola
- Program 2 (code-short_eval) - program na testovanie postupnosti vykonávania operácií
- Program 3 (code-bubble_sort) - bublinkové usporiadanie s obmedzeným množstvom prechodov
- Program 4 (code-recursion) - rekurzívna funkcia na spočítanie párnych čísiel
- Program 5 (code-numbers) - zmena hodnôt 3 premenných počas cyklu

- Program 6 (code-fruit) - postupnosť volania viacerých funkcií na základe operácií

Všetci participanti dostali programy v rovnakom poradí ako sú napísané v zozname vyššie. Participanti mali počas experimentu zaznamenávaný pohľad pomocou zariadenia na sledovanie pohľadu.

Pri vyhodnotení boli napísané výstupy rozdelené do 2 kategórií na základe správnosti. Hodnotou 1 boli označené správne odpovede a hodnotou 0 boli označené nesprávne odpovede. Výsledky je vidieť na obrázku 4.1. Na základe obrázku s výsledkami je vidieť, že najťažším programom na pochopenie bol program 2 a 3 (code-short_eval a code-bubble_sort). Pri programe 3 si len málo participantov uvedomilo, že bublinkové usporiadanie má obmedzené množstvo prechodov.

Obr. 4.1: *Obrázok s výsledkami z menšieho experimentu. Červenou farbou sú znázornené nesprávne odpovede a zelenou správne. Úlohy* ***code-fruit*** *a* ***code-numbers*** *dostalo len zopár participantov preto majú menej vzoriek. Úloha* ***code-fruit*** *bola pre participantov jednoduchá. Ostatné úlohy majú ako tak vyvážené triedy.*

4.1.2 Väčší experiment so začínajúcimi programátormi

Tento experiment prebiehal v 4 behoch, pričom medzi behmi boli dlhšie prestávky (niekoľko týždňov až mesiacov). Štvrtý beh na rozdiel od ostatných behov obsahoval ešte pred začiatkom experimentu aj prezentáciu s vysvetlením, ako programy v experimente fungujú. Medzi jednotlivými behmi sa menili hodnoty v programoch, aby sa predišlo naučeniu správnych odpovedí účastníkmi medzi behmi. Experiment obsahoval 139 participantov pre ktorých

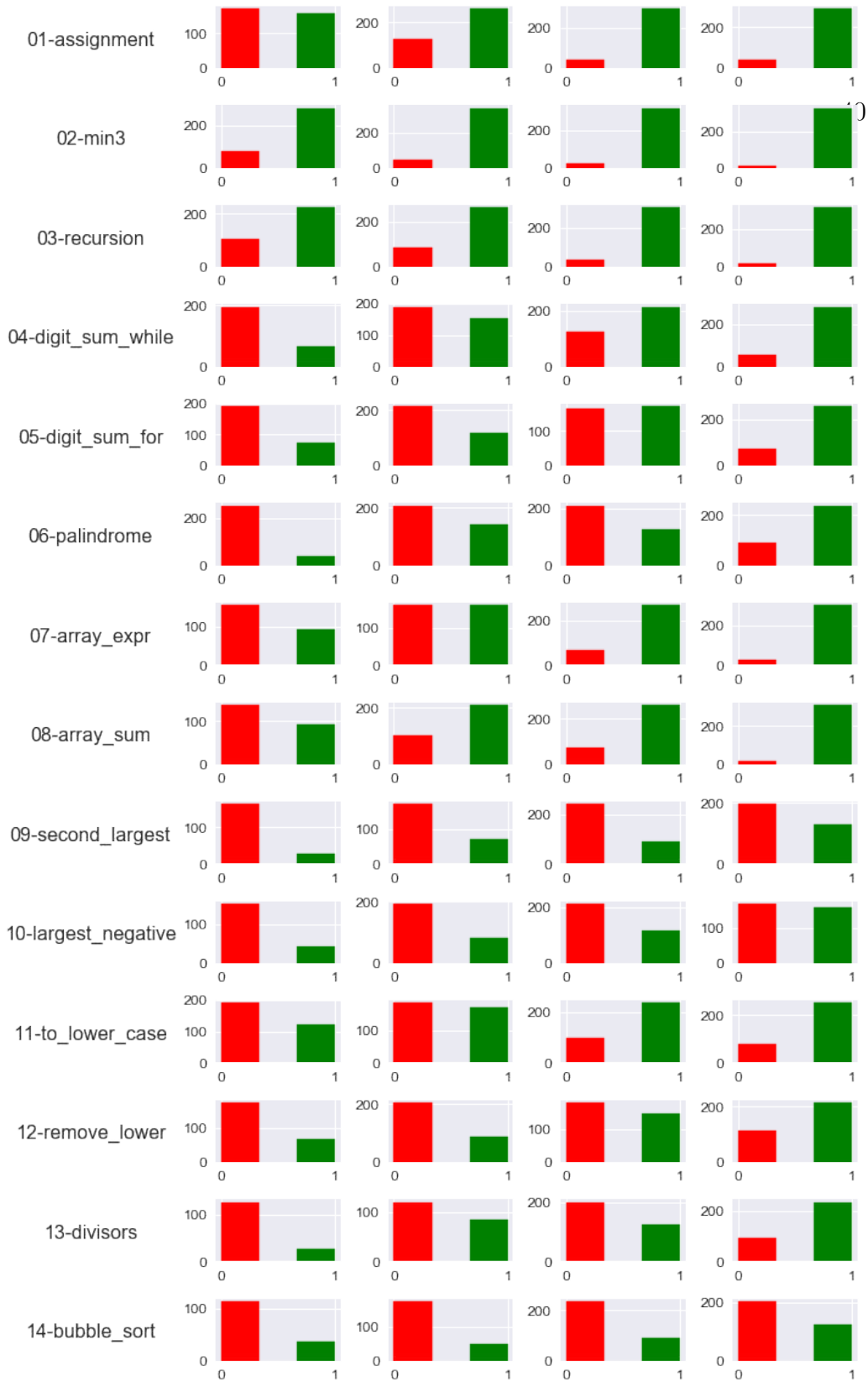
sú dostupné dáta, pričom ale nie všetci participanti boli zúčastnení v každom behu. V behoch bola nasledovná účasť:

- 1. beh - 135 participantov
- 2. beh - 139 participantov
- 3. beh - 114 participantov
- 4. beh - 111 participantov

Úlohou experimentu bolo pochopiť zdrojový kód programov v programovacom jazyku C a zadať výstup po ich vykonaní. Experiment obsahoval 15 variácií programov, pričom každá variácia obsahovala 3 verzie. 1 variáciu programu (00-tutorial) som vyradil, pretože išlo o tutorial do experimentu s jasným výsledkom.

Všetci participanti dostali programy v rovnakom poradí a na vypracovanie každého programu mali obmedzený čas, ktorý videli na vrchu obrazovky. V rámci tohto experimentu nemuseli participanti ručne zadávať výstupy, ale museli si vybrať z 5 možností. 3 možnosti obsahovali hodnotu a ďalšie 2 hodnoty boli 'iné' a 'neviem'. Participanti mali počas experimentu zaznamenávaný pohľad pomocou zariadenia na sledovanie pohľadu.

Pri vyhodnotení boli napísané výstupy rozdelené do 2 kategórií na základe správnosti. Hodnotou 1 boli označené správne odpovede, hodnotou 0 boli označené nesprávne odpovede. V prípade, ak participant úlohu neriešil, alebo zadal možnosť 'neviem', tak daný záznam bol vyradený. Výsledky je vidieť na obrázku 4.2. Náročnosť programov na vyriešenie postupne rástla, čo je vidieť aj na výsledkoch.



Obr. 4.2: *Obrázok s výsledkami z väčšieho experimentu. Pozícia stĺpca vyjadruje, o ktorý beh ide (1. stĺpec = 1. beh). Červenou farbou sú znázornené nesprávne odpovede a zelenou správne. Ako je na obrázkoch vidieť, pri neskorších úlohách je menej vzoriek (pri úlohe 01-priradenia je vzoriek okolo 300 a pri 14-bubble je vzoriek už len okolo 150), pretože nie každému participantovi sa tam podarilo dostať alebo viacej participantov vybralo možnosť 'neviem' (hlavne viditeľné na behoch 1 a 2).*

4.2 Realizácia

Všetky výsledky v tejto kapitole boli získané pomocou implementácie navrhnutej metódy v jazyku *Python 3.6*. Na analýzu a spracovanie dát boli použité knižnice *pandas*, *numpy* a *scikit-learn*. Na tvorbu Skrytých Markovovych modelov bola použitá knižnica *pomegranate*. Na normalizáciu dát, výpočet metrík a štandardné klasifikátory bola použitá knižnica *scikit-learn*. Konkrétnejší opis realizácie je v technickej dokumentácií v prílohe B. Riešenie boli vytvorené v jupyter notebookoch (web aplikácia, ktorá umožňuje spájať kód s dokumentáciou cez spustiteľné bunky) spustiteľných cez knižnicu *jupyter*.

4.2.1 Načítanie a príprava dát

Po načítaní dát odstraňujeme neplatné záznamy (záznamy, ktoré sa nepodarilo zariadeniu na sledovanie pohľadu správne zachytiť, alebo tie ktoré sú mimo obrazovky) a necháme len tie sekvencie pohľadu, ktoré obsahujú výsledok.

Zostávajúce sekvencie rozdelíme na základe vykonávanej úlohy a participanta a pre každé rozdelenie vypočítame nasledujúce agregované črty:

- Celkové trvanie
- Pomer fixácií
- Priemer dĺžky sakád
- Smerodajná odchýlka dĺžky sakád
- Najdlhšie trvanie fixácie
- Priemer trvaní fixácií
- Smerodajná odchýlka trvaní fixácií

Po agregácií zahadzujeme záznamy, ktoré obsahujú neplatné hodnoty (chýbajúce hodnoty, alebo nekonečno).

4.2.2 Implementácia metódy

Logika navrhnutej metódy je vyjadrená v triede, ktorá dostane pri inicializácií akú kombináciu vstupných čŕt má použiť na trénovanie a počet skrytých stavov. Po vytvorení inštancie tejto triedy sa dá použiť metóda na trénovanie modelov (2 variácie - bez výpočtu a s výpočtom pravdepodobnostnej hranice) a generovanie nových čŕt. Funkcionalita je opísaná v návrhu tejto práce v kapitole 3.

4.2.3 Trénovanie a výpočet výsledkov

Pri trénovaní používame na rozdelenie dát stratifikovanú krížovú validáciu (angl. *stratified cross validation*) nad každou úlohou v dátovej sade. Strafikácia zabezpečuje aby každé rozdelenie lepšie reprezentovalo celkovú množinu. Na základe rozdelenia agregovaných dát na trénovaciu a testovaciu množinu z krížovej validácie vypočítame aj trénovaciu a testovaciu množinu sekvencií fixácií.

Pri trénovaní sa vytvára klasifikátor pre celú trénovaciu dátovu množinu a taktiež pre každú úlohu zvlášť. Na základe vstupnej kombinácie sa rozhodne, či sa majú dáta rozšíriť o temporálne črty vytvárané navrhnutou metódou. Výsledky sa zbierajú pre každý vytvorený klasifikátor a následne vyhodnocujú na základe zvolených metrík.

4.3 Analýza a optimalizácia parametrov pravdepodobnostných modelov

4.3.1 Optimalizácia parametrov nad menšou dátovou sadou

Aby sme našli parametre pravdepodobnostných modelov, ktoré majú najlepšiu úspešnosť, tak sme otestovali pravdepodobnostné modely s rôznymi kombináciami vstupných čŕt, ktoré sú opísané v časti 3.4.2 (otestované boli všetky črty okrem trvania fixácie, ktorá vtedy ešte nebola implementovaná) a počtom skrytých stavov (otestované počty 3 až 5). Testovanie prebiehalo na menšej dátovej sade 4.1 a na testovanie bol použitý klasifikátor sklonom

posilňovaných stromov (angl. *gradient boosted trees*). Na výpočet úspešnosti bola použitá metrika F1 micro (opísaná v sekcii 4.4)

Pre každú kombináciu parametrov bola otestovaná všeobecná úspešnosť pomocou 1 klasifikátora pre celé dáta a taktiež úspešnosť pre úlohy, kde bol vytvorený 1 klasifikátor pre každú úlohu. Úlohy v tomto prípade predstavovali jednotlivé programy a celková úspešnosť bola vypočítaná ako priemerná úspešnosť klasifikátorov.

Parametre modelov s najlepšou všeobecnou úspešnosťou je vidieť v tabuľke 4.1. Parametre modelov s najlepšou úspešnosťou pre úlohy sú v tabuľke 4.2.

Tabuľka 4.1: *Parametre modelov s najlepšou všeobecnou úspešnosťou (úspešnosť 1 klasifikátora netrénovaného nad celými dátami).*

Fixácia X	**Fixácia Y**	**Dĺžka sakády**	**Abs. uhol**	**Rel. uhol**	**Skryté stavy**	**Úspešnosť**
áno	áno	áno	áno	nie	3	0.688
áno	nie	áno	áno	nie	3	0.664
áno	áno	áno	nie	áno	3	0.656
nie	nie	nie	áno	nie	5	0.648
nie	nie	nie	áno	áno	3	0.648

Tabuľka 4.2: *Parametre modelov s najlepšou úspešnosťou pre úlohy (priemerná úspešnosť klasifikátorov natrénovaných pre jednotlivé úlohy).*

Fixácia X	**Fixácia Y**	**Dĺžka sakády**	**Abs. uhol**	**Rel. uhol**	**Skryté stavy**	**Úspešnosť**
áno	áno	áno	áno	nie	5	0.670
áno	nie	nie	nie	nie	4	0.652
áno	nie	áno	áno	nie	4	0.652
nie	áno	áno	nie	nie	5	0.643
áno	áno	áno	nie	nie	4	0.634

4.3.2 Doménová analýza parametrov

Na základe analýzy zdrojového kódu sme rozdelili úlohy vo väčšej dátovej sade 4.1.2 do kategórií na základe toho, či boli v danej oblasti zložité. Cieľom

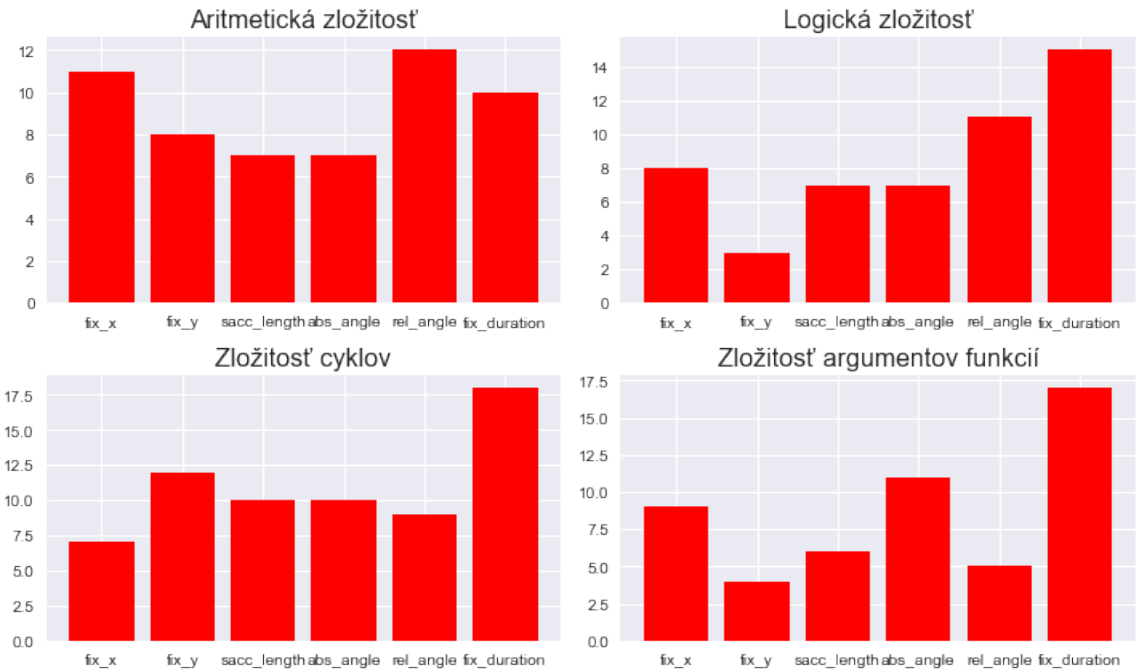
bolo zistiť, aké kombinácie čŕt a jednotlivé črty majú najlepšiu úspešnosť pri daných kategóriach. Na rozdelenie sme použili nasledovné kategórie:

- Aritmetická zložitosť - vyjadruje, či v programe boli komplexnejšie aritmetické operácie
- Logická zložitosť - určená na základe počtu a dĺžky podmienok
- Zložitosť cyklov - určená na základe počtu cyklov a vnorených cyklov
- Zložitosť argumentov funkcií - vyjadruje, či v programe boli funkcie s veľkým množstvom argumentov

Pre každú kategóriu sme vyhodnotili všetky kombinácie vstupných čŕt a zoradili ich na základe úspešnosti (angl. *Accuracy*). Úspešnosť bola vypočítaná na základe črty *predikovaná trieda* vytvorenej Skrytými Markovovymi modelmi. Kombinácie čŕt s najväčšou úspešnosťou pre jednotlivé kategórie sú v prílohe C. Na zistenie, ktoré črty majú najväčší význam sme zobrali 20 kombinácií s najväčšou úspešnosťou pre každú kategóriu a vypočítali početnosť danej črty. Výsledky sú vidieť na obrázku 4.3.

Na základe výsledkov je vidieť, že vstupná črta na trénovanie pravdpodobnostných modelov, ktorá má najväčší vplyv na správne určenie, či používateľ správne vyrieši úlohu je trvanie fixácie. Pomerne vysokú úspešnosť medzi úlohami majú ešte črty fixácia X a absolútny a relatívny uhol sakády.

Rozdielna vhodnosť vstupných čŕt v závislosti od kategórie úlohy môže súvisieť s rozdielnym spôsobom vykonávania programov a stratégií prehľadávania kódu pozorovateľom. Príkladom môže byť kategória úloh so zložitejšími cyklami, kde sa ako druhý najlepší parameter umiestnila y-ová súradnica fixácie, ktorá čiastočne reprezentuje na aký riadok sa používateľ pozerá. Z tohto vyplýva, že je nejaký zásadný rozdiel medzi tým ako sa v čase pozerá na jednotlivé riadky používateľ, ktorý správne vyriešil úlohu od toho, ktorý ju vyriešil nesprávne pri úlohách, ktoré sú zložité kvôli cyklom.



Obr. 4.3: *Dôležitosť jednotlivých vstupných čŕt pre rôzne kategórie zložitosti kódov. Na x-ovej osi sú za sebou v poradí črty Fixácia X, Fixácia Y, Dĺžka sakády, Absolútny uhol sakády, Relatívny uhol sakády a Dĺžka fixácie, ktoré sú bližšie opísané v časti 3.4.2. Y-ová os vyjadruje ako často sa vyskytovala daná črta v 20 najlepších kombináciach.*

4.4 Výsledky

V tejto sekcii sú opísané aktuálne výsledky mojej metódy nad dostupnými dátovými sadami a ich porovnanie so základným modelom, ktorý nepoužíva navrhnuté črty. Agregované črty základného modelu boli prebrané z článku Simola et al. [2008] a čiastočne z článku Coutrot et al. [2017]. Na zvýšenie presnosti som taktiež použil pri vyhodnocovaní krížovú validáciu, pričom výsledné metriky som vypočítal ako priemer z jednotlivých behov. Na vyhodnotenie používam nasledovné metriky:

- F1 micro - metrika F1 je harmonický priemer presnosti a pokrytia. Priemerovanie cez 'micro' namiesto 'macro' je výhodné, ak sú v dátovej sade pomerne vyvážené triedy.

- presnosť (angl. *precision*) - vyjadruje aké percento zo všetkých elementov, ktoré sme označili danou triedou, bolo určené správne
- pokrytie (angl. *recall*) - vyjadruje aké percento zo všetkých elementov, ktoré majú danú triedu, sa nám podarilo nájsť (správne označiť)

Na vyhodnotenie som použil viacero druhov klasifikátorov, ktoré reprezentujú skupiny modelov spomínaných v časti 2.6. Na vyhodnotenie som použil sklonom posilňované stromy (angl. *gradient boosted trees*), podporný vektorový stroj (angl. *support vector machine*) a logistickú regresiu (angl. *logistic regression*). Pre klasifikátory *podporný vektorový stroj* a *logistická regresia* som ešte normalizoval dáta pomocou strednej hodnoty a variancie.

Na vyhodnotenie menšej dátovej sady používam skupiny parametrov, ktoré som získal pomocou hľadania optimálnych parametrov v časti 4.3. Keďže počas hľadania som nevyskúšal parameter dĺžky trvania fixácií, tak som rozšíril použité kombinácie o tento parameter.

Na vyhodnotenie väčšej dátovej sady sme znovu spravili mriežkové prehľadávanie, pomocou ktorého vyhodnocujeme klasifikátor sklonom posilňovaných stromov. Na vyhodnotenie ostatných klasifikátorov používame len podskupinu najlepších kombinácií zistených z mriežkového prehľadávania kvôli časovým obmedzeniam a veľkosti dátovej sady.

4.4.1 Menší experiment so začínajúcimi a mierne pokročilými programátormi

V tejto časti sú opísané výsledky klasifikátorov použitými nad dátami z experimentu opísaného v sekcii 4.1.1. Výsledky pre klasifikátor *sklonom posilňovaných stromov* sú v tabuľke 4.3, pre klasifikátor *podporný vektorový stroj* v tabuľke 4.4 a pre klasifikátor *logistickej regresie* v tabuľke 4.5. Riadok, kde majú parametre hodnotu '-' ukazuje výsledky bez použitia pravdepodobnostných čŕt.

Ako je vidieť, nie pre každý klasifikátor majú pravdepodobnostné črty rovnaké zlepšenie. Najväčší rozdiel v úspešnosti má použitie pravdepodobnost-

ných čŕt pre klasifikátor *sklonom posilňovaných stromov*, kde je pri určitých kombináciach parametrov vidieť zlepšenie až o 5%.

Tabuľka 4.3: *Výsledky modelov pre klasifikátor* ***sklonom posilňovaných stromov****. Stĺpce* ***Fix. X****,* ***Fix. Y****,* ***Dĺžka sak.****,* ***Abs. uhol****,* ***Rel. uhol*** *a* ***# stavov*** *(počet skrytých stavov) vyjadrujú použité parametre na trénovanie pravdepodobnostného modelu.* ***F1 gen*** *vyjadruje úspešnosť všeobecného klasifikátora a* ***F1 task*** *vyjadruje priemernú úspešnosť klasifikátorov pre jednotlivé úlohy.*

Fix. X	Fix. Y	Dĺžka sak.	Abs. uhol	Rel. uhol	Dĺžka fix.	# stavov	F1 gen	F1 task
áno	nie	nie	nie	nie	nie	4	0.648	0.648
áno	nie	nie	nie	nie	áno	4	0.612	0.678
áno	áno	áno	nie	nie	nie	3	0.618	0.642
nie	nie	nie	áno	nie	áno	3	0.625	0.632
-	-	-	-	-	-	-	0.603	0.623

Tabuľka 4.4: *Výsledky modelov pre klasifikátor* ***podporný vektorový stroj****. Stĺpce* ***Fix. X****,* ***Fix. Y****,* ***Dĺžka sak.****,* ***Abs. uhol****,* ***Rel. uhol*** *a* ***# stavov*** *(počet skrytých stavov) vyjadrujú použité parametre na trénovanie pravdepodobnostného modelu.* ***F1 gen*** *vyjadruje úspešnosť všeobecného klasifikátora a* ***F1 task*** *vyjadruje priemernú úspešnosť klasifikátorov pre jednotlivé úlohy.*

Fix. X	Fix. Y	Dĺžka sak.	Abs. uhol	Rel. uhol	Dĺžka fix.	# stavov	F1 gen	F1 task
áno	áno	áno	nie	nie	nie	3	0.621	0.674
nie	áno	áno	nie	nie	nie	3	0.644	0.638
áno	nie	nie	nie	nie	áno	3	0.638	0.643
áno	áno	áno	nie	áno	nie	3	0.625	0.643
-	-	-	-	-	-	-	0.616	0.651

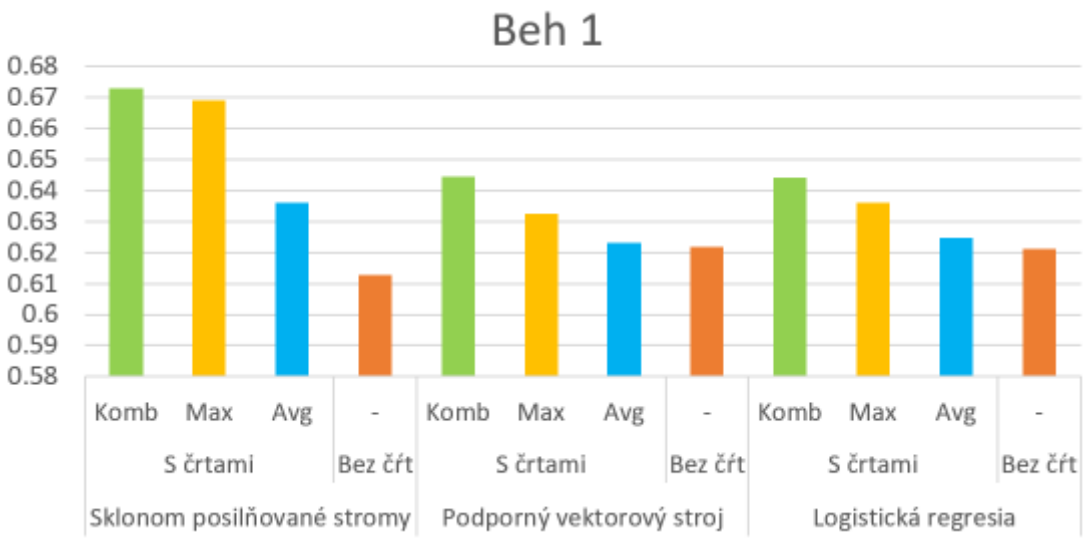
Tabuľka 4.5: *Výsledky modelov pre klasifikátor* ***logistickou regresiou****. Stĺpce* ***Fix. X****,* ***Fix. Y****,* ***Dĺžka sak.****,* ***Abs. uhol****,* ***Rel. uhol*** *a* ***# stavov*** *(počet skrytých stavov) vyjadrujú použité parametre na trénovanie pravdepodobnostného modelu.* ***F1 gen*** *vyjadruje úspešnosť všeobecného klasifikátora a* ***F1 task*** *vyjadruje priemernú úspešnosť klasifikátorov pre jednotlivé úlohy.*

Fix. X	Fix. Y	Dĺžka sak.	Abs. uhol	Rel. uhol	Dĺžka fix.	# stavov	F1 gen	F1 task
áno	áno	áno	áno	nie	nie	3	0.623	0.672
áno	nie	nie	nie	nie	áno	3	0.619	0.666
-	-	-	-	-	-	-	0.613	0.666
nie	nie	nie	áno	nie	nie	3	0.598	0.680
áno	áno	áno	nie	nie	áno	3	0.612	0.666

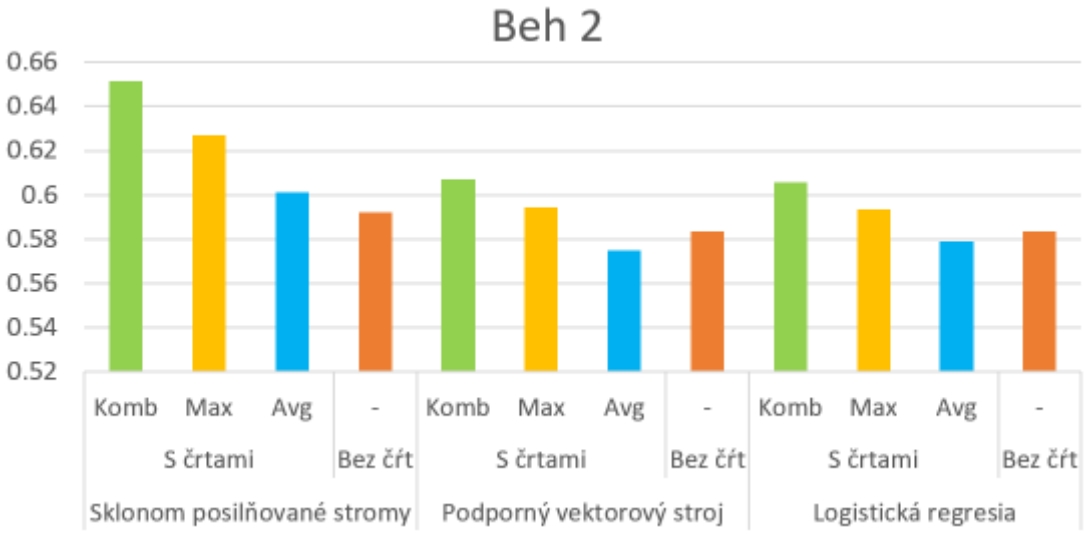
4.4.2 Väčší experiment so začínajúcimi programátormi

Na obrázkoch 4.4, 4.6, **??** a 4.7 je vidieť výsledky pre behy 1, 2, 3 a pre kombináciu behov 1, 2 a 3. Na obrázkoch sú znázornené 3 výsledky s použitím čŕt a 1 výsledok bez použitia čŕt pre každý klasifikátor. Priemerný výsledok s črtami bol vypočítaný ako priemer všetkých vyskúšaných kombinácií čŕt. Tento výsledok reprezentuje úspešnosť po 1 použití metódy s náhodnou kombináciou čŕt. Maximálny výsledok s črtami je kombinácia s najvyššou úspešnosťou. Kombinovaný model používa na rozdiel od ostatných na naučenie viacero kombinácií a jeho funkcionalita je bližšie opísaná v návrhu v časti 3.7. Na naučenie kombinovaného modelu bola pre každú úlohu zvolená kombinácia, ktorá mala najvyššiu úspešnosť na trénovacích dátach.

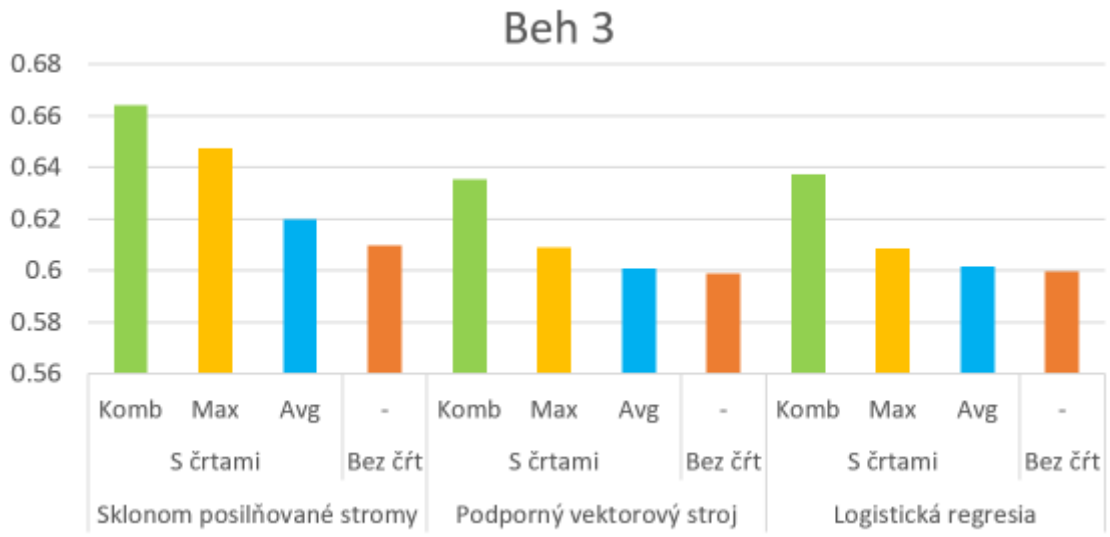
Výsledky všetkých kombinácií pre prvý beh (zvyšné sú na elektronickom médiu) je vidieť v prílohe C.



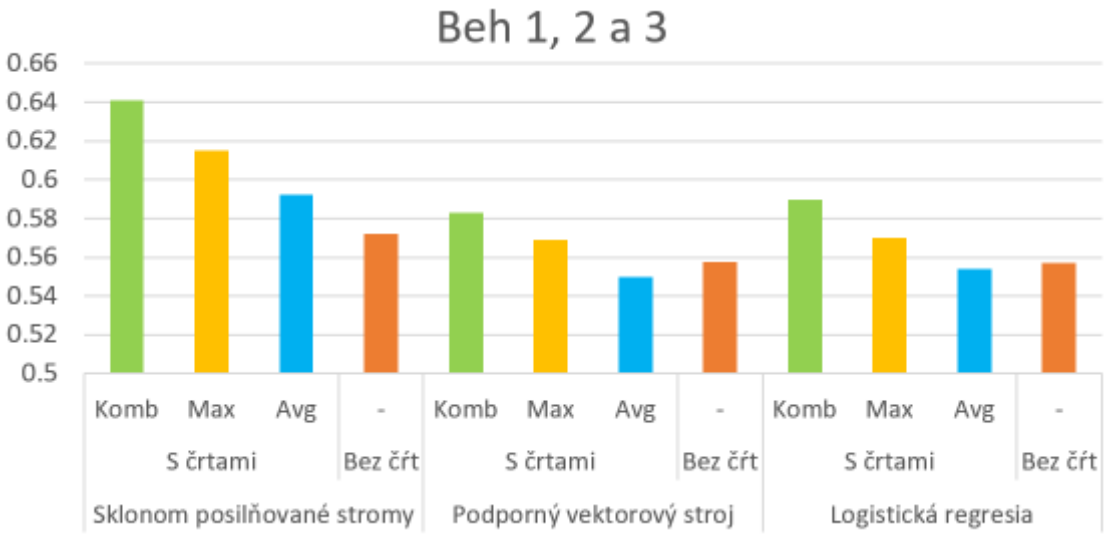
Obr. 4.4: *Graf výsledkov pre prvý beh. Stĺpec* Komb *reprezentuje výsledok s použitím kombinovaného modelu. Stĺpce* Max *a* Avg *vyjadrujú maximálny a priemerný výsledok s použitím čŕt.*



Obr. 4.5: *Graf výsledkov pre druhý beh. Stĺpec* Komb *reprezentuje výsledok s použitím kombinovaného modelu. Stĺpce* Max *a* Avg *vyjadrujú maximálny a priemerný výsledok s použitím čŕt.*



Obr. 4.6: *Graf výsledkov pre tretí beh. Stĺpec* Komb *reprezentuje výsledok s použitím kombinovaného modelu. Stĺpce* Max *a* Avg *vyjadrujú maximálny a priemerný výsledok s použitím čŕt.*



Obr. 4.7: *Graf výsledkov pre spojené dáta z prvého, druhého a tretieho behu. Stĺpec* Komb *reprezentuje výsledok s použitím kombinovaného modelu. Stĺpce* Max *a* Avg *vyjadrujú maximálny a priemerný výsledok s použitím čŕt.*

Aj v rámci týchto dát je vidieť najväčšie zlepšenie úspešnosti pri použití pravdepodobnostných čŕt klasifikátorom *sklonom posilňovaných stromov*. Vysoká úspešnosť kombinovaných modelov môže vznikať kvôli tomu, že dané úlohy sú od seba odlišné a tým pádom sú pre rôzne úlohy vhodné iné kombinácie vstupných čŕt použitých na trénovanie Skrytých Markovovych modelov.

5 Zhodnotenie

V tejto práci sme preskúmali možnosti pravdepodobnostného modelovania postupnosti pohľadov očí na vytvorenie nových čŕt, ktoré by v sebe obsahovali temporálnu informáciu pohľadu. Vytvorené črty používame na zlepšenie výsledkov štandardných modelov, ktoré spracovávajú dáta z pohľadu používateľa ako agregované dáta. Na riešenie tejto úlohy sme vytvorili základnú architektúru, ktorá využíva Skryté Markovove modely natrénované pomocou vstupných čŕt, ktoré vznikajú transformáciou postupnosti pohľadu používateľa (fixácie a sakády). Nové črty sa vytvárajú použitím pravdepodobnosti vygenerovania (vierohodnosť) sekvencie natrénovaními modelmi. Ukážku architektúry navrhovanej metódy je vidieť na obrázku 3.3.

Na zistenie či navrhované riešenie má zmysel a či dokáže zlepšiť úspešnosť, sme spravili základné vyhodnotenie dát na základe porovnania klasifikátorov, ktoré používajú navrhnuté črty s tými, ktoré navrhnuté črty nepoužívajú a používajú len agregované črty, ktoré boli prebrané z článku Simola et al. [2008] a čiastočne z článku od Coutrot et al. [2017]. Na vyhodnotenie sme použili dáta z domény čítania štruktúrovaného textu (zdrojového kódu) kvôli dostupnosti dátových sád. Na vyhodnotenie sme použili viacero klasifikátorov, ktoré reprezentujú často používané klasifikátory v oblasti spracovania pohľadu.

Na základe výsledkov spomenutých v časti 4.4 je vidieť najväčšie zlepšenie pre klasifikátor *Sklonom posilňovaných stromov* so zlepšením medzi najlepším modelom, ktorý používa navrhnuté črty (kombinovaný model) a modelom bez použitia čŕt o 5% až 7% v metrike F1. Vysoká úspešnosť kombinovaných modelov (opísané v časti 3.7) oproti iným modelom používajúcich len jednu kombináciu vstupných čŕt môže byť spôsobená odlišnosťou úloh.

V rámci práce sme spravili všeobecnú analýzu kombinácií vstupných čŕt pomocou otestovania rôznych kombinácií parametrov mriežkovým prehľadáva-

ním (angl. *grid-search*), ktorej výsledky sme využili pri vyhodnocovaní menšej dátovej sady. Taktiež sme spravili analýzu kombinácií pre rôzne kategórie úloh obsahujúcich kód na základe rozdelenia podľa oblasti zložitosti (aritmetická, logická, cyklová alebo argumentová zložitosť) v časti 4.3.2. Na základe tejto analýzy sme zistili, že najlepšiu úspešnosť pri väčšine kategórií má vstupná črta dĺžky fixácií (trvanie). Vhodnosť použitia ostatných vstupných čŕt súvisí s kategóriou zložitosti zvoleného kódu.

Hlavnou nevýhodou navrhovaného riešenia je časová náročnosť na tvorbu čŕt (niekoľko násobne dlhšie ako je vytvorenie agregovaných čŕt), čo znamená, že riešenie nemusí byť vhodné použiť pri každom druhu úlohy (ak je väčší dôraz na čas ako na výsledky). Taktiež navrhované riešenie aktuálne neobsahuje možnosť postupného učenia (angl. *online*). Navrhnuté riešenie by malo byť použiteľné aj na dátach z iných domén súvisiacich s pohľadom používateľa, ale v rámci časových obmedzení bolo otestované iba na doméne čítania štruktúrovaného textu.

Na základe doménovej analýzy sme zistili, že dôležitú úlohu pri porovnávaní postupností pohľadu majú nielen často používané súradnice fixácií a dĺžka sakád, ale aj ďalšie parametre ako uhol sakády alebo trvanie fixácie. Medzi oblasti ďalšieho výskumu môže patriť využitie týchto parametrov pri ďalších metódach porovnávania postupností pohľadu spomínaných v analýze v časti 2.6.1.

Literatúra

Nicola C Anderson, Fraser Anderson, Alan Kingstone, and Walter F Bischof. A comparison of scanpath comparison methods. *Behavior research methods*, 47(4):1377–1392, 2015.

Davide Bacciu, Paulo JG Lisboa, Alessandro Sperduti, and Thomas Villmann. Probabilistic modeling in machine learning. In *Springer Handbook of Computational Intelligence*, pages 545–575. Springer, 2015.

Roman Bednarik. Expertise-dependent visual attention strategies develop over time during debugging with multiple code representations. *International Journal of Human-Computer Studies*, 70(2):143–155, 2012.

Martin Böhme, André Meyer, Thomas Martinetz, and Erhardt Barth. Remote eye tracking: State of the art and directions for future development. In *Proc. of the 2006 Conference on Communication by Gaze Interaction (COGAIN)*, pages 12–17, 2006.

Teresa Busjahn, Carsten Schulte, and Andreas Busjahn. Analysis of code reading to gain more insight in program comprehension. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*, pages 1–9. ACM, 2011.

Teresa Busjahn, Roman Bednarik, and Carsten Schulte. What influences dwell time during source code reading?: analysis of element type and frequency as factors. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 335–338. ACM, 2014.

Teresa Busjahn, Roman Bednarik, Andrew Begel, Martha Crosby, James H Paterson, Carsten Schulte, Bonita Sharif, and Sascha Tamm. Eye movements in code reading: Relaxing the linear order. In *Program Comprehension (ICPC), 2015 IEEE 23rd International Conference on*, pages 255–265. IEEE, 2015.

Ronald P Carver. *Reading rate: A review of research and theory.* Academic Press, 1990.

Tim Chuk, Antoni B Chan, and Janet H Hsiao. Understanding eye movements in face recognition using hidden markov models. *Journal of vision*, 14(11): 8–8, 2014.

Antoine Coutrot and Nathalie Guyader. How saliency, faces, and sound influence gaze in dynamic social scenes. *Journal of vision*, 14(8):5–5, 2014.

Antoine Coutrot, Nicola Binetti, Charlotte Harrison, Isabelle Mareschal, and Alan Johnston. Face exploration dynamics differentiate men and women. *Journal of vision*, 16(14):16–16, 2016.

Antoine Coutrot, Janet H Hsiao, and Antoni B Chan. Scanpath modeling and classification with hidden markov models. *Behavior Research Methods*, pages 1–18, 2017.

Emanuele Coviello, Antoni B Chan, and Gert RG Lanckriet. Clustering hidden markov models with variational hem. *The Journal of Machine Learning Research*, 15(1):697–747, 2014.

Martha E Crosby and Jan Stelovsky. How do we read algorithms? a case study. *Computer*, 23(1):25–35, 1990.

Claudia Ehmke and Stephanie Wilson. Identifying web usability problems from eye-tracking data. In *Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI... but not as we know it-Volume 1*, pages 119–128. British Computer Society, 2007.

Michael Hansen, Robert L Goldstone, and Andrew Lumsdaine. What makes code hard to understand? *arXiv preprint arXiv:1304.5257*, 2013.

Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259, 1998.

RJ Jacobs. Visual resolution and contour interaction in the fovea and periphery. *Vision research*, 19(11):1187–1195, 1979.

Ahmad Jbara and Dror G Feitelson. How programmers read regular code: a controlled experiment using eye tracking. *Empirical software engineering*, 22(3):1440–1477, 2017.

Tony Jebara, Yingbo Song, and Kapil Thadani. Spectral clustering and embedding with hidden markov models. In *European Conference on Machine Learning*, pages 164–175. Springer, 2007.

Omid Kardan, Marc G Berman, Grigori Yourganov, Joseph Schmidt, and John M Henderson. Classifying mental states from eye movements during scene viewing. *Journal of Experimental Psychology: Human Perception and Performance*, 41(6):1502, 2015.

Enkelejda Kasneci, Gjergji Kasneci, Thomas C Kübler, and Wolfgang Rosenstiel. The applicability of probabilistic methods to the online recognition of fixations and saccades in dynamic scenes. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 323–326. ACM, 2014.

Dmitry Kit and Brian Sullivan. Classifying mobile eye tracking data with hidden markov models. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, pages 1037–1040. ACM, 2016.

Jeff Klingner, Rakshit Kumar, and Pat Hanrahan. Measuring the task-evoked pupillary response with a remote eye tracker. In *Proceedings of the 2008 symposium on Eye tracking research & applications*, pages 69–72. ACM, 2008.

Kathryn Koehler, Fei Guo, Sheng Zhang, and Miguel P Eckstein. What do saliency models predict? *Journal of vision*, 14(3):14–14, 2014.

Sepp Kollmorgen, Nora Nortmann, Sylvia Schröder, and Peter König. Influence of low-level stimulus features, task dependent factors, and spatial

biases on overt visual attention. *PLoS computational biology*, 6(5):e1000791, 2010.

Thomas C Kübler, Colleen Rothe, Ulrich Schiefer, Wolfgang Rosenstiel, and Enkelejda Kasneci. Subsmatch 2.0: Scanpath comparison and classification based on subsequence frequencies. *Behavior research methods*, 49(3):1048–1064, 2017.

Dmitry Lagun, Cecelia Manzanares, Stuart M Zola, Elizabeth A Buffalo, and Eugene Agichtein. Detecting cognitive impairment by eye movement analysis using automatic classification algorithms. *Journal of neuroscience methods*, 201(1):196–203, 2011.

Olivier Le Meur and Thierry Baccino. Methods for comparing scanpaths and saliency maps: strengths and weaknesses. *Behavior research methods*, 45 (1):251–266, 2013.

Olivier Le Meur and Zhi Liu. Saccadic model of eye movements for free-viewing condition. *Vision research*, 116:152–164, 2015.

Clare A McGrory and DM Titterington. Variational bayesian analysis for hidden markov models. *Australian & New Zealand Journal of Statistics*, 51 (2):227–244, 2009.

Paul A Orlov. Primary investigation of applying hidden markov models for eye movements in source code reading. *Eye Movements in Programming Education II: Analyzing the Novice's Gaze*, page 18, 2015.

Jose P Ossandon, Selim Onat, and Peter Koenig. Spatial biases in viewing behavior. *Journal of vision*, 14(2):20–20, 2014.

Keith Rayner. Eye movements in reading and information processing: 20 years of research. *Psychological bulletin*, 124(3):372, 1998.

Keith Rayner and Susan A Duffy. Lexical complexity and fixation times in reading: Effects of word frequency, verb complexity, and lexical ambiguity. *Memory & cognition*, 14(3):191–201, 1986.

Jaana Simola, Jarkko Salojärvi, and Ilpo Kojo. Using hidden markov model to uncover processing states from eye movements in information search tasks. *Cognitive Systems Research*, 9(4):237–251, 2008.

Ben Steichen, Giuseppe Carenini, and Cristina Conati. User-adaptive information visualization: using eye gaze data to infer visualization tasks and user cognitive abilities. In *Proceedings of the 2013 international conference on Intelligent user interfaces*, pages 317–328. ACM, 2013.

Ben Steichen, Cristina Conati, and Giuseppe Carenini. Inferring visualization task properties, user performance, and user cognitive abilities from eye gaze data. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 4(2):11, 2014.

Rachel Turner, Michael Falcone, Bonita Sharif, and Alina Lazar. An eye-tracking study assessing the comprehension of c++ and python source code. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 231–234. ACM, 2014.

Hidetake Uwano, Masahide Nakamura, Akito Monden, and Ken-ichi Matsumoto. Analyzing individual performance of source code review using reviewers' eye movement. In *Proceedings of the 2006 symposium on Eye tracking research & applications*, pages 133–140. ACM, 2006.

Susan Wiedenbeck, Vikki Fix, and Jean Scholtz. Characteristics of the mental representations of novice and expert programmers: an empirical study. *International Journal of Man-Machine Studies*, 39(5):793–812, 1993.

Alfred L Yarbus. Eye movements during perception of complex objects. In *Eye movements and vision*, pages 171–211. Springer, 1967.

Príloha A: Plán práce a jeho vyhodnotenie

V tabuľke 5.1 je vidieť stanovený plán pre diplomovú prácu.

Tabuľka 5.1: *Plán práce na diplomovej práci v priebehu týždňov a predmetov.*

Predmet	**Týždeň**	**Úlohy**
DP1	1. - 4. týždeň	Hladanie článkov k súvisiacim oblastiam Stav implementacií HMM a podporných algoritmov
DP1	5. - 7. týždeň	Čítanie a konzultácia článkov Návrh metódy Vytvorenie a konzultácia osnovy správy
DP1	8. - 11. týždeň	Písanie správy Hladanie nových článkov Formulácia cieľov a ďalších výskumných otázok
DP1	12. - 13. týždeň	Dopísanie priebežnej spŕavy a kontrola Odovzdanie správy
DP2	1. - 4. týždeň	Testovanie knižníc na učenie HMM Implementácie podporných algoritmov pre HMM Spracovanie dát
DP2	5. - 8. týždeň	Tvorba architektúry pre riešenie Písanie správy Plánovanie experimentu v závislosti od dát
DP2	9. - 12. týždeň	Tvorba architektúry pre riešenie Trénovanie a vyhodnocovanie modelu Dopísanie priebežnej spŕavy a kontrola
DP3	1. - 4. týždeň	Vylepšovanie modelu Článok na IITSRC Písanie správy

DP3	5. - 8. týždeň	Vylepšovanie modelu Vyhodnocovanie výsledkov Písanie správy
DP3	9. - 13. týždeň	Vylepšovanie modelu Finálne vyhodnotenie Dokončovanie a odovzdanie diplomovej práce

Vyhodnotenie plánu pre DP1

Poznámky k plneniu plánu pre DP1:

- Články som čítal a konzultoval priebežne počas celého semestra.
- V rámci 3. týždňa som konzultoval nájdené implementácie pre HMM.
- Návrh metódy ako aj výskumné otázky sa vyvíjali priebežne aj mimo 5. - 11. týždňa.
- Osnovu správy som vytvoril v 7. týždni a konzultoval vo 8. týždni.
- V rámci 11. týždňa som spravil základnú analýzu dát.
- Správu som dopísal vo 12. týždni.

Podarilo sa mi splniť úlohy, ktoré som si naplánoval na predmet DP1.

Vyhodnotenie plánu pre DP2

Poznámky k plneniu plánu pre DP2:

- Základné spracovanie dát a tvorbu agregovaných čŕt som stihol do 4. týždňa.
- Základnú verziu celkovej architektúry som dokončil v 6. týždni a pridal aj vyhodnocovanie pomocou modelu vo 8.-9. týždni.

- Ako experimentálne overenie mojej metódy spracovávam získanú väčšiu dátovú sadu od 10. týždňa, ktorá je opísaná v 4.1.2.
- Parametre a pravdepodobnostné modely som vyhodnocoval v 9.-12. týždni.
- Správu som dopísal vo 12. týždni.

Podarilo sa mi splniť úlohy, ktoré som si naplánoval na predmet DP2.

Vyhodnotenie plánu pre DP3

Poznámky k plneniu plánu pre DP3:

- Do 3. týždňa som sa prímarne venoval písaniu článku na IIT.SRC a získavaniu ďalších dátových sád
- V 4. a 5. týždeň som vylepšoval model cez rozšírenie o optimalizáciu čŕt na základe pravdepodobnostnej hranice
- Od 6. po 10. týždeň som pripravoval robustnú verziu metódy a vyhodnocoval metódu na školských počítačoch
- V 7. a 8. týždni som rozširoval IITSRC správu (príloha D) o pripomienky a pracoval na tvorbe posteru (príloha E)
- V 10. týždni som rozšíril model o možnosť použitia viacerých kombinácií na jedných dátach
- V 9. až 11. týždni som spravil finálne vyhodnotenie modelu
- Hlavnú časť písania a kontroly správy som začal robiť od 9. týždňa

Okrem trošku neskorého začania písania správy sa mi podarilo splniť úlohy, ktoré som si naplánoval na predmet DP3.

Príloha B: Technická dokumentácia

Inštalačná príručka a použitie

Na použitie implementovaného riešenia je potrebné mať nainštalovaný programovací jazyk Python verziu 3.6 a cestu k Pythonu v premenných prostredia. Následne je potrebné v adresári s aplikáciou spustiť nasledovné príkazy cez príkazový riadok:

1. pip install –upgrade pip
2. pip install -r requirements.txt

Na spustenie jupyter notebookov v adresári je potrebné použiť nasledujúci príkaz a následne si v prehliadači vybrať súbor na otvorenie:

1. jupyter lab

Na použitie vytvoreného riešenia v inom Python kóde:

1. Premiestniť priečinok *FeatureGeneration* do cieľového priečinku s kódom
2. Importovať v kóde modul *FeatureGeneration.hmm_features*

Opis verejných funkcií z modulu a postupu práce s ním je v používateľskej príručke.

Na pridanie čŕt do inej dátovej sady obsahujúcej sekvencie fixácií bez implementácie vlastného riešenia je treba postupovať podľa nasledujúcich krokov:

1. V adresári s aplikáciou spustiť príkaz *jupyter lab*
2. Otvoriť notebook *AddHMMFeatures*

3. Prispôsobiť konfiguráciu

4. Spustiť bunky v notebooku (po riadkoch cez skratku - Shift + Enter, alebo všetky cez príkaz *Run all cells*)

Používateľská príručka

Postup práce

1. Vytvorenie novej inštancie triedy *ProbabilisticFeatureGeneration* s vybranou kombináciou vstupných čŕt (povinné), počtom skrytých stavov (povinné) a kombinácií pre úlohu (nepovinné)

2. Natrénovanie Skrytých Markovovych modelov cez metódu *train_models* alebo *train_models_with_optimization*

3. Vytvorenie nových čŕt nad vloženou dátovou sadou s metódou *generate_features*

Dokumentácia verejných funkcií

Konštruktor pre ProbabilisticFeatureGeneration:

```
__init__(self, fix_x=True, fix_y=True, saccadic_length=True, »
  » abs_angle=True, rel_angle=True, duration=True, »
  » number_of_states=3, split_combinations=None)
```

Trieda ProbabilisticFeatureGeneration slúži na tvorbu pravdepodobnostných čŕt pre dátovú sadu po natrénovaní Skrytých Markovovych modelov.

Parametre:

- *fix_x*: bool - použitie x-ovej súradnice fixácie ako vstupnej črty pri trénovaní

- *fix_y*: bool - použitie y-ovej súradnice fixácie ako vstupnej črty pri trénovaní

- *saccadic_length*: bool - použitie dĺžky sakády ako vstupnej črty pri trénovaní
- *abs_angle*: bool - použitie absolútneho uhla ako vstupnej črty pri trénovaní
- *rel_angle*: bool - použitie relatívneho uhla ako vstupnej črty pri trénovaní
- *duration*: bool - použitie trvania fixácie ako vstupnej črty pri trénovaní
- *number_of_states*: int - počet skrytých stavov vo vytvorených Skrytých Markovovych modeloch

Poznámky:

- Aspoň jeden parameter z *fix_x*, *fix_y*, *saccadic_length*, *abs_angle*, *rel_angle* a *duration* musí byť nastavený na hodnotu *True*.

train_models:

```
train_models(self, basic_df, aggregation_columns, y_column, »
  » movement_column='Movement', fix_x_column='FixationScreenX', »
  » fix_y_column='FixationScreenY', duration_column='Duration', »
  » split_column=None)
```

Natrénuje Skryté Markovove modely z danej dátovej sady obsahujúcej sekvencie fixácií.

Parametre:

- *basic_df*: pandas Dataframe - dátová sada so základnými dátami obsahujúcimi sekvencie fixácií
- *aggregation_columns*: list - názvy stĺpcov z dátovej sady *basic_df*, ktoré slúžia na rozdelenie sekvencií fixácií
- *y_column*: str - názov stĺpca s predikovanými hodnotami (používa sa na zistenie počtu rôznych tried) v dátovej sade *basic_df*

- *movement_column*: str - názov stĺpca, ktorý obsahuje o aký druh pohybu ide (obsahuje hodnotu *Fixation*) v dátovej sade *basic_df*
- *fix_x_column*: str - názov stĺpca, ktorý obsahuje x-ovú súradnicu fixácií v dátovej sade *basic_df*
- *fix_y_column*: str - názov stĺpca, ktorý obsahuje y-ovú súradnicu fixácií v dátovej sade *basic_df*
- *duration_column*: str - názov stĺpca, ktorý obsahuje trvanie fixácií v dátovej sade *basic_df*
- *split_column*: str, default=None - ak je definovaný (nerovná sa None), tak viacero modelov bude vytvorených a použitých na tvorbu čŕt. Názov stĺpca, na základe ktorého hodnôt sa rozdelí dátová sada (rozdelenie na základe vizuálneho stimulu). Príklad použitia - úlohy s rôznym rozložením

train_models_with_optimization:

```
train_models_with_optimization(self, basic_df, »
  » aggregation_columns, y_column, movement_column='Movement', »
  » fix_x_column='FixationScreenX', fix_y_column='»
  » FixationScreenY', duration_column='Duration', split_column=»
  » None, validation_split=0.2)
```

Natrénuje Skryté Markovove modely z danej dátovej sady obsahujúcej sekvencie fixácií.

Parametre:

- *basic_df*: pandas Dataframe - dátová sada so základnými dátami obsahujúcimi sekvencie fixácií
- *aggregation_columns*: list - názvy stĺpcov z dátovej sady *basic_df*, ktoré slúžia na rozdelenie sekvencií fixácií
- *y_column*: str - názov stĺpca s predikovanými hodnotami (používa sa na zistenie počtu rôznych tried) v dátovej sade *basic_df*

- *movement_column*: str - názov stĺpca, ktorý obsahuje o aký druh pohybu ide (obsahuje hodnotu *Fixation*) v dátovej sade *basic_df*
- *fix_x_column*: str - názov stĺpca, ktorý obsahuje x-ovú súradnicu fixácií v dátovej sade *basic_df*
- *fix_y_column*: str - názov stĺpca, ktorý obsahuje y-ovú súradnicu fixácií v dátovej sade *basic_df*
- *duration_column*: str - názov stĺpca, ktorý obsahuje trvanie fixácií v dátovej sade *basic_df*
- *split_column*: str, default=None - ak je definovaný (nerovná sa None), tak viacero modelov bude vytvorených a použitých na tvorbu čŕt. Názov stĺpca, na základe ktorého hodnôt sa rozdelí dátová sada (rozdelenie na základe vizuálneho stimulu). Príklad použitia - úlohy s rôznym rozložením
- *validation_split*: float, default=0.2 - Množstvo dát z *basic_df*, ktoré budú použité ako validačná vzorka na výpočet pravdepodobnostnej hranice.

generate_features:

```
generate_features(aggregated_df, basic_df, aggregation_columns, »
  » y_column, movement_column='Movement', fix_x_column='»
  » FixationScreenX', fix_y_column='FixationScreenY', »
  » duration_column='Duration', split_column=None, strict=False)
```

Vygeneruje pravdepodobnostné črty pre dátovú sadu po natrénovaní Skrytých Markovovych modelov.

Parametre:

- *aggregated_df*: pandas Dataframe - dátová sada s agregovanými črtami
- *basic_df*: pandas Dataframe - dátová sada so základnými dátami (dáta na základe ktorých bol vytvorený *aggregated_df*)

- *aggregation_columns*: list - názvy stĺpcov z dátovej sady *basic_df*, ktoré boli použité na tvorbu dátovej sady *aggregated_df*.
- *y_column*: str - názov stĺpca s predikovanými hodnotami (používa sa na zistenie počtu rôznych tried) v dátovej sade *basic_df*
- *movement_column*: str - názov stĺpca, ktorý obsahuje o aký druh pohybu ide (obsahuje hodnotu *Fixation*) v dátovej sade *basic_df*
- *fix_x_column*: str - názov stĺpca, ktorý obsahuje x-ovú súradnicu fixácií v dátovej sade *basic_df*
- *fix_y_column*: str - názov stĺpca, ktorý obsahuje y-ovú súradnicu fixácií v dátovej sade *basic_df*
- *duration_column*: str - názov stĺpca, ktorý obsahuje trvanie fixácií v dátovej sade *basic_df*
- *split_column*: str, default=None - ak je definovaný (nerovná sa None), tak viacero modelov bude vytvorených a použitých na tvorbu čŕt. Názov stĺpca, na základe ktorého hodnôt sa rozdelí dátová sada (rozdelenie na základe vizuálneho stimulu). Príklad použitia - úlohy s rôznym rozložením
- *strict*: bool - vyjadruje či sa použije prísna verzia. Prísna verzia nastaví všetky črty na základné hodnoty, na rozdiel od nastavenia iba črty predikovanej triedy v prípade ak nie je dosiahnutá pravdepodobnostná hranica alebo je použitý Null object.

Vracia:

- *aggregated_df_HMM*: pandas Dataframe - dátová sada *aggregated_df* rozšírená o nové črty

Príloha C: Výsledky pre rôzne kombinácie

Výsledky pre 1. beh usporiadané podľa úspešnosti so všeobecným klasifikátorom. Ostatné výsledky sú uložené na elektronickom médiu.

Sklonom posilňované stromy:

Fix. X	**Fix. Y**	**Dĺžka sak.**	**Abs. uhol**	**Rel. uhol**	**Dĺžka fix.**	**# stavov**	**F1 gen**	**F1 task**
comb	comb	comb	comb	comb	comb	comb	0.672	0.673
nie	nie	nie	nie	nie	áno	2	0.667	0.68
nie	nie	nie	áno	áno	nie	3	0.665	0.681
nie	nie	nie	áno	nie	nie	4	0.664	0.689
nie	nie	áno	nie	áno	nie	3	0.663	0.68
nie	nie	áno	áno	áno	áno	3	0.662	0.685
nie	nie	áno	áno	áno	nie	3	0.658	0.681
nie	áno	nie	nie	áno	nie	4	0.657	0.684
nie	nie	nie	áno	áno	nie	4	0.656	0.675
nie	áno	nie	nie	nie	nie	4	0.655	0.678
nie	nie	nie	nie	áno	nie	3	0.655	0.682
nie	nie	nie	nie	nie	áno	4	0.654	0.682
áno	nie	áno	nie	áno	nie	3	0.654	0.682
áno	nie	nie	áno	áno	nie	4	0.653	0.684
áno	nie	áno	áno	áno	nie	3	0.652	0.677
nie	nie	nie	áno	nie	nie	3	0.652	0.685
nie	nie	nie	nie	áno	nie	4	0.651	0.691
nie	nie	áno	nie	nie	nie	4	0.649	0.681
nie	nie	áno	nie	nie	áno	3	0.649	0.687
nie	nie	áno	nie	nie	nie	3	0.648	0.683
áno	nie	áno	nie	áno	áno	3	0.648	0.684
nie	áno	nie	áno	áno	áno	3	0.648	0.678
áno	nie	áno	áno	áno	áno	3	0.648	0.678
nie	áno	nie	nie	áno	áno	3	0.647	0.679
áno	nie	nie	nie	nie	nie	4	0.646	0.677
áno	nie	áno	nie	áno	áno	4	0.646	0.681
nie	nie	áno	áno	áno	áno	4	0.646	0.687
nie	áno	nie	áno	áno	nie	3	0.645	0.678
áno	nie	nie	nie	nie	nie	3	0.645	0.681
áno	nie	nie	áno	áno	nie	3	0.645	0.68
áno	áno	nie	áno	áno	áno	4	0.645	0.686
nie	áno	nie	nie	nie	nie	3	0.645	0.679

Fix. X	Fix. Y	Dĺžka sak.	Abs. uhol	Rel. uhol	Dĺžka fix.	# stavov	F1 gen	F1 task
nie	áno	nie	nie	áno	nie	3	0.643	0.68
nie	nie	áno	nie	áno	nie	4	0.642	0.681
áno	áno	nie	áno	áno	áno	3	0.642	0.685
nie	nie	nie	áno	áno	áno	3	0.641	0.684
áno	áno	nie	áno	nie	nie	4	0.641	0.684
nie	áno	áno	áno	nie	áno	3	0.641	0.68
áno	nie	nie	nie	áno	nie	4	0.64	0.684
áno	nie	nie	áno	nie	áno	4	0.64	0.694
nie	nie	áno	nie	áno	áno	4	0.639	0.683
áno	nie	nie	áno	nie	nie	3	0.639	0.678
nie	áno	áno	nie	áno	nie	3	0.638	0.685
áno	áno	nie	nie	nie	áno	3	0.638	0.68
áno	nie	áno	nie	nie	áno	3	0.638	0.676
nie	nie	nie	áno	áno	áno	4	0.638	0.684
nie	áno	nie	áno	áno	nie	4	0.637	0.681
áno	áno	áno	áno	áno	áno	4	0.637	0.679
nie	nie	áno	áno	áno	nie	4	0.637	0.682
nie	áno	áno	áno	áno	áno	3	0.636	0.675
nie	nie	nie	áno	nie	áno	4	0.636	0.681
nie	áno	áno	nie	nie	áno	3	0.636	0.685
nie	nie	nie	áno	nie	áno	3	0.636	0.685
áno	áno	nie	nie	nie	nie	3	0.635	0.687
nie	áno	áno	nie	áno	áno	3	0.635	0.685
áno	nie	áno	nie	nie	áno	4	0.635	0.678
nie	áno	nie	áno	nie	áno	3	0.634	0.693
nie	áno	nie	nie	nie	áno	3	0.634	0.678
áno	nie	nie	áno	nie	áno	3	0.633	0.679
nie	nie	nie	nie	áno	áno	4	0.633	0.679
áno	nie	nie	nie	áno	nie	3	0.633	0.688
nie	áno	nie	áno	nie	nie	4	0.633	0.681
áno	áno	áno	áno	áno	nie	3	0.632	0.68
nie	nie	áno	áno	nie	nie	4	0.632	0.679
nie	áno	nie	áno	nie	nie	3	0.632	0.676
nie	áno	áno	nie	áno	nie	4	0.632	0.682
áno	nie	nie	nie	áno	áno	4	0.631	0.689
áno	áno	áno	nie	áno	nie	4	0.631	0.686
nie	áno	áno	áno	nie	nie	3	0.631	0.685
nie	áno	áno	nie	nie	áno	4	0.631	0.687

Fix. X	Fix. Y	Dĺžka sak.	Abs. uhol	Rel. uhol	Dĺžka fix.	# stavov	F1 gen	F1 task
áno	áno	nie	áno	nie	nie	3	0.631	0.678
áno	áno	nie	áno	nie	áno	3	0.631	0.685
nie	áno	nie	áno	áno	áno	4	0.63	0.682
áno	áno	áno	áno	áno	nie	4	0.63	0.685
áno	áno	nie	áno	áno	nie	3	0.63	0.687
áno	áno	áno	áno	nie	áno	3	0.63	0.686
áno	áno	áno	nie	áno	nie	3	0.63	0.676
áno	áno	nie	áno	áno	nie	4	0.63	0.683
áno	áno	nie	nie	áno	nie	4	0.63	0.682
áno	áno	nie	áno	nie	áno	4	0.63	0.68
áno	nie	nie	áno	áno	áno	4	0.629	0.686
áno	áno	nie	nie	nie	nie	4	0.629	0.682
áno	áno	nie	nie	áno	nie	3	0.629	0.679
áno	nie	nie	nie	nie	áno	4	0.629	0.689
nie	áno	áno	nie	áno	áno	4	0.629	0.676
nie	nie	áno	áno	nie	áno	3	0.629	0.683
nie	nie	áno	nie	nie	áno	4	0.629	0.679
áno	áno	nie	nie	nie	áno	4	0.629	0.68
nie	áno	áno	áno	áno	nie	4	0.629	0.686
nie	áno	áno	áno	nie	áno	4	0.628	0.683
áno	nie	áno	nie	nie	nie	3	0.628	0.673
nie	áno	áno	nie	nie	nie	4	0.628	0.688
nie	áno	áno	áno	áno	nie	3	0.628	0.689
áno	áno	nie	nie	áno	áno	4	0.628	0.682
nie	áno	nie	nie	áno	áno	4	0.628	0.675
áno	nie	áno	áno	áno	áno	4	0.628	0.678
áno	áno	áno	áno	nie	nie	4	0.627	0.677
nie	nie	áno	áno	nie	nie	3	0.627	0.679
nie	nie	nie	nie	áno	áno	3	0.627	0.683
nie	áno	nie	áno	nie	áno	4	0.627	0.681
áno	nie	nie	nie	áno	áno	3	0.627	0.68
áno	nie	áno	nie	nie	nie	4	0.626	0.674
áno	áno	áno	nie	nie	nie	3	0.626	0.678
nie	áno	nie	nie	nie	áno	4	0.625	0.682
áno	áno	áno	nie	áno	áno	3	0.625	0.692
áno	áno	nie	nie	áno	áno	3	0.625	0.688
áno	nie	áno	áno	nie	áno	3	0.624	0.68
nie	áno	áno	áno	nie	nie	4	0.624	0.683

Fix. X	Fix. Y	Dĺžka sak.	Abs. uhol	Rel. uhol	Dĺžka fix.	# stavov	F1 gen	F1 task
áno	áno	áno	áno	áno	áno	3	0.624	0.681
áno	nie	áno	áno	nie	áno	4	0.623	0.688
áno	áno	áno	áno	nie	áno	4	0.622	0.684
áno	nie	nie	áno	áno	áno	3	0.622	0.685
áno	nie	áno	áno	áno	nie	4	0.622	0.68
nie	áno	áno	áno	áno	áno	4	0.622	0.679
áno	nie	nie	áno	nie	nie	4	0.622	0.69
áno	áno	áno	nie	nie	áno	3	0.622	0.681
áno	nie	áno	nie	áno	nie	4	0.621	0.686
áno	áno	áno	nie	áno	áno	4	0.62	0.681
nie	áno	áno	nie	nie	nie	3	0.618	0.686
áno	áno	áno	áno	nie	nie	3	0.618	0.681
áno	áno	áno	nie	nie	nie	4	0.614	0.681
áno	nie	áno	áno	nie	nie	3	0.614	0.677
-	-	-	-	-	-	-	0.613	0.683

Podporný vektorový stroj:

Fix. X	Fix. Y	Dĺžka sak.	Abs. uhol	Rel. uhol	Dĺžka fix.	# stavov	F1 gen	F1 task
comb	comb	comb	comb	comb	comb	comb	0.644	0.714
áno	nie	áno	nie	áno	nie	3	0.632	0.714
nie	nie	áno	áno	áno	nie	3	0.632	0.715
nie	nie	áno	nie	áno	nie	3	0.631	0.718
nie	nie	nie	áno	áno	nie	3	0.631	0.715
nie	nie	nie	áno	nie	nie	3	0.63	0.716
nie	áno	nie	nie	áno	nie	4	0.629	0.715
áno	nie	áno	áno	áno	nie	3	0.629	0.711
nie	nie	nie	áno	áno	nie	4	0.628	0.715
áno	nie	nie	áno	áno	nie	4	0.627	0.708
nie	nie	nie	áno	nie	nie	4	0.627	0.718
nie	nie	nie	nie	áno	nie	3	0.625	0.712
-	-	-	-	-	-	-	0.622	0.718
nie	áno	nie	nie	nie	nie	4	0.613	0.711
nie	nie	nie	nie	nie	áno	2	0.613	0.714

Fix. X	**Fix. Y**	**Dĺžka sak.**	**Abs. uhol**	**Rel. uhol**	**Dĺžka fix.**	**# stavov**	**F1 gen**	**F1 task**
nie	nie	nie	nie	nie	áno	4	0.607	0.712
nie	nie	nie	nie	nie	áno	3	0.594	0.718

Logistická regresia:

Fix. X	**Fix. Y**	**Dĺžka sak.**	**Abs. uhol**	**Rel. uhol**	**Dĺžka fix.**	**# stavov**	**F1 gen**	**F1 task**
comb	comb	comb	comb	comb	comb	comb	0.644	0.715
nie	áno	nie	nie	áno	nie	4	0.636	0.718
nie	nie	nie	áno	nie	nie	4	0.633	0.719
nie	nie	nie	áno	áno	nie	4	0.633	0.722
áno	nie	áno	nie	áno	nie	3	0.632	0.717
nie	nie	nie	áno	áno	nie	3	0.631	0.719
nie	nie	áno	nie	áno	nie	3	0.63	0.719
nie	nie	áno	áno	áno	nie	3	0.63	0.717
áno	nie	áno	áno	áno	nie	3	0.629	0.716
áno	nie	nie	áno	áno	nie	4	0.628	0.713
nie	nie	nie	nie	áno	nie	3	0.622	0.715
-	-	-	-	-	-	-	0.621	0.72
nie	nie	nie	nie	nie	áno	2	0.619	0.719
nie	áno	nie	nie	nie	nie	4	0.617	0.714
nie	nie	nie	nie	nie	áno	4	0.61	0.715
nie	nie	nie	nie	nie	áno	3	0.596	0.72

Riadok označený slovom *comb* obsahuje výsledok pre kombinovaný model. Riadok označený znakom - obsahuje výsledok bez použitia navrhnutých čŕt.

Kombinácie v doménovej analýze

V tabuľkách stĺpce **Fix. X**, **Fix. Y**, **Dĺžka sak.**, **Abs. uhol**, **Rel. uhol** a **# stavov** (počet skrytých stavov) vyjadrujú použité parametre na trénovanie pravdepodobnostného modelu. Výsledky sú usporiadané od najväčšieho po najmenší.

Aritmetická zložitosť:

Fix. X	**Fix. Y**	**Dĺžka sak.**	**Abs. uhol**	**Rel. uhol**	**Dĺžka fix.**	**# stavov**	**Accuracy**
nie	nie	nie	áno	áno	áno	4	0.545
áno	áno	áno	nie	nie	nie	3	0.542
nie	áno	nie	nie	áno	nie	3	0.539
nie	áno	áno	áno	áno	nie	3	0.537
áno	nie	nie	nie	nie	áno	4	0.536
nie	áno	nie	áno	áno	áno	4	0.533
áno	nie	áno	áno	áno	nie	4	0.531
nie	nie	nie	nie	áno	áno	4	0.528
áno	nie	áno	nie	áno	nie	4	0.525
nie	áno	nie	nie	áno	áno	4	0.522

Logická zložitosť:

Fix. X	**Fix. Y**	**Dĺžka sak.**	**Abs. uhol**	**Rel. uhol**	**Dĺžka fix.**	**# stavov**	**Accuracy**
nie	nie	nie	áno	áno	áno	4	0.576
nie	nie	nie	áno	nie	áno	3	0.572
nie	nie	áno	nie	áno	áno	3	0.570
nie	nie	nie	nie	nie	áno	3	0.568
áno	nie	nie	nie	nie	áno	3	0.564
nie	nie	nie	nie	áno	áno	4	0.562
nie	nie	nie	áno	áno	nie	4	0.558
áno	nie	nie	áno	nie	áno	3	0.557
nie	áno	áno	nie	áno	nie	3	0.556
áno	nie	nie	nie	nie	áno	4	0.556

Zložitosť cyklov:

Fix. X	**Fix. Y**	**Dĺžka sak.**	**Abs. uhol**	**Rel. uhol**	**Dĺžka fix.**	**# stavov**	**Accuracy**
áno	áno	áno	áno	nie	áno	4	0.666
nie	áno	áno	áno	áno	áno	4	0.652
nie	áno	nie	nie	áno	áno	4	0.645
nie	áno	áno	áno	nie	áno	3	0.632
áno	nie	nie	nie	áno	áno	4	0.632
nie	nie	áno	áno	áno	áno	3	0.628
áno	nie	nie	áno	áno	áno	3	0.625
nie	áno	nie	áno	nie	áno	3	0.624
nie	áno	nie	áno	nie	áno	4	0.620
áno	áno	nie	áno	áno	áno	4	0.620

Zložitosť argumentov funkcií:

Fix. X	**Fix. Y**	**Dĺžka sak.**	**Abs. uhol**	**Rel. uhol**	**Dĺžka fix.**	**# stavov**	**Accuracy**
nie	nie	nie	nie	nie	áno	3	0.653
áno	nie	nie	áno	nie	áno	3	0.650
áno	nie	nie	nie	nie	áno	4	0.639
nie	nie	nie	áno	nie	áno	4	0.633
nie	nie	nie	áno	áno	áno	4	0.628
áno	nie	nie	nie	nie	áno	3	0.625
áno	nie	nie	áno	nie	áno	4	0.614
nie	nie	nie	áno	nie	áno	3	0.602
nie	nie	nie	nie	áno	áno	4	0.602
nie	nie	nie	nie	nie	áno	4	0.596

Príloha D: Článok na IIT.SRC

Extracting Temporal Eye Gaze Features Using Hidden Markov Modelling

Andrej VÍTEK*

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova 2, 842 16 Bratislava, Slovakia
xviteka@stuba.sk

Abstract. Eye movements and eye gaze patterns contain a lot of information about the observer, performed task and visual stimuli. Models which are using information about gaze, are often based only on the evaluation of aggregated metrics of gaze, thereby losing temporal information about visual exploration of the observer. In this paper, we focus on the creation of additional features by using probabilistic models, which could partially abstract temporal information from eye-tracking data. These features are then used to improve results of standard models which are only processing eye-tracking data as aggregated features. To evaluate our method, we use eye movement data from domain of structured text reading.

1 Introduction

Surrounding environment contains too much visual information for us to be processed all at once. For this reason we have adopted to use selective attention, where we focus our perception only on one point and its close surroundings and quickly shift its position on most relevant areas.

Our eye movements are guided by 3 mechanisms - *Bottom-up* (related to visual stimuli), *Top-down* (related to observer) and *Spatial viewing biases* (related to oculomotor system). In study done by Coutrot [2] it is shown, that probabilistic models like Hidden Markov model can capture gaze patterns linked to each mechanism. Main advantage of probabilistic approaches is that, they are data-driven and can learn most of their parameters and their probability distributions from data. Additionally Hidden Markov models model data as a time series instead of set of features, which better represents eye movement.

To effectively use certain algorithms and models like decision trees, support vector machines or others, the eye movement data has to be pre-processed into features which are used for training. Eye movement features can be divided into following categories:

- *Basic features and aggregations* - this category includes statistical characteristics of scanpaths and areas of interest like *sum*, *mean* or *standard deviation* of fixations, saccades and other properties like angles.
- *Spatial features* - features created by spatially mapping eye movement sequences or features of visual stimuli. Common example of spatial features is a heatmap or saliency map.
- *Sequential and geometric comparisons* - features created by comparing multiple scanpaths.
- *Probabilistic features* - most familiar feature in this category is a transition matrix (usually created for areas of interest), which is created during Markov processes.

By only using aggregated features, models lose temporal information about visual exploration of the observer. We focus on creating features by using probabilistic approaches to partially abstract temporal information from eye-tracking data. Our hypothesis is that by adding temporal features to models with only aggregated features we could improve their results.

The key novel thing in our approach is creating model for each predicted class and then using likelihood to generate features. Additionally we work with greater set of input features for training probabilistic models as related work.

1.1 Overview of probabilistic approaches for modelling eye movement

There are many probabilistic approaches and models, but not every model is suitable for modelling eye

* Master study programme in field: Software Engineering
Supervisor: Dr. Jozef Tvarožek, Institute of Informatics, Information Systems and Software Engineering, Faculty of Informatics and Information Technologies STU in Bratislava

IIT.SRC 2019, Bratislava, April 17, 2019, pp. 1–8.

movement data. We have identified 2 main groups of probabilistic models, which are used in articles to model eye-tracking data:

- *Bayesian Mixture model* - uses mixture of probability distributions to model data. Usage of this approach to model eye-tracking data in context of user authentification can be seen in article by Kinnunen [3].
- *Hidden Markov model* - consists from hidden and observable states and additional properties like transition matrix, priors and emissions. Hidden Markov model uses priors and transitions between hidden states to model statistical changes over time in modelled series. Emissions are based on data and are usually represented by probability distribution while modelling eye-tracking data [2, 5].

In this work we focus on Hidden Markov model, because unlike Bayesian Mixture model it can partially learn temporal information about eye movement sequences with priors and transition matrix.

2 Related work

Probabilistic approaches are used in various ways in domain of eye movement modelling from analysis [1], direct classification using probabilistic model [4, 5] to using probabilistic model to create features for additional standard model [2].

Selected input features (features used for training) for probabilistic models in articles are based on task of study and characteristics of data. In article from Kit [4] eye movements were recorded in dynamic scenery (open environment) as such, they chose to use saccade length and direction. In articles [1, 2] they focus on simplicity and interpretability of results so they only use fixation coordinates. In paper from Simola [5] they focus on domain of reading and use input features - fixation duration, saccade length and direction and if given movement is regression.

Similar approach of generating additional features for other models is shown in article by Coutrot [2], where he focuses on creating general Matlab toolbox for processing and classifying eye movement data 'SMAC with HMM'. He uses attributes of trained Hidden Markov models (priors, transition matrix and Gaussians center and covariance of hidden states) as features for other models.

3 Goals of study

Articles mentioned in related work use various input features to train probabilistic models. One of the aims of this paper is to find, which combination of input features can represent eye movement data the best in selected domain. Another goal is to find if adding our features can improve models with only aggregated features.

Based on our goals we seek answers to following research questions:

- *Which probabilistic model with what parameters and input features can represent data from selected domain the best?*
- *Can adding features created by probabilistic models improve results of models which are only using aggregated features?*

4 Likelihood based feature generation with Hidden Markov models

We are focusing on different approach as [2], where used features are attributes of trained models. Our proposed method trains Hidden Markov models and uses them to generate features by comparing likelihood measures for each sequence. Method consists from 3 following parts:

- Data preprocessing and preparation - the data is divided into chunks and transformed into input features
- Training Hidden Markov models - groups of input features are used to train Hidden Markov models
- Feature generation - new features are created by comparing likelihood for sequences between trained Hidden Markov models

Main usage of proposed method is to improve models with classification tasks trained using eye movement data. Architecture of whole method can be seen on Figure 1.

4.1 Data preprocessing and preparation

Before preprocessing and transforming data, it can be additionally divided into chunks based on selected parameters. This step can be used, if data contains tasks with different enough visual stimuli, which could prevent model from learning correct parameters. If used, then for each chunk a different Hidden Markov model will be trained and used to generate features.

Before training models, data sequences are preprocessed and transformed into sequences of input features for training. Number of used input features affects by how many dimensions will hidden states in created Hidden Markov models be represented, which impacts their complexity and interpretability. To be more generic, we only use input features, which can be created from basic eye movement components - fixations and saccades. We use combination of following features to train models:

- Fixation X - horizontal position of fixation on screen
- Fixation Y - vertical position of fixation on screen

– Saccade length - length of outgoing saccade from current fixation
– Absolute angle of saccade - angle between outgoing saccade and vector [1, 0]
– Relative angle of saccade - angle between previous and outgoing saccade
– Fixation duration - time length of current fixation

Given features are inspired by input features used in related work and by features used in article by Steichen [6].

4.2 Training Hidden Markov models

Before training, input features are further divided into groups based on their predicted class. Number of groups is equal to number of predicted classes. Sequences of input features in each group are then used to train Hidden Markov models. We use Baum-Welch algorithm to train individual attributes of models.

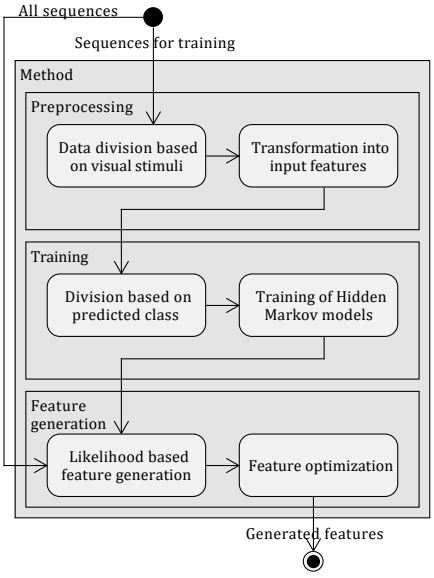


Figure 1. Architecture of proposed method.

4.3 Feature generation

Before generating new features, given sequences must be first transformed into combination of same input features as the ones used for training.

Features are generated by using likelihood measure, which represents probability of how likely would given Hidden Markov model generate given sequence. By training Hidden Markov model for each predicted class individually, we use likelihood measure to represent probability of sequence belonging into given predicted class. Using this we generate following features for each sequence:

– log-likelihood - logarithm of likelihood measure for each model
– relative likelihood - likelihood normalized so it represents percentage probability for each model. Sum of all relative likelihood features for given sequence is 1
– difference - difference between relative likelihoods - only used if there are only 2 predicted classes
– predicted class - is 1 for model with highest probability of generating given sequence. For other models it is 0

Features can be further optimized to maximise their usefulness. Using validation part of data a border probability is found, which represents minimal probability needed to assign predicted class. If model with highest likelihood doesn't achieve border probability, its value for predicted class will be 0.

5 Evaluation

Evaluation of proposed method is related to given research questions. To test if added features can improve other models we compare results of models with added features to a baseline. As a baseline we use models with features created only by using aggregation, because of their frequent use in other works. Additionally by using only features created by aggregation, we lose information about changes in eye movements over time, which allows us to test if partially adding temporal information can improve results. Features used in baseline are taken from articles by Simola [5] and partially from article by Coutrot [2].

To find which parameters are best for representing eye-movement data we test our method with different combinations of input features and parameters on multiple datasets.

5.1 Datasets

We are using 2 datasets from domain of structured text reading and code comprehension, which were collected during user studies with eye-trackers. Task of participants in both user studies was to understand given source code in programming language C and to find correct output after source code execution.

First dataset was collected in smaller study, which contained 55 participants in 2 groups - 27 participants in first group were beginners and 28 participants from second group were intermediate programmers. Participants were working on 6 different programs given in same order and had to manually write an answer into form. Last 2 programs were given only to small

subsection of participants.

Second dataset was collected in larger study with 148 novice programmers and 45 programs (15 different programs with 3 variations each) given in same order. For each program participants had to select 1 from 5 different options representing program's output. 3 options contained values and 4th and 5th options were *Other* and *I don't know*.

5.2 Parameter grid search

To find best combinations of input features for selected domain and other parameters of Hidden Markov models like number of hidden states we used grid search. Grid search was used on smaller dataset with classifier Gradient boosted trees with all input features except fixation duration (not implemented at given time) and number of hidden states from 3 to 5 (frequently used numbers of hidden states from other articles [2, 5]).

Each combination of parameters was evaluated in 2 ways - with general classifier for whole dataset and by classifier for each task with their results averaged.

5.3 Preliminary results

To test our method, we have created classifiers with task of predicting if given participant has selected correct output. We tested 3 classifiers from groups of frequently used classifiers for modelling eye-tracking data - Gradient boosted trees (GBT), Support Vector machine (SVM) and Logistic regression (LR). For classifier Support Vector machine, we also normalized data with mean and standard deviation. To improve precision of results 5-fold cross validation was used. We tested multiple configurations of parameters found with parameter grid search. As a main metric for comparing results we used *F1 micro*. Feature optimization mentioned in section 4.3 was not used.

Configurations were evaluated in 2 different ways, same as during parameters grid search (5.2). You can see results in Table 1 and 2. As seen from tables, there is a slight improvement in results while using generated features. Greatest improvement is achieved by Gradient boosted trees classifier.

Table 1. Results for first dataset. Column used features *represents if generated features were used*

classifier	used features	general results	per task results
GBT	Yes	0.648	0.648
GBT	No	0.603	0.623
SVM	Yes	0.621	0.674
SVM	No	0.616	0.651
LR	Yes	0.623	0.672
LR	No	0.612	0.666

Table 2. Results for second dataset. Column used features *represents if generated features were used*

classifier	used features	general results	per task results
GBT	Yes	0.654	0.670
GBT	No	0.587	0.668
SVM	Yes	0.608	0.708
SVM	No	0.599	0.710
LR	Yes	0.608	0.709
LR	No	0.598	0.714

6 Conclusion

We have proposed a method which uses features created by Hidden Markov models to improve results of standard models. To solve this task, we have created an architecture which transforms data into input sequences, uses them to train models and then creates features by using likelihood.

We have evaluated our method on datasets mentioned in section 5.1. As you can see on results in tables 1, 2 even basic probabilistic features can have impact on results of model. In future work we will focus on creating and testing feature optimization and further evaluation of our method. We also have additional datasets on which we will test portability of trained models.

References

[1] Coutrot, A., Binetti, N., Harrison, C., Mareschal, I., Johnston, A.: Face exploration dynamics differentiate men and women. *Journal of vision*, 2016, vol. 16, no. 14, pp. 16–16.

[2] Coutrot, A., Hsiao, J.H., Chan, A.B.: Scanpath modeling and classification with hidden Markov models. *Behavior Research Methods*, 2017, pp. 1–18.

[3] Kinnunen, T., Sedlak, F., Bednarik, R.: Towards task-independent person authentication using eye movement signals. In: *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, ACM, 2010, pp. 187–190.

[4] Kit, D., Sullivan, B.: Classifying mobile eye tracking data with hidden Markov models. In: *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, ACM, 2016, pp. 1037–1040.

[5] Simola, J., Salojärvi, J., Kojo, I.: Using hidden Markov model to uncover processing states from eye movements in information search tasks. *Cognitive Systems Research*, 2008, vol. 9, no. 4, pp. 237–251.

[6] Steichen, B., Carenini, G., Conati, C.: User-adaptive information visualization: using eye gaze data to infer visualization tasks and user cognitive abilities. In: *Proceedings of the 2013 international conference on Intelligent user interfaces*, ACM, 2013, pp. 317–328.

Príloha E: Poster na IIT.SRC

Extracting Temporal Eye Gaze Features Using Hidden Markov Modelling

Andrej Vítek - andrej2024@gmail.com

Motivation and goals

Focus on:
- Eye-tracking data
- Models with aggregated features

Goals:
- Adding temporal features to improve model's results
- Optimal input feature combination

Data Preprocessing

Data division:
- Important for tasks with different layouts

Input features:
- Fixation coordinates
- Saccade length
- Saccade angle (relative/absolute)
- Fixation duration

Hidden Markov Models

Advantages:
- Data based approach
- Sequence based learning

Training:
- Model for each predicted class
- Multivariate Gaussian Distribution

Each model represents 1 class

Feature Generation

Likelihood measure:
- Sequence generation probability

Generated features:
- Log-likelihood
- Normalized likelihood
- Predicted class

Features generated for each model

Datasets

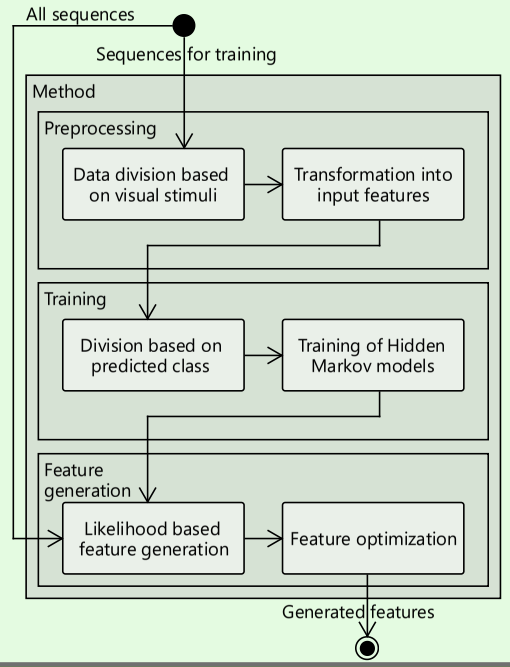
Properties:
- Eye-tracking
- Tasks given in same order

Dataset	Participants	Programs
1	55	4(6)
2	135	15(45)
3	139	15(45)
4	114	15(45)
5	111	15(45)

Task:
- Code comprehension
- Correct output after execution

```
// Uvažujte funkciu gn(). Čo vypíše nasledujúci program?

int gn( int aa, int bb )
{
    int cc = aa + bb;

    bb = cc / aa;

    return bb + cc;
}


int main()
{
    printf("%d", gn( 3, 2 ));

    return 0;
}

```

Architecture



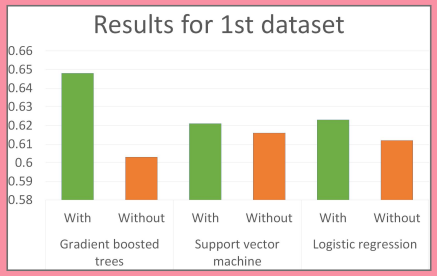
Evaluation and analysis

Evaluation:
- Cross validation
- F1 metric

Multiple classifiers:
- Gradient boosted trees
- Support vector machine
- Logistic regression

Testing:
- Combinations of input features
- Number of hidden states
- Feature optimization

Domain-specific analysis:
- Best combination of features for task and task type



STU FIIT

Príloha F: Opis digitálnej časti práce

Evidenčné číslo práce v informačnom systéme: FIIT-182905-74349

Obsah digitálnej časti práce (archív ZIP):

/Application

- adresár s implementáciou opisovaného riešenia

/Application/FeatureGeneration

- modul s implementovanou metódou

/Application/Data/data

- dáta z experimentov

/Application/Data/results

- výsledky modelov a hľadania optimálnych parametrov

/Documentation

- diplomová práca spolu s anotáciami v slovenskom a anglickom jazyku

/Documentation/Latex

- latex zdrojový priečinok s TeX súborom, BibTeX súborom s použitými referenciami a obrázkami

Digitálna časť práce má veľkosť 1.8 GB, kvôli čomu je uložená v systéme G Suite For Education.

Názov odovzdaného archívu: DP_prilohy_digital_AndrejVitek.zip