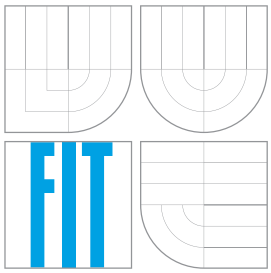
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

KLÁVESNICE POMOCÍ POHLEDU

GAZE-BASED KEYBOARD

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB SZNAPKA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL HRADIŠ

BRNO 2013

Abstrakt

Cílem bakalářská práce je vytvoření nástroje pro psaní pohledem. Zabývá se problematikou snímání pohledu a jeho vyhodnocováním. Obsahuje popis metody Swype, která se používá při psaní na dotykových displejích. Následuje rozbor různých způsobů, pomocí kterých je možné modelovat jazyk, který nástroj používá. Hlavní část práce se věnuje samotnému návrhu nástroje, jenž umožňuje psaní pohledem a jeho implementaci za pomoci toolkitu Kaldi.

Abstract

The goal of this bachelor's thesis is to create a tool for gaze typing. It deals with gaze tracking and evalution issues. It contains a description of the Swype method which is used for typing on touch screen devices. Then follows the analysis of different ways which could be used to model the language used by model. The main part is dedicated to design of the gaze typing tool and implementation using the Kaldi toolkit.

Klíčová slova

Sledování pohledu, Swype, jazykový model, akustický model, Kaldi, konečné stavové automaty, psaní pohledem.

Keywords

Gaze Tracking, Swype, language model, acoustic model, Kaldi, finite state automata, gaze typing.

Citace

Jakub Sznapka: Klávesnice pomocí pohledu, bakalářská práce, Brno, FIT VUT v Brně, 2013

Klávesnice pomocí pohledu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Hradiše. Další informace mi poskytl pan Dipl. Ing. Mirko Hannemann. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jakub Sznapka
13. května 2013

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Michalu Hradišovi za odborné vedení a podněty, které při řešení projektu poskytl. Rád bych také poděkoval Dipl. Ing. Mirko Hannemannovi za pomoc a ochotu při odborných konzultacích.

Obsah

Kapitola 1

Úvod

V současnosti existuje několik způsobů, které lze využít pro psaní na počítači. Nejrozšířenější a také nejběžnější metodou je psaní na klávesnici pomocí prstů. S rostoucím výkonem počítačů a možnostmi zpracování dat se ale začaly objevovat i přístupy nové a doposud neznámé, mezi které patří například rozpoznávání ručně psaného písma. Objevují se ale i metody, které pro psaní nepotřebují použití rukou a spoléhají se na jiné možnosti, které lidské tělo nabízí. Jejich využití je vhodné i pro uživatele, kteří nemohou končetiny pro interakci běžným způsobem použít. Do této kategorie metod můžeme zařadit například diktování textu, které využívá řečové schopnosti uživatele.

Další možností užití lidských smyslů pro psaní spočívá ve využití zraku a psaní pohledem. Systémy pracující s pohledem mají uplatnění zejména u hendikepovaných lidí, kteří nejsou schopni psát jinými způsoby, ale i u obyčejných lidí a činností, při kterých končetiny nelze plně využít. Již v současnosti můžeme evidovat několik nástrojů, které pracují se záznamem pohledu. Hlavním problémem je ale rychlost, s jakou jsou uživatelé schopni při jejich použití psát. Ta dosahuje přibližně $7-20$ slov za minutu [11, 10]. Řešení se nabízí ve využití některého z nově vzniklých pokročilých přístupů k zadávání textu, kterých se díky rozvoji chytrých mobilních telefonů objevilo hned několik. Moderní přístupy, které se při psaní na těchto zařízeních používají, se totiž dají aplikovat i v jiných oblastech a u jiných zařízení. Jednou z takových metod, jež se díky svému konceptu rozšířila i do jiných odvětví, je metoda Swype.

Využití moderních způsobu psaní umožňuje aplikaci ve specifických úlohách, kde běžné metody nedosahují tak dobrých výsledků. Právě prozkoumání možností metody Swype je cílem této práce, ve které se pokusím o implementaci nástroje pro psaní pohledem využívajícího metodu Swype.

Nejprve v kapitole 2 přiblížím problematiku sledování pohledu. Rozeberu jednotlivé fáze procesu snímání, vyhodnocování a představím některé existující řešení pro psaní pohledem. Kapitola 3 charakterizuje metodu Swype, konkrétně pak její princip a využití. Obsahem kapitoly 4 je tvorba rozpoznávacích modelů a struktur, sloužících k jejich uložení. Kapitola 5 se věnuje samotnému návrhu aplikace. V kapitole 6 rozebírám konkrétní implementaci aplikace na základě kapitoly 5 a prezentuji výsledky získané při experimentech.

Kapitola 2

Sledování pohledu

Zpracování pohledu je komplexní a složitý proces, který se skládá z několika kroků. Pro seznámení se se základními problémy a principy zpracování pohledu je určena následující úvodní kapitola. V první podkapitole 2.1, se pokusím definovat pojmy související se zpracováním pohledu a krátce se podívám do historie tohoto odvětví. Definuji pojem Eye Tracking, který se zpracováním pohledu úzce souvisí. Následující podkapitola 2.2 se zabývá snímáním pohledu. Jsou zde uvedeny základní postupy a metody určené pro snímání. Podkapitola 2.3 se zabývá zpracováním dat získaných při nahrávání aktivity oka, jejich vyhodnocením a vhodnou reprezentací. V poslední podkapitole 2.4 se podívám blíže na existující systémy pro psaní pohledem. Provedu jejich rozdělení do dvou základních kategorií a popíši výhody a nevýhody jednotlivých kategorií.

2.1 Eye Tracking

Sledování pohledu se nejčastěji v cizojazyčné literatuře souhrnně označuje jako Eye Tracking. Eye Tracking je charakterizován jako proces měření aktivity oka za účelem určení místa spočinutí pohledu. Jedná se o kompletní proces snímání, vyhodnocování a interpretaci pohledu. [17]

S termínem Eye Tracking souvisí také Gaze Tracking. Tyto pojmy jsou dnes často označovány jako identické, ale není to vždy pravda. Při Eye Trackingu se měří pozice oka vzhledem k hlavě, zatímco u Gaze Trackingu měříme orientaci oka v prostoru, nebo-li takzvaný „point of regard“ (PoR)[1], který bude pro tuto práci nejdůležitější. V případě, kdy je hlava fixovaná a nemění svou polohu vzhledem k okolnímu prostoru, metody Eye a Gaze Tracking splývají. [4]

O prvních pokusech sledování pohledu lze najít zmínku už v roce 1878, kdy francouzský oftalmolog Lousie Émile Javal zjistil, že se oko při čtení nepohybuje plynule po textu, ale provádí krátké zastavení (fixaci) a prudké pohyby (sakády). Dalším důležitým milníkem byl rok 1901, kdy byla vynalezena první relativně přesná neinvazivní technika snímání využívající odrazu světla od zornice. Krátce nato, v roce 1905, se objevila i první metoda, která používala sekvenci obrázků. [7]

V současnosti existuje několik metod snímání s rozdílnými vlastnostmi, které blíže rozebereme v sekci 2.2. Uplatnění těchto metod v praxi můžeme nalézt například v automobilovém a leteckém průmyslu. V těchto oborech se primárně sleduje chování a reakce uživatelů. Typickým výstupem jsou teplotní mapy, které barevně vizualizují místa, kde

[1] Bod na sítnici na který jsou zaměřeny paprsky přicházející přímo z objektu.

spočinul pohled uživatele a barevně rozlišují místa podle doby, po kterou se uživatel na toto místo díval. O něco dále zachází řešení společnosti Tobii[2], které umožňuje ovládání celého počítače výhradně pohledem.

2.2 Snímání pohledu

V první fázi zpracování pohledu je zapotřebí získat pohledová data. V případě experimentů se tato data většinou získají od uživatelů, kteří pohledem realizují předem daný problém, lišící se podle zkoumaného jevu. Proces je nahráván a následně vyhodnocován. Konkrétní způsob vyhodnocení se liší od použité metody. Tento postup snímání pohledu lze souhrnně označit jako proces sloužící k zaznamenání pohledových dat a měření potřebných veličin, které data obsahují. Snímání se většinou provádí za pomocí externích měřících přístrojů. Jejich typ a použití závisí na konkrétním zkoumaném jevu. Může se jednat o běžné přístroje, jako je například videokamera, ale také o vysoce specializované nástroje, jakými jsou například speciálně upravené kontaktní čočky [4]. Na základě zvoleného přístupu nebo použitých metod, lze snímání rozdělit několika způsoby do kategorií. Jedním z kritérií, podle kterého je možné provádět rozdělení, je míra kontaktu zařízení s hlavou uživatele. Na jeho základě můžeme snímání podle Duchowského [4] rozdělit do následujících kategorií:

- Invazivní kategorie – vyžadují přímý kontakt zařízení s okem.
- Neinvazivní kategorie – nejsou s okem přímým kontaktem spojeny.

V počátcích, kdy neexistovaly jiné možnosti snímání pohledu, se používaly pouze invazivní mechanické metody. Důležitým zlomem v tomto odvětví byl již výše zmíněný rok 1901 a první použití neinvazivní metody snímání. Od té doby se postupně přecházelo právě k bezkontaktním neinvazivním metodám. [7]

V dnešní době se invazivní metody využívají zejména pro vysokou přesnost. I přesto, že jejich použití je aktuálně šetrnější než dříve, manipulace a práce s nimi ovšem pořád není úplně snadná. Větší uplatnění nalézají metody neinvazivní, které umožňují snazší, rychlejší a jednodušší manipulaci a jsou mnohem lépe uplatnitelné při běžných činnostech. [4]

Při vyhodnocování záznamu pohledu se může měřit a sledovat několik různých vlastností a s nimi spojených veličin. Je potřeba předem určit, co se má sledovat a jak se mají výsledky použít. Na základě rozhodnutí o použité metodě se zvolí vhodný způsob záznamu a měřené vlastnosti. Kromě pohybů oka je to například tvar zornice, odlesk světelného paprsku od zornice, frekvence mrknutí, klidový potenciál sítnice a další. Při snímání je zapotřebí zajistit eliminaci nebo naopak přítomnost dalších vlivů, které mohou výsledek ovlivnit. U některých metod těmito vlivy mohou být fixace hlavy, u jiných zase přítomnost vhodného typu osvětlení. Pokud nelze dosáhnout zmíněných požadavků, je možné ovlivnit nebo dokonce znemožnit produkování výstupních dat [4]. Samotné způsoby, jak uvádí Duchowski [4], využívající různé přístupy snímání, rozdělujeme do níže uvedených tříd:

- Electro-OculoGraphy (EOG) – Invazivní metoda využívající elektrody umístěné na obličeji uživatele. Využívá toho, že se lidské oko chová jako dipól. Pokud se oko pohne, změní se elektrický potenciál na elektrodách, jež jsou poté schopny zaznamenat i sebemenší pohyb.

[2] http://www.tobii.com

- Scleral contact lens coil – Invazivní metoda, při které je do oka nasazena čočka s cívkou navinutou uvnitř. Při pohybu cívky v magnetickém poli se pak mění napětí a proud na cívce. Signály se následně zaznamenávají a dále převádí podle konkrétních potřeb.

- Video-OculoGraohy (VOG), Photo-OculoGraphy (POG) – Neinvazivní metody, která jsou založeny na snímání oka z určité vzdálenosti a ukládající obrazové informace.

Každá z těchto tříd je unikátní, co se měřeného způsobu týče a s tím souvisí i různorodost jejich použití a vlastností. EOG je relativně šetrnou metodou, která umožňuje snadné nahrávání dat. Využívá se například pro měření schopnosti adaptace na světlo a tmu, kdy se mění klidový potenciál sítnice. Metoda se dokonce používá k interakci s počítačem, a to konkrétně u zařízení Neural Impulse Actuator, které využívá pro svou funkci změny naměřené v nervové a svalové aktivitě v oblasti hlavy. Naproti tomu Scleral contant lens coil zařízení, které využívají upravené čočky, nejsou pro běžné použití vhodné. Jejich použití je diskomfortní a neumožňuje nasazení po delší dobu [4]. Zcela určitě nejkomfortnější, nejjednodušší, a proto také nejrozšířenější metodou, je POG, respektive VOG. Metody jsou si velice podobné, liší se pouze nosičem informace. U POG se jedná o obrázek, zatímco u VOG o sekvenci obrázků – video. Obě metody jsou neinvazivní a při nahrávání se podle Duchowského [4] využívá dvou kategorií přístrojů:

- Head mounted – Záznamové zařízení je pevně spojeno konstrukcí s hlavou uživatele.

- Table mounted – Zařízení je umístěno v blízkosti uživatele s přímým výhledem na oko.

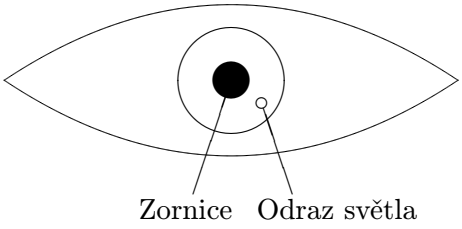
Head mounted konstrukce umožňují jednodušší měření, neboť zde není nutno uvažovat s pohyby hlavy. Pro detekci většinou stačí lokalizovat pouze zornici a sledovat její pohyb. U table mounted konstrukcí už měření není tak jednoduché. Uživatel totiž může pohybovat s hlavou, a je proto zapotřebí získat ještě další informaci ze záznamu, která nám orientaci hlavy vůči okolí pomůže určit. V praxi je touto informací například odlesk světla od rohovky (viz. obrázek 2.1) [4]. Snadné použití, s prakticky nulovou potřebou instalace, činí z table mounted zařízení nejlepšího kandidáta pro běžné použití i za cenu náročnější softwarové realizace.

2.3 Zpracování pohledových dat

Podle účelu, ke kterému se pohledová data po vyhodnocení použijí, se volí i způsob zpracování. Ten se liší už i podle metody, jakou se snímání provádělo, protože různé veličiny nelze interpretovat stejným způsobem. Volba zpracování je taky závislá na využití jejích výstupů. Prozkoumání všech způsobů není předmětem této bakalářské práce, a proto se zaměřím pouze na konkrétní oblast využitelnou při psaní pohledem. Pro psaní pohledem je klíčovou vlastností místo pohledu, nebo-li PoR. Pomocí něj budu určovat, nad kterým písmenem se aktuálně nachází pohled uživatele. Abych s nahranými daty mohl pracovat, je potřeba zajistit jejich vhodnou reprezentaci. Jako nejvhodnější reprezentace pohledových dat v počítači se jeví dvourozměrná Kartézská soustava souřadnic. To je taková soustava, která obsahuje dvojici os x, y, které splňují následující podmínky:

- Jsou na sebe navzájem kolmé.

- Jejich průsečík O odpovídá na obou osách hodnotě 0.



Obrázek 2.1: Měření Point of Regard za pomocí určení vzájemné polohy odlesku světla od zornice.

Kartézská soustava umožňuje zanést místo pohledu jako uspořádanou dvojici (x, y), značící souřadnice pohledu. Pokud budu předem znát přesné rozmístění prvků na klávesnici v době psaní, můžu uspořádané dvojice zpětně mapovat na jednotlivé klávesy. Abych souřadnice vůbec získal, je nejprve potřeba provést vyhodnocení dat získaných při nahrávání. Ty jsou obvykle ve formě sekvence obrázků nebo hodnot veličin naměřených v čase. Vyhodnocení je individuální pro různé metody snímání, a proto záleží na konkrétní metodě, jež se při snímání použila.

Jako příklad lze uvést postup, který se nejčastěji využívá u VOG metod pro měření PoR [4]. Pravděpodobně se tak bude uplatňovat i v případě psaní pohledem. Spočívá v osvětlování oka externím zdrojem světla a odlesku světelného paprsku od zornice (viz. obrázek 2.1). V praxi se nejčastěji používá infračervený zdroj osvětlení[3]. Jeho výhoda spočívá v tom, že lidské oko nemá potřebné receptory, které by jej detekovaly. Uživatel nemá potřebu mrkat nebo odvracet zrak, jako je tomu u paprsků z viditelné části spektra. Tím se zvyšuje uživatelský komfort a uživatel je schopný bez větších obtíží nástroj delší dobu používat. Psaní na nástroji je nahráváno na videozáznam, z něhož je následně počítána vzájemná pozice odlesku infračerveného světla a středu zornice. Samozřejmostí je pořizování záznamu, který umožňuje zachycení světla v infračervené spektru. Výpočet může být prováděn ze záznamu, ale také v reálném čase. Na základě vzájemné polohy je nástroj schopen určit místo pohledu. [17]

2.4 Psaní pohledem

Jednou z možností interakce mezi člověkem a počítačem, při které lze využít snímání pohledu, je již zmíněné psaní. Psaní pohledem umožňuje nasazení v případech, kdy klasické metody využívající končetiny nelze použít. Mimo aplikaci v rozšířené realitě má daný způsob interakce obrovský přínos pro osoby s tělesným hendikepem nebo s poruchami hrubé motoriky. Pro tyto osoby je psaní pohledem mnohdy jediný způsob umožňující interakci s počítačem, potažmo s okolím. Většina nástrojů pro psaní pohledem je založena na displeji se zobrazenou klávesnicí. Konkrétní rozvržení klávesnice se liší. Co je ovšem u všech systémů podobné, je způsob zadávání textu. Ten spočívá v zaměření pohledu na místo nebo symbol, který chceme zapsat. Postupným skládáním jsme pak schopni získat požadované slovo.

Aktuálně je k dispozici několik nástrojů sloužících k psaní pohledem, které ovšem pracují

[3]Světlo s vlnovou délkou 700 nm - 1 mm. Pro osvětlování se z hlediska bezpečnosti používá světlo s vlnovou délkou okolo 900 nm.

na různých principech. Jejich velkým nedostatkem je rychlost zadávání textu [8]. Abychom mohli porovnat jednotlivé přístupy, je nejprve zapotřebí odlišit od sebe dva různě využívané způsoby zadávání textu. Ty vycházejí z poznatku, že lidské oko není schopno podobně jako například prst provést potvrzení, nebo-li volbu klávesy. K psaní je potřeba tuto možnost zprostředkovat jiným způsobem. Buďto doprovodným jevem nebo zvolením takového způsobu zadávání textu, který potvrzení nevyžaduje. Na uvedeném základě pak můžeme systémy rozdělit do dvou odlišných kategorií [8]:

- Dwell – psaní pomocí zaměření (potvrzení) symbolu.
- Dwell free – kontinuální psaní bez potvrzování.

Vlastnosti systémů spadajících do jednotlivých kategorií jsou vcelku odlišné. Obecně lze tvrdit, že dwell techniky dosahují horších rychlostí zadávání než techniky dwell free [8]. Princip dwell zadávání spočívá v nutnosti potvrzení symbolu, které může být provedeno různými způsoby. Psaní může být v nejjednodušším případě realizováno pouze pomocí zaměření a nejsou potřeba žádné další doprovodné jevy nebo výpočty.

K volbě symbolu se v praxi nejčastěji používá zaměření pohledu na kýžený symbol po určitý čas. Čas se průměrně pohybuje v rozmezí od 400 ms do 1000 ms. [11]

Časový úsek potřebný k napsání znaku je samozřejmě závislý na uživatelské zkušenosti. Platí, že zkušenější uživatelé jsou schopni provést fixaci na symbol za mnohem kratší čas než uživatelé, kteří se systémem pracují poprvé. Některé pokročilé nástroje proto umožňují, aby si každý uživatel čas přizpůsobil. [10]

Nicméně i přes všechny zmíněné možnosti, se nejmenší experimentálně změřená doba potřebná k zaměření u trénovaných uživatelů pohybuje okolo 300 ms. Teoreticky jsme schopni se zaměřením napsat až 26 slov za minutu. [11]

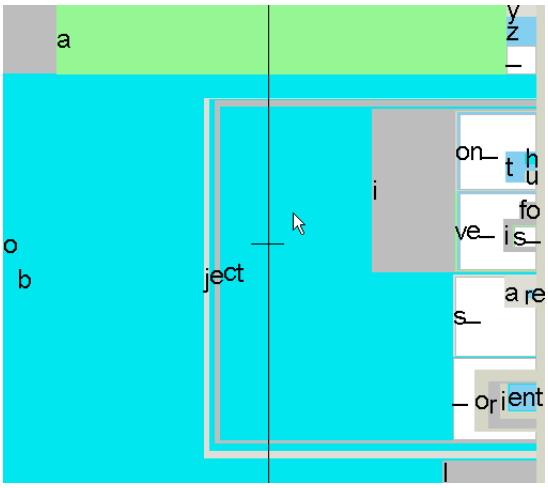
Oproti klasickému způsobu psaní na klávesnici, kdy se průměr psaní pohybuje okolo 39 slov za minutu, je tento způsob relativně pomalý[16]. Pokud vezmeme v potaz i soustředění a úsilí, které uživatel musí při psaní vynaložit, dojdeme k závěru, že ačkoliv jsou dwell systémy realizačně jednoduché, jejich použití v praxi není příliš dobré.

Jako řešení problému s nízkou rychlostí je použití dwell free technik. Systémy tohoto typu nemají potřebu zaměření a je pouze na uživateli, jak rychle dokáže najít správný symbol a poté zrak přemístit jinam. I u netrénovaných jedinců, kteří znají alespoň přibližné rozvržení klávesnice, je možné dosáhnout mnohem lepších časových výsledků. Experimentálním způsobem bylo zjištěno, že jsme dwell free přístupem schopni dosáhnout rychlosti psaní až 46 slov za minutu [8]. V porovnání s nástroji vyžadujícími zaměření si pak tento způsob psaní stojí velice dobře. Odstranění fixací ale také skýtá několik problémů. Hlavním problémem zmíněné metody jsou nepřesnosti způsobené uživatelem. Nejsme schopni přesně říci, které symboly chtěl uživatel skutečně napsat a přes které pouze přejel pohledem a provedl krátkou fixaci. Výskyt falešných fixací je častý při zmateném hledání požadovaného symbolu a je tím větší, čím je větší neznalost rozvržení klávesnice. Lze tedy říci, že zkušený uživatel bude dosahovat lepších výsledků, v závislosti na počtu chybných fixací než uživatel nezkušený. Odbourání problému způsobených uvedeným jevem proto vyžaduje vytvoření takového modelu, který bude umožňovat větší míru tolerance chybných vstupů.

Jedním z nástrojů, jenž se pokouší odstranit neduhy dwell systému je Dasher[4]. I přesto, že je založen na dwell systému, pracuje podobně jako dwell free systémy. Nástroj využívá neustále se zvětšující (zoomující) rozhraní, které obsahuje různě velké, barevně oddělené pole

[4] http://www.inference.phy.cam.ac.uk/dasher/

obsahující jednotlivé symboly. Počáteční velikost pole je určena pravděpodobností, s jakou bude symbol obsahující pole následovat za předchozí sekvencí symbolů. Více pravděpodobné symboly jsou umísťovány do větších, barevně výraznějších a více do středu umísťovaných bloků. Psaní pak spočívá v přesouvání pohledu do kýžených polí. Pomocí nástroje uživatelé dosahují rychlosti psaní okolo 26 slov za minutu. Nástroj umožňuje přidání vlastních jazyků. Výhoda spočívá v jednoduchosti zadávání nových slov. Uživatel se nemusí přepínat do zvláštního režimu. Pouze počká než se objeví požadovaný symbol. [23, 22]



Obrázek 2.2: Psaní slova *objection* pomocí nástroje Dasher

Zdroj: (http://www.inference.phy.cam.ac.uk/dasher/)

Kapitola 3

Swype

Technik, které lze pro psaní pohledem použít, je několik. Je ale potřeba vybrat takovou techniku, která bude dwell free způsobu zadávání vyhovovat nejlépe. Jako možná varianta se jeví upravení klávesnice T9 pro psaní pohledem.

Princip T9 klávesnice spočívá v seskupování symbolů. Psaní textu pak vypadá tak, že uživatel přesune pohled nad klávesu, která obsahuje ve skupině symbolů požadované písmeno. Tímto způsobem uživatel postupně zapíše sekvenci symbolů a na základě predikce jsou mu následně vybrány nejpravděpodobnější slova. [9]

Výhodou je právě ono seskupování, které umožňuje vytvoření různě velkých polí pro různě pravděpodobná písmena. Seskupování je ale zároveň i hlavní nevýhoda. Při psaní pohledem totiž dochází k postupnému přemístění pohledu z písmena na písmeno, během kterého je sekvence neustále zaznamenávána. V napsané sekvenci se pak objevují i klávesy, které uživatel zapsat nechtěl. Při rozhodování o nejpravděpodobnějším slovu je pak nutné započítat všechny varianty slov složených ze seskupených symbolů. Proto je vhodné použít jiný přístup využívající plnou klávesnici, kde každá klávesa označuje právě jeden symbol. U něj sice taky dojde k zapsání nepožadovaných symbolů, ale v závislosti na jejich posloupnosti není potřeba kontrolovat tolik různých variant.

Podkladem pro nalezení optimální metody nám může být zadávání textu na moderních dotykových displejích. Pro tento typ displejů existuje velké množství různě rozvržených klávesnic a způsobu psaní na nich. Společnou vlastností dotykových zařízení a zařízení pro psaní pohledem je nepřesnost, se kterou pracují. U dotykových displejů použitých v chytrých mobilních telefonech je nepřesnost dána zejména malou velikostí kláves v poměru k palci, kterým se text zadává a neschopností odezvy kláves. U psaní pohledem pak nepřesným snímáním a již výše zmíněným přesouváním pohledu. U obou přístupů se ji snažíme co možná nejvíce eliminovat analýzou již napsaného textu a predikcí textu následujícího. Tento účel plní jazykový model, který obsahuje seznam slov a určitá pravidla pro skládání slov za sebou.

3.1 Princip zadávání textu

Nejdále je v oblasti rychlého psaní technika využitá v nástroji Swype[1][2]. Jedná se o metodu a současně aplikaci, která vznikla pro chytré mobilní telefony s operačním systémem, jako

[1] http://www.swype.com

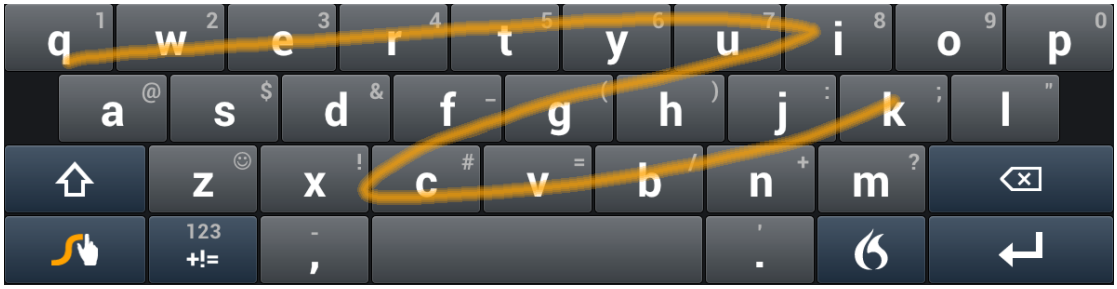
[2] Existuje několik produktů využívajících stejný způsob. Swype je ale nejrozšířenější z nich, a proto její název budeme používat.

možná alternativa výchozí klávesnice. Umožňuje psaní dotykem, a to posouváním prstu po symbolech nebo ručním psaním izolovaných písmen a jejich detekcí. Obsahuje také možnost diktování textu za pomoci hlasu. Zaměříme se ale zejména na techniku, která se používá pro kontinuální psaní bez nutnosti dotykových prodlev mezi jednotlivými písmeny.

Rozhraní, které se používá při psaní, je plná QWERTY klávesnice[3]. Podstatou a hlavním rozdílem, který dělí Swype od jiných metod, je možnost psaní celých slov bez nutnosti zdvihnout prst z obrazovky. Psaní textu probíhá tak, že uživatel prstem přejíždí mezi symboly a nezvedá prst. Zvednutí prstu je označeno jako konec slova. Trajektorie pohybu je vyhodnocena a je nabídnuto nejpravděpodobnější slovo. V případě, že uživatel dané slovo nechtěl napsat, mu je k dispozici seznam dalších méně pravděpodobných slov. Pokud ani zde nenajde kýžený výraz, je zde možnost napsat slovo klasickým způsobem a tím jej přidat do slovníku. Důležitý je fakt, že uživatel nemusí vždy prstem přejet po poli přesně vyhrazeném pro daný symbol. Pro nalezení správného výsledku totiž systém nevyužívá pouze místo dotyku, ale i jeho okolí. Napsání slova *quick* metodou Swype je vizualizován na obrázku 3.1. [14]

Swype vyniká jednoduchým způsobem definování vlastních slov. To může být učiněno buďto ručně, nebo také skenováním emailů a textů. Nástroj je založen na historii a jeho chování se tedy adaptuje pro každého uživatele individuálně na základě používání. Podporuje resistivní i kapacitní displeje, a je proto dostupný na všech běžně rozšířených platformách[4]. V době vzniku bakalářské práce ovládá více než 55 jazyků. Pokročilý systém predikce je dokonce schopen nápovědy následujících slov.

Jelikož aplikace není dostupná pod některou z otevřených licencí, nelze přesně říci jak funguje a jaké metody pro vyhodnocování využívá. Analýzou chování lze ale některé z klíčových vlastností vydedukovat. Jednou z nich je už samotný způsob zadávání. Namísto použití vlastností vestavěné systémové klávesnice, která je schopna určit konkrétní písmeno na které uživatel klepl, se použije celá trajektorie pohybu. Tu tvoří sekvence souřadnic značících polohu prstu na displeji v čase. Tento přístup umožňuje rozdílný způsob interpretace, který by u systémové klávesnice vedl k jiným nebo žádným vstupům. Další, očividně přítomnou části, je využití pravděpodobnostních modelů pro predikci. Jejich funkce se využívá pro vyhodnocení trajektorie a získání nejpravděpodobnějších slov. Realizace je nejspíše obstarána pomocí jedné z variant automatů, která modeluje použitý jazyk. Vyhodnocování spočívá v průchodu automatem a počítání pravděpodobností, na něž systém cestou modelem narazil. Seznam slov s největší pravděpodobností je předán uživateli jako výstup.



Obrázek 3.1: Psaní slova *quick* metodou Swype.

Zdroj: (http://www.swype.com)

[3]Standart dnešních Anglických klávesnic. Název je odvozen od prvních 7 písmen klávesnice. Toto rozvržení je mimo jiné součástí patentu podaného v roce 1874 Christopherem Sholesem.

[4]iOS, Android, Symbian, a jiné.

3.2 Využití

Výhodou techniky psaní Swype je především rychlost. Ta se samozřejmě liší podle úrovně uživatelských zkušeností. U trénovaného člověka se rychlost pohybuje až okolo 60 slov za minutu[5], čímž se řadí mezi nejrychlejší způsoby zadávání textu na zařízeních využívajících dotykové rozhraní.

O různorodých možnostech použití metody svědčí také její aplikace na jiných než dotykových zařízení. Konkrétní příklad můžeme nalézt v systémech umožňujících psaní hendikepovaným uživatelům. Pro tuto skupinu uživatelů tak může psaní, ať už pohledem nebo například pohyby hlavy, představovat dobrý a mnohdy i jediný způsob komunikace. Jako konkrétní příklad lze uvést aplikaci metody Swype, která byla realizována při pokusu o překonání Guinnessova světového rekordu v psaní bez pomoci rukou [1]. K tomu bylo využito zařízení trackující pohyb hlavy TrackerPro a zadávání metodou Swype. Uživatel, který nový rekord stanovil, byl vyjma hlavy kompletně paralyzován. Za využití headpointeru TrackerPro byl schopen napsat 26 slov za 83,09 sekund. Po převodu a zaokrouhlení je tedy naměřená rychlost přibližně 19 slov za minutu.

[5] Této rychlosti bylo dosaženo překonáním světového rekordu. Zdroj: (http://www.guinnessworldrecords.com/world-records/8000/fastest-time-to-write-a-text-message-(sms)-on-a-touchscreen-mobile-phone)

Kapitola 4

Rozpoznávací modely

K realizaci psaní pomocí metody Swype bude zapotřebí vytvoření vhodného modelu pro reprezentaci jazyka. K tomu slouží v této kapitole zmíněné jazykové modely. V reálných úlohách zpracování přirozeného jazyka existuje několik struktur pomocí kterých je možno jazyk modelovat. Jednou z částí, kterou se budu v této bakalářské práci zabývat, je jazykový model. Jeho funkce je popsána v podkapitole 4.1. Další částí, která je v této bakalářské práci důležitá, je akustický model, kterému se věnuji v kapitole 4.2. Ten se sice používá při rozpoznávání řeči, ale použití jeho modifikace je stěžejní pro tuto bakalářskou práci. V poslední podkapitole 4.3 se věnuji datové struktuře Weighted Finite State Transducers, která implementuje zmíněné modely do jediné struktury.

4.1 Jazykový model

Jazykový model je pravděpodobnostní mechanismus pro generování textu [2]. Funkcí a účelem jazykových modelů je vytvoření struktury a vazeb mezi prvky uvnitř této struktury, které umožní popis jazyka. Jazykové modely můžeme rozdělit na deterministické (založeny na gramatikách) a stochastické (nebo též statistické). Deterministické modely jsou tvořeny odborníky, kteří sestavují formální gramatiku na základě jejich znalostí o daném jazyku. Naproti tomu stochastické modely se trénují na velkých datových sadách a využívají pravděpodobnosti výskytů slov a symbolů. Použité datové sady obsahují velké množství textu, z něhož se extrahují informace o počtu výskytů symbolů, slov a jejich vzájemné poloze. V současnosti se pro modelování využívají zejména nedeterministické stochastické jazykové modely. Je to dáno zejména tím, jak se mění požadavky na zařízení využívající zmíněné modely. Zatímco zpočátku se jednalo o pouhá izolovaná slova a výrazy, dnes jde především o rozpoznávání souvislých textů. K tomuto požadavku se uvedené stochastické modely hodí více, protože jejich tvorba je jednodušší, vyžaduje méně úsilí a výsledný model je pro účely práce s velkými slovníky robustnější a komplexnější. [15]

V praxi mohou být stochastické jazykové modely a jejich tvorba založeny na několika přístupech. Jednou z možností, jak přirozeně jazyk reprezentovat je užití statistických N-gramových modelů. Ty vyjadřují pravděpodobnost výskytu prvků na základě četností jeho výskytu s jinými slovy, která jsou s ním ve vztahu. Modely tedy dokáží vyjadřovat pravděpodobnosti prvků vzhledem k jejich okolí. Pomocí „chain rule“ můžeme vyjádřit pravděpodobnost výskytu slova $P(W)$ pomocí obecného N-gramového modelu následující rovnicí [15]:

$$P(W) = P(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n} P(w_i | w_1 w_2 \ldots, w_{i-1})$$

Výpočet takové pravděpodobnosti ale není výpočetně proveditelný. Znamenalo by to vyhodnocení $V^n - 1$ nezávislých parametrů pro slovník o velikost V a n-té slovo. Tento problém se řeší aproximací. N-gramové modely mohou mít různou podobu v závislosti na tom, s kolika okolními prvky pracují. N-gramu o velikosti 1 se říká unigram. Unigramové modely vyjadřují pravděpodobnost výskytu jednoho samotného symbolu. N-gram o velikost 2 se nazývá bigram. Bigramové modely vyjadřují pravděpodobnost, že daný symbol následuje za symbolem předchozím. Použití N – gramových modelů umožňuje pracovat se sekvencemi podle konkrétních potřeb a trénovat na menších datových sadách. [15]

4.2 Akustické modely

Pro modelování zvukové stránky jazyka existuje akustický model, který se používá při zpracování řeči k modelování jednotlivých zvuků, které tvoří slova. Těmto zvukům říkáme fonémy a značí nejmenší zvukovou jednotku jazyka, která je ještě rozlišitelná. Získání akustického modelu spočívá v použití velkého řečového korpusu, nad kterým se pomocí vhodných algoritmů vytvoří statistická reprezentace fonému. Dekódování následně probíhá tak, že se zvuková stopa rozdělí na menší části a ty se porovnávají s akustickými modely jednotlivých fonémů. V případě shody se uloží název fonému a pokračuje dokud nenarazí na pauzu. Následně dochází k porovnání získaných fonémů a fonémových předpisů slov obsažených v lexikonu. [6]

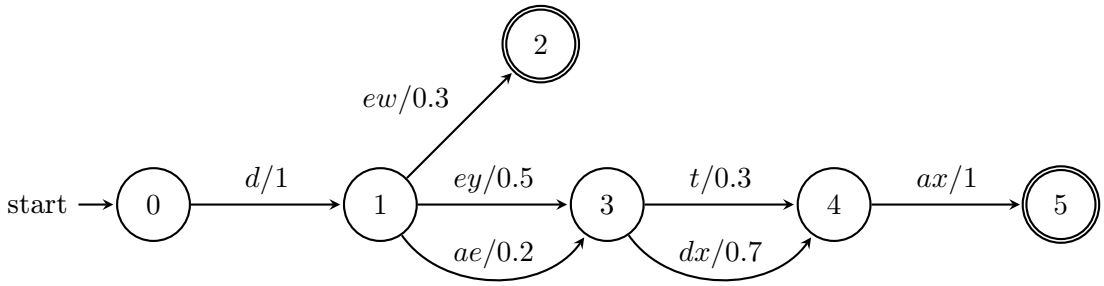
I přesto, že se akustické modely používají při zpracování řeči, mají pro tuto práci svůj význam. Jejich úpravou jsem totiž schopen vytvářet modely písmen.

Jednou z několika možností reprezentace akustických modelů jsou Skryté Markovovy modely (HMM). HMM jsou konečné stochastické automaty. V současnosti jsou považovány za specifickou formu dynamických Bayesovských sítí. HMM se skládají ze dvou stochastických procesů. Prvním z nich je Markovovský řetězec, který je charakterizován stavy a přechodovými pravděpodobnostmi. Druhý stochastický proces produkuje „emise“ pozorovatelné v každém okamžiku, závisející na rozdělení pravděpodobnosti v závislém stavu. V každém stavu můžeme pozorovat události, aniž bychom museli vědět, ve kterém stavu se automat nachází. Takovým stavům se říká skryté. Pojem skryté se vztahuje ke stavům, ne k parametrům. Příklad použití HMM k tvorbě akustických modelů je znázorněn na obrázku 6.1 a 6.2. [5]

4.3 Weighted Finite-State Transducers

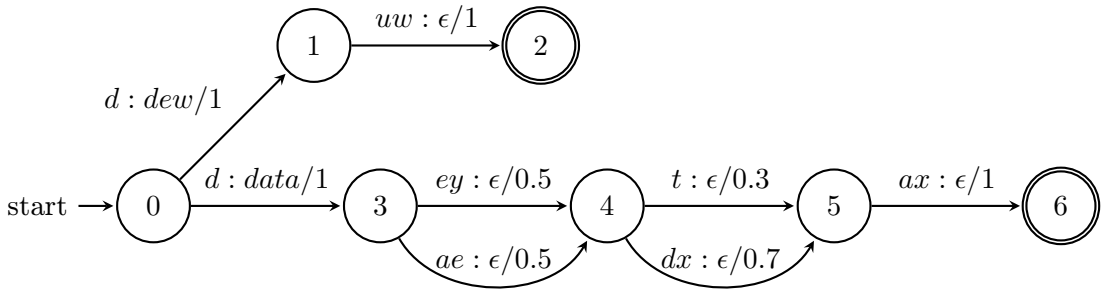
Pro jazykové modelování se využívá různých typů automatů. Speciálním případem jsou Weighted Finite-State Transducers (WFST). Jedná se o automaty, které na každém přechodě obsahují mimo vstupního symbolu také symbol výstupní. Hojně se využívají při rozpoznávání zvuku, obrazu, ale také textu. WFST umožňují mapování různých typů informací v rámci jednoho modelu. Lze pomocí nich využít jednoduchou a přirozenou reprezentaci HMM modelů, kontextově závislých modelů, gramatik a dalších. [12]

Obrázek 4.1 a 4.2 srovnává transducery a automaty na příkladu rozpoznávání slov *data* a *dew*. V automatu jsou vždy všechny možné výslovnosti pro danou posloupnost uloženy



Obrázek 4.1: Graf automatu pro slova *data* a *dew*.

Zpracování na základě Mohriho [12]



Obrázek 4.2: Graf transduceru pro slova *data* a *dew*.

Zpracování na základě Mohriho [12]

jako fonémy na přechodech. Výslednou pravděpodobnost slova získáme sečtením všech dílčích pravděpodobností, které jsme po cestě automatem prošli. U transducerů jsou možné výslovnosti uloženy jako fonémy na přechodech, ale model navíc obsahuje výstupní symbol, který definuje následující sekvenci. Ve chvíli, kdy se vyskytujeme na přechodu, u kterého víme, že následující sekvence je jednoznačná, je výstupním symbolem přechodu odpovídající slovo. Je možné kombinovat slova se stejnou výslovností, aniž bychom ztratili jejich identitu. Z obrázků 4.1 a 4.2 lze také vidět, že transducery mohou reprezentovat vztah mezi více úrovněmi pomocí jediného modelu. [12]

Kapitola 5

Návrh a provedení

Cílem bakalářské práce je vytvoření nástroje pro psaní pohledem metodou Swype. Před vytvořením nástroje jsem si jej nejprve rozdělit na několik dílčích částí, které dále detailněji rozeberu. Celý proces je možno rozdělit na dvě části:

- Nahrání a vyhodnocení pohledu uživatele.
- Zpracování pohledových dat modelem a získání výsledků.

Implementace nástroje pro nahrávání dat není předmětem bakalářské práce. Budu ale potřebovat výstupy z této fáze zpracování, abych nástroj mohl otestovat a provádět experimenty. Nahraná data mi budou sloužit jako vstup do další části, kterou je jejich zpracování. Získané zdrojové soubory z fáze nahrávání obsahují dva typy potřebných informací. Konkrétně se jedná o umístění jednotlivých kláves na obrazovce v době nahrávání a samotná nahraná data. Vytvoření datového obrazu klávesnice z dat obsahující jejich fyzické rozložení se budeme věnovat v podkapitole 5.1. Na vytváření jazykového modelu podle konkrétních požadavků se zaměříme v podkapitole 5.2.

5.1 Mapa kláves

Pozice uživatelova pohledu získané při nahrávání jsou, jak už bylo zmíněno v sekci 2.3, tvořeny vektorem uspořádaných dvojic (x, y) umístěným do Kartézské soustavy souřadnic. Z pouhých souřadnic ovšem nemůžeme určit konkrétní písmeno na které se uživatel díval nebo pozici uživatelova pohledu na klávesnici. Abych byl schopen toto zpětné mapování provést, je potřeba vytvořit obraz klávesnice pomocí datového modelu. Právě vytvářením takového modelu se budu v této podkapitole zabývat. K jeho vytvoření potřebuji znát rozložení kláves v době nahrávání – konkrétně střed klávesy a písmeno k ní přiřazené. Data jsem odečetl z klávesnice, která nám sloužila jako podklad při nahrávání psaní. Stejně jako u vektoru s nahranými daty, i zde jsem zvolil uložení informace pomocí uspořádané dvojice (x, y) a k ní pouze přiřadil odpovídající písmeno. Informace o fyzickém rozložení v tomto tvaru mi stačí k sestavení geometrického modelu rozložení klávesnice. Nezbytné je už pouze určit rozestup mezi jednotlivými klávesami. Získané rozdíly určují přibližnou velikost každé klávesy. Pro osu x se získání rozestupu provede odečtením x-ové souřadnice kláves sousedících na jednom řádku vedle sebe. Pro osu y pak rozdílem y-ových souřadnic kláves sousedících o jeden řádek nad nebo pod sebou.

Pro vytvoření vektoru příznaků je k dispozici nahraná sekvence. Abych se sekvencí mohl dále pracovat, je zapotřebí pro každý její prvek určit, ke které klávese patří a s jakou pravděpodobností. To je sice možné realizovat až v samotném jazykovém modelu, který může mít informace o rozložení klávesnice, ale manipulace s modelem je poté výrazně složitější a vede ke stejným výsledkům. V mém případě provedu vyhodnocení již na samotném vektoru a pouze zařídím, aby odezva prvků uvnitř modelu zůstala nezměněna. Získám tím možnost aplikovat pokročilejší metody reprezentace, než jakou by mi nabídly metody realizované v rámci toolkitů pro práci s modelem. Cílem je vyrobit takovou reprezentaci klávesnice, jež bude nejlépe odpovídat potřebám metody Swype. To znamená, že reprezentace musí umožňovat napsání znaku, i když pohled neprochází přímo polem (klávesou) vyhrazeným pro daný znak. Toho docílím tím, že pro každý prvek vyjádřím příslušnost k jednotlivým klávesám a nebudu stanovovat tvrdé rozhodnutí, které by jej přiřadilo pouze k jedné klávese. Druhým požadavkem na model je možnost ignorovat některá písmena, přes která uživatel přejel pohledem při přesunu zraku na znak jiný. Tento problém ovšem není tak triviální jako problém předchozí. Při přesunu pohledu se totiž zaznamená několik vzorků, které uživatel nezamýšlel napsat. Detekce sakád[1] a fixací[2] by teoreticky mohla pomoci problém vyřešit. Při zaměření na požadované písmeno budu schopen detekovat fixaci, zatímco přesun k jiné klávese bude odpovídat sakádě. Je ale nutné zvážit i situace, kdy uživatel hledá požadovaný znak a vytvoří tak několik chybných fixací. Chybné fixace by pak byly také označeny společně s ostatními jako správné. K odstranění tohoto problému jsem proto navrhl vytvoření nové klávesy, která se na fyzické klávesnici v době psaní nevyskytuje, protože je přítomna všude a mezi každým záznamem pozice pohledu. Symbol, který klávesa reprezentuje, pak bude možné vložit mezi kterékoliv dva jiné symboly. Stanovením vhodné pravděpodobnosti pro výskyt „prázdného“ symbolu potom budu moci dosáhnout jeho vybrání modelem. Jelikož symbol nereprezentuje žádné písmeno, bude při vyhodnocování přeskočen, nebo ignorován.

Jako vhodná reprezentace zmíněných požadavků se jeví Multivarite Gaussian Mixture Model (dále jen MGMM). MGMM je parametrická funkce hustoty pravděpodobnosti, která je vyhodnocována jako vážený součet komponent [19]. Komponenty jsou modelovány jako multidimenzionální varianty Gaussovy funkce hustoty pravděpodobnosti. Obecný tvar MGMM popisuje rovnice [19]:

$$\begin{aligned} p(x|\lambda) = & \sum_{i=1}^{M} w_i g(x|\mu_i, \Sigma_i), \\ \lambda = & \{\mu_i, \Sigma_i\}\, i \ = 1, \ldots M, \end{aligned} \tag{5.1}$$

kde x značí $D-$dimenzionální vektor příznaků, $w_i, i = 1, \ldots, M$ značí váhy komponent a $g(x|u_i, \Sigma_i), i = 1, \ldots, M$ jsou Gaussovské hustoty, které pro nás mají tvar dvourozměrné Gaussovy funkce. Ta je definována pomocí μ, které označuje střední hodnotu a Σ, vyjadřující kovarianční matici[3]. Hustotu pravděpodobnosti spočítáme pomocí rovnice [19]:

[1] Krátké prudké pohyby způsobené při přesouvání pohledu

[2] Upření pohledu na jedno místo po velmi krátkou dobu.

[3] Matice, jež umožňuje charakterizovat přesnost a závislost (korelaci) vícerozměrných veličin. Na její diagonále leží variance, kvadráty směrodatných odchylek σ^2, a na ostatních místech jsou kovariance.

$$g(x|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} exp\left\{-\frac{1}{2}(x-\mu_i)'\Sigma_i^{-1}(x-\mu_i)\right\}, \tag{5.2}$$

s vektorem středních hodnot μ_i a kovarianční maticí Σ_i.

Odezva konstanty v modelu je jasná, ale výpočet odezev Gaussovských komponent podle rovnice 5.2 je možno dále upravit. Pokud je kovarianční matice Σ pouze jednorozměrná, je možno počítat odezvu pro každou dimenzi zvlášť. Situace nastane, pokud jsou data nezávislé. O souřadnicích určujících polohu klávesy můžeme říct, že jsou nezávislá, a proto lze výpočet provést pomocí jednorozměrné Gaussovy funkce s rovnicí [3]:

$$g(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} exp - \frac{(x-\mu)^2}{2\sigma^2}, \tag{5.3}$$

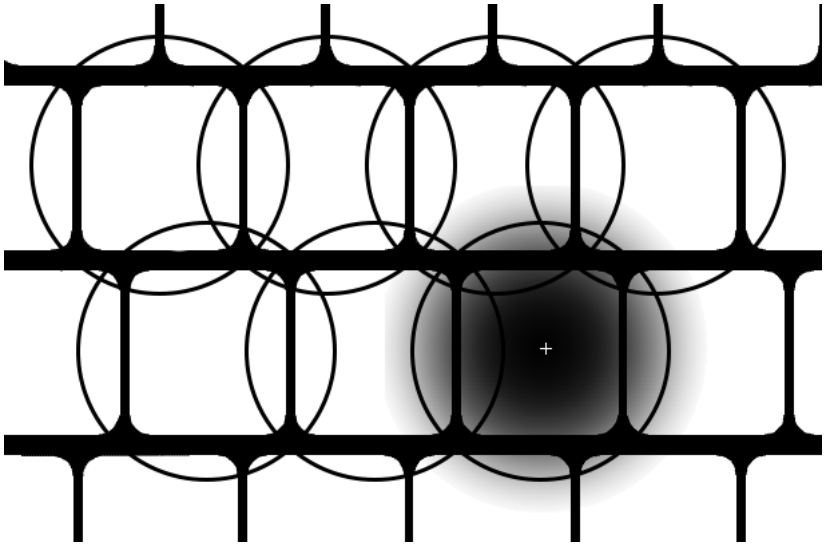
kde μ označuje střední hodnotu a σ značí směrodatnou odchylku souřadnic a výsledné hodnoty stačí pouze vynásobit. Výpočet odezvy jedné komponenty je znázorněn rovnicí:

$$\begin{aligned} p(x, y, \lambda) &= g(x, \mu_x, \sigma_x) * g(y, \mu_y, \sigma_y), \\ \lambda &= \{\mu_x, \mu_x, \sigma_x, \sigma_y\}, \end{aligned} \tag{5.4}$$

kde x, y značí souřadnice z Kartézského systému souřadnic, μ_x, μ_y jejich střední hodnotu, σ_x, σ_y směrodatnou odchylku.

V mém případě, kdy budu pracovat s možností přeskakování písmen, ale tento model rozšířím. Speciální symbol označený `SIL` sloužící k přeskakování zde bude muset být modelován jinak, než je u modelů MGMM běžné. Jeho vyjádření pomocí Gaussovské komponenty by mělo za následek nízkou pravděpodobnost na okrajových klávesách, nebo velmi vysokou pravděpodobnost u kláves umístěných více do středu. Gaussovské komponenty zde proto nejde použít, a jelikož pravděpodobnost přeskočení symbolu je po celé ploše klávesnice stejná, bude zde mít symbol `SIL` konstantní pravděpodobnost. Aplikací této úpravy tedy znemožním použití klasického MGMM modelu, ale vytvořím model vlastní.

Ze získaných informací o rozvržení klávesnice jsem schopen podle zmíněného modelu vytvořit datovou reprezentaci. Ta bude reprezentována Gaussovskými komponentami a konstantou. Jelikož jsem se rozhodl modelovat klávesnici mimo samotný nástroj, který bude jazykový model vytvářet, budu k vyhodnocení muset přistupovat sám. To spočívá ve vyhodnocení získaných dat a jejich mapování na klávesnici ještě před samotným dekódováním na jazykovém modelu. V principu se jedná o spočítání odezvy na každé Gaussovské komponentě a přidání konstanty `SIL`. Pokud bychom chtěli vizualizovat výslednou mapu nad klávesami, vypadala by přibližně jako na obrázku 5.1. Dvourozměrná Gaussova funkce označující jednu komponentu je zde symetrická. Směrodatné odchylky jsou v obou dimenzích stejné, a proto má tvar kruhu. Černé kružnice značí vzdálenost 2σ od středu jednotlivých kláves. Tmavší oblast pak příklad hustoty rozdělení pravděpodobnosti jedné klávesy. Na obrázku ale není znázorněna konstanta `SIL`, která má po celé ploše klávesnice stejnou hodnotu



Obrázek 5.1: Vizualizace datového modelu klávesnice.

5.2 Jazykový model

Abych dokázal vstupní vektor pravděpodobností, získaný mapováním na klávesnici, transformovat na slova, je zapotřebí použít jazykový model, jehož funkce je blíže popsána v kapitole 4.1. Pro vytvoření odpovídajícío modelu je zapotřebí získat korpus (datovou sadu). Ten obvykle obsahuje texty, ze kterých se vyextrahují slova a jejich četnosti. Na základě této sady je poté model trénován.

Vhodný korpus, který byl dostupný zdarma, pro trénování mého jazykového modelu s již vyextrahovanými daty jsem získal z osobních stránek autora Hermita Dava[4]. Ten jej získal zpracováním textů z veřejné databáze titulků. U korpusu jsem následně provedl jeho úpravu (odstranění slov obsahujících nelegální znaky a odstranění jednopísmenných slov). Jazykový model natrénovaný na tomto korpusu tak zná pouze slova obsažená v něm. Tento korpus po odstranění nevalidních znaků ovšem není dostatečně velký a k reálnému nasazení nástroje by bylo zapotřebí použití větší datové sady, která ovšem není volně k dispozici. Jen taková sada poskytne dostatečně rozmanitý seznam slov, který bude možné použít při vyhodnocování na reálných datech. Protože se v práci budu pouze pokoušet o vytvoření nástroje pro psaní pohledem a není kladen důraz na jeho reálné nasazení, zvolil jsem jednodušší přístup, kterým je rozpoznávání izolovaných slov. K tomu jsem zvolil aproximaci unigramovým modelem. To znamená, že nebudu používat žádný kontext, ale každé slovo má vlastní pravděpodobnost.

Na základě pravděpodobností písmen se model v každém stavu rozhoduje o dalším kroku. V případě málo pravděpodobných písmen tím umožňuje jejich přeskočení. Naopak u písmen více pravděpodobných umožňuje jejich vybrání, popř. několikanásobné vybrání. Bylo proto potřeba zjistit, s jakými pravděpodobnostmi budou operovat jednotlivé symboly (písmena a konstanta `SIL`) nebo skupiny symbolů. Klíčovými prvky celého modelu jsou sekvence symbolů, obsahující $1 - N$ prvků. Jako základní prvky sekvencí mohou být použita například písmena nebo slova.

K vytvoření modelu, který bude implementovat výše zmíněná pravidla, bylo potřeba najít vhodnou strukturu, která by jej reprezentovala. Při prozkoumávání možností použití jazykových modelů jsem narazili na případy, které se využívají při rozpoznávání řeči.

[4] http://invokeit.wordpress.com/frequency-word-lists/

Ty využívají pro modelování jednotlivých slov fonémové[5] předpisy. Jednotlivé fonémy pak slouží jako základní jednotka pro stavbu slov. V tomto způsobu zpracování jsem nalezl jistou podobnost s mým problémem. Pokud dokážu nahradit fonémy za písmena, budu moci využívat metody a nástroje určené k rozpoznávání řeči. Na základě tohoto poznatku budu při vytváření modelů vycházet ze struktur a modelů, které se při zpracování řeči využívají. Při hledání odpovídající reprezentace proto budu vycházet ze struktur automatů a jejich variant.

Jednou z jejich variant jsou Markovovy Modely. V případě zpracování řeči jejich varianta Skrytých Markovových modelů (HMM). Jejich popis a definici lze nelézt v podkapitole 4.2. Výhodou modelů při zpracování řeči je jednoduché trénování a snadná použitelnost. V praxi se pro každé slovo natrénuje jeden model a pro rozpoznání sekvence slov se provede konkatenace individuálně natrenovaných modelů pro izolovaná slova. Sekvence tak po průchodu modelem získají skóre na základě kterého se stanoví výsledek. Jednou z charakteristik těchto modelů je to, že na přechodech mezi stavy se nachází jeden symbol. Dále v tomto směru jsou WFST, které jsou popsány v kapitole 4.3. Ty obsahují na přechodu jak vstupní, tak výstupní symbol. Přechod slouží k překladu mezi těmito symboly.

Práce se strukturami je možná za pomocí dostupných toolkitů. Z dostupných variant jsem si vybral Kaldi [18]. Kaldi je toolkit pro rozpoznávání řeči. Obsahuje nástroje pro trénování modelu, testování a také dekódování. Jeho výhodou je publikování pod otevřenou licencí, jež umožňuje modifikaci. Kaldi využívá pro vytváření modelů WFST, které implementuje pomocí knihovny OpenFst[6]. Algoritmy pro práci nad WFST strukturami jsou dostupné včetně metod pro determinizaci, minimalizaci a weight-pushing algoritmu pro optimální redistribuci vah. Jazykový model se v tomto nástroji vytváří po částech. Celkově se vytváří 4 specifické části, jejichž funkci blíže rozebereme v kapitole 6.3. Výsledný model se získá konkatenací dílčích modelů. [13]

Výstupními daty při dekódování v Kaldi je sekvence slov, kterou dekodér označil za nejpravděpodobnější. Abych mohl změřit, jak je nástroj přesný a provést potřebné experimenty, bude potřeba připravit vhodná vstupní data. Ideální možností je získání reálných dat. V počátečních fázích práce jsem se proto rozhodl vytvořit vlastní nahrávací zařízení a to použít pro získání dat. Přesnost nástroje ovšem neodpovídala mým potřebám a řešení nebylo využito. Protože nástroj bylo potřeba důkladně ověřit, přistoupil jsem ke generování umělých dat. Ty bylo možno lépe přizpůsobit konkrétním požadavkům a odhalit tak slabé stránky systému. Konkrétně jsem se zaměřil na práci se šumem a v kapitole 5.1 zmíněnými falešnými fixacemi.

[5] Foném je nejmenší zvukovou jednotkou řeči, která má rozlišovací schopnosti v konkrétním jazyku.
[6] http://www.openfst.org

Kapitola 6

Realizace a vyhodnocení

Pro implementaci výše zmíněných modelů a postupů byl použit počítač s OS Linux a distribucí Ubuntu ve verzi 12.04. Použité zrojové kódy využívají toolkit Kaldi, který je napsán v jazyce C++. Jeho akutální kopie byla získána z SVN repozitáře projektu[1]. Pro chod systému je podmínkou mít tento toolkit k dispozici. Námi vytvořené skripty pro obsluhu toolkitu využívají skriptovací jazyk Bash. Při práci nad textovými daty a jejich zpracování byl použit jazyk Python ve verzi 2.7, s knihovnou Scipy[2], umožňující rozšířené statistické operace. K vytvoření rozhraní EyeTrackeru byla využita knihovna Pythonu TkInter[3], umožňující tvorbu grafického uživatelského rozhraní a open source ITU Gaze Tracker [21, 20]. V následujících kapitolách rezeberu celý proces přípravy, tvorby a testování nástroje. V první podkapitole 6.1 se budu věnovat pokusu o tvorbu vlastního Eye Trackeru. Ten měl původně sloužit k testování celého nástroje. Sekce 6.2 se zabývá extrakcí informací ze zdrojových dat a vytváření příznakových vektorů. Poslední fází přípravy nástroje je tvorba jazykového modelu. Tomu se věnuji v podkapitole 6.3. Na závěr je uvedena sekce 7, která se zabývá tvorbou a úpravou tesovacích dat. Pro práci s mnou vytvořenými nástroji je zapotřebí funkční toolkit Kaldi a správně nastaveny cesty ke všem jeho nástrojům. Do složky s experimenty, která se nachází po stažení z SVN v `kaldi-trunk/egs/rm/`, se umístí projektový adresář. Struktura projektu v hlavním adresáři je následovná:

- Adresář `dataset/`, obsahující trénovací data a mapování klávesnice.
- Adresář `data/`, ve kterém se nachází výsledný automat a další soubory potřebné pro dekódování, které Kaldi vytvoří za běhu.
- Adresář `scripts/` s dílčími skripty, které se používají při tvorbě modelu.
- Skript `makeFst.sh`, který vytvoří kompletní model včetně přípravy všech zdrojových souborů.
- Skript `prepData.sh`, sloužící k dekódování reálných dat.
- Skript `prepRandom.sh`, který slouží k vygenerování náhodné sekvence dat a jejího dekódování.

[1] `svn://svn.code.sf.net/p/kaldi/code/trunk` – větev kaldi-trunk
[2] `http://www.scipy.org`
[3] `http://docs.python.org/2/library/tkinter.html`

6.1 EyeTracker

K snazšímu provádění experimentů na reálných datech jsem se pokusil nad rámec práce vytvořit vlastní Eye Tracker. Byl založen na upravené PlayStation Eye kameře[4], která umožňuje nahrávání záznamu o rozměrech 640 x 480 pixelů s frekvencí snímků 60 Hz. U kamery byl odebrán původní objektiv a místo něj byl použit objektiv s větší ohniskovou vzdáleností, propouštějící infračervené světlo. Nový objektiv nám umožnil umístění kamery dále od uživatele. Vytvořili jsme tak table mounted tracker. Tracker tohoto typu ale potřebuje ještě další informaci, podle které je schopen zjistit pohyby hlavy vzhledem k okolí. K tomu jsem použil další, pro nahrávání potřebnou komponentu. Tou byl bodový zdroj infračerveného světla, který byl využit pro nasvícení oka uživatele. K vyhodnocování pohledu byl použit open source ITU Gaze Tracker, vyvíjený na University of Copenhagen. Pro dosažení co nejlepších výsledků nástroje bylo použito nastavení s jednou kamerou a jedním zdrojem světla. Nástroj při nastavení jako table mounter zařízení sám detekoval zornici a odlesk bodového světla v oku. Na základě získaných dat pak prováděl transformaci na pohyb kurzoru myši.

K otestování přesnosti trackeru byl vyvinut program, který na monitoru zobrazil plnou QWERTY klávesnici a nahrával pohyb kurzoru myši (pohledu). Uživateli rozhraní vypsalo vždy jedno slovo z předem vytvořené sady a uživatel měl za úkol slovo pomocí pohledu napsat. Pro snazší oddělení sekvencí bylo vždy mezi slovy použito doprovodné gesto, kterým byl stisk mezerníku na fyzické klávesnici pro jasné označení začátku a konce slova. Takto získaná data jsem poté zpětně namapoval na klávesnici a kontroloval shodu se slovy tak, jak bylo po uživateli požadováno. Vyhodnocováním dat jsem zjistil, že nástroj nemá dostatečnou přesnost pro použití v našem nástroji. Největší problémy způsobovala kalibrace, která už při počátečním nastavení neposkytovala přesné výsledky. V rozhraní byly označována okolní písmena namísto těch, na které uživatel upřel pohled. S přibývajícím časem se tato odchylka od přesného bodu pohledu jen zvětšovala. Z výše uvedených důvodů jsem od řešení nakonec upustil a zvolil jsem generování umělých dat.

6.2 Extrakce příznakového vektoru

Protože data generována pomocí vlastního Eye Trackeru jsem nebyl schopen získat, bylo zapotřebí vygenerovat data jiným způsobem. K tomu jsem využil toolkit Kaldi. Ten po vytvoření struktury jazykového modelu umožňuje pomocí nástroje `randfst` projít automat po náhodně cestě a vytvořit tak sekvenci stavů. Tuto sekvenci lze poté zpětným mapováním na klávesnici převést na souřadnice. K tomu je ale potřeba uměle vytvořit rozvržení klávesnice. Použil jsem prostor o velikosti 1440 x 560 pixelů. Do prostoru jsem podle zvyklostí klávesnic typu QWERTY umístil klávesy, respektive jejich středy. Z výsledných informací jsem následně postupem uvedeným v sekci 5.1 získal mapu klávesnice. V této fázi již bylo možné provádět reverzní mapování. To probíhalo postupným průchodem sekvencí vygenerovanou pomocí `randfst`. Každý vzorek sekvence odpovídá stavu, ve kterém se model nachází v čase t. Tvar sekvence je uveden na obrázku 6.1. Jednotlivé, pomlčkou oddělené položky, vyjadřují číslo písmene, stav ve kterém se model nachází a zvolený přechod, přes který model prošel (mapování přechodů a stavů je znázorněno na obrázku 6.1 pro přechod s číslem 0 nebo 6.2 pro 1). Všechna vygenerovaná slova jsou zde brána jako jediná sekvence. Nástroj ale pracuje s pravděpodobnostmi izolovaných slov a je postaven na unigramových modelech.

[4] http://us.playstation.com/ps3/accessories/playstation-eye-camera-ps3.html

Proto je potřeba jednotlivá slova oddělit do vlastních sekvencí, kde každá obsahuje pouze izolované slovo. Analýzou každého stavu v sekvenci jsem schopen určit, nad kterým písmenem se model nachází. Platí, že číslo písmene značí písmeno, nad kterým se model nacházel. Číslo stavu značí, zda-li se model nacházel nad konkrétním písmenem nebo symbolem SIL. 0 zde značí konkrétní písmeno a 1 stav `SIL`. Podle těchto údajů jsem tedy schopen říci, jestli model zapisuje písmeno, v tom případě se nachází nad klávesou nebo přeskakuje písmeno. V případě, že přeskakuje, pak počítám s konstantou. Číslo přechodu je pro detekci písmene nepodstatné. Slouží nám zde pouze k ověření správnosti modelu. Následně je možno vygenerovat konkrétní souřadnice (x, y) pro daný čas. Generování je realizováno pomocí modulu `math` jazyka Python, jenž obsahuje funkci pro generování pseudonáhodných čísel s normálním rozložením. Jako střed normálního rozložení jsem zvolil souřadnice středu klávesy, které jsem si před spuštěním navrhl. Parametr σ, vyjadřující směrodatnou odchylku, je možné měnit podle požadavků jednotlivých experimentů. To umožňuje přidání šumu do generovaných dat a provádět tak experimenty s jeho různými úrovněmi. Menší hodnota, která nepřesahuje za okraj klávesy, bude generovat přesnější data. Naopak hodnota větší bude generovat data, která budou obsahovat více šumu. V případě, kdy chceme do modelu přidat falešné fixace, jsou do generovaného vektoru náhodně vloženy další vzorky. Ty obsahují libovolné souřadnice, které se nachází v prostoru klávesnice. Pro reálnější simulaci je vybrán jeden bod a okolo něj je vytvořeno několik vzorků. Tímto postupem je možno získat stejný format dat jako při nahrávání na trackeru. Detailní popis tohoto postupu je k nalezení v sekci 7. Výstupem je potom vektor obsahující hodnoty (x, y), vyjadřující pozici pro každý čas. Totožný vektor také získám z dat reálných, proto u nich nebylo potřeba provádět operace, jako s daty umělými.

8	9	15_0_2	<eps>
9	10	15_1_1	<eps>
10	11	15_1_1	<eps>
11	12	15_1_2	<eps>
12	13	eps	<eps>
13	14	6_0_3	<eps>
14	15	eps	next
15	16	25_0_1	<eps>
16	17	25_0_3	<eps>
17	18	21_0_1	<eps>
18	19	21_0_3	<eps>

Tabulka 6.1: Sekvence stavů vygenerovaná nástrojem `randFst` pro slovo *next*.

Předtím, než data budu moci dekódovat, musím spočítat odezvu dat na komponentách. K tomu jsem využil mixture model. Konkrétně Gaussian Mixture Model v jeho multidimenzionální formě (MGMM). Pro realizaci výpočtu Gaussovských komponent tohoto modelu se v Pythonu využívá knihovna Scipy. Jejími parametry jsou středy kláves, směrodatná odchylka a samotná souřadnice. První dva zmíněné parametry získáme z mého modelu klávesnice. Jedinou výjimku tvoří symbol `SIL`, jehož hodnota je určena konstantou. Abych mohl s daty dále v dekodéru pracovat je třeba zajistit, aby suma všech odezev byla rovna 1. K dosažení této skutečnosti mi poslouží Bayesův teorém. Zabývá se podmíněnými pravděpodobnostmi jevů. Jedná se o vyjádření pravděpodobnosti výskytu jevu A, za předpokladu výskytu jevu B. Tento vztah může být vyjádřen rovnicí:

$$p(W|L) = \frac{P(L|W)P(W)}{P(L)} \tag{6.1}$$

Jelikož v našem případě neuvažujeme apriorní pravděpodobnosti, které jsou v teorému označeny jako $P(W)$, je výpočet pouhou normalizací hodnot na interval $\langle 0, 1 \rangle$.

Výsledně získaný vektor příznaků je ve formě matice s C sloupci a N řádky, kde C je počet všech symbolů modelu a N je sekvence prvků, které byly nahrány pro každý čas t.

Než tento vektor, respektive matici, vložíme do dekodéru, je zapotřebí jej ještě upravit do interní reprezentace, které Kaldi vyžaduje. Kaldi používá pro uvedený účel vlastní knihovnu `Kaldi Matrix Library`. Práce s daty zahrnující předchozí kroky, byla realizována ve skriptech napsaných v jazyce Python. Proto i ukládání do matice ve správném formátu jsem v rámci těchto skriptů implementoval. Zpočátku bylo potřeba rozkódovat libovolnou matici, vytvořenou pomocí knihovny `Kaldi Matrix Library` a zjistit její strukturu (viz. obrázek 6.2). Na základě získaných znalostí už zbývalo pouze vytvořit soubor s tímto formátem. Jelikož se pokouším rozpoznávat jednotlivá slova, tak i slova v této matici byla rozdělena do sekvencí. Pravděpodobnosti v matici musely být ještě před uložením převedeny na přirozený logaritmus ln, protože s tímto tvarem Kaldi pracuje.

```
seq  [
     0  0  0
     0  0  0
     0  0  0  ]
```

Tabulka 6.2: Formát matice příznaků, které využívá Kaldi.

6.3 Vytvoření jazykového modelu

Než jsem přistoupil k samotné tvorbě modelu, připravil jsem si několik souborů. Jejich generování zajišťuje skript `utils/prepDependencies.py`. Soubory se po vytvoření nalézají ve složce `data/` a jejich funkce je následná:

- phones.txt – seznam obsahující všechny používané symboly
- words.txt – seznam všech použitých slov
- lexicon.txt – seznam všech slov a symbolů, ze kterých se slova skládají
- topo.proto – soubor obsahující topologii symbolů
- input.arpa – pravděpodobnosti výskytu slov

Korpus pro daný model může být zadán jako parametr skriptu `makeFst.sh`. Korpus je při tvorbě modelu rozparsován a je z něj získán seznam všech písmen, slov a četností, se kterými bude model pracovat. Seznam písmen byl seřazen a očíslován vzestupně od 0. Na začátek seznamu jsem přidal povinný symbol `<eps>`, který značí prázdný stav. Za něj byl přidán symbol `SIL`, jenž vyjadřuje stav, ve kterém nezapisuji žádné písmeno. Jelikož může

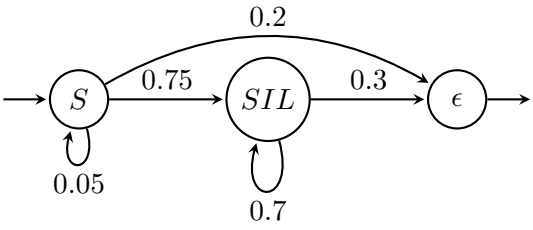
existovat několik slov, které se stejně píší, ale mají různý význam, přidám ještě disambiguační symboly. Ty umožňují rozlišení stejných slov. Disambiguační symboly mají tvar `#X`, kde `#` symbol uvozuje a `X` je unikátní číslo symbolu v rámci stejných slov. Seznam slov je taktéž uložen do souboru a očíslován vzestupně od 0. Následujícím krokem je vytvoření lexikonu, jenž obsahuje seznam všech slov a písmen, ze kterých se slovo skládá. V lexikonu je tedy každé slovo rozděleno po písmenech, a v případě dvou stejných slov, je přidán disambiguační symbol. Tímto jsou připraveny všechny seznamy a zbývá už pouze určit pravděpodobnosti, se kterými bude model pracovat. Pravděpodobnosti výskytu slov se určily z korpusu, který se skriptu `makeFst.sh` předává jako parametr. Z něj jsou vyextrahována všechna slova a určena jejich četnost. Do souboru jsou uložena postupně všechna slova ze seznamu a k nim odpovídající pravděpodobnosti sumované do 1. Pravděpodobnosti jsou ještě před uložením přepočteny na log_2, neboť Kaldi pracuje s pravděpodobnostmi v tomto tvaru. Posledním krokem, nutným pro spuštění procesu výroby modelu, je vytvoření topologie pro jednotlivé symboly. Tu jsem si stanovil již při plánování. Struktura topologie je na obrázku 6.1 a 6.2. Pravděpodobnosti jednotlivých přechodů jsem experimentálně určil při ladění modelu.

Ze získaných dat jsem tedy schopen začít vytvářet model. Vytváření modelu se provádí pomocí skriptu `makeFst.sh` v hlavní adresářové složce. Ten nejprve vytváří čtyři dílčí WFST modely, které je poté možno různými způsoby komponovat do výsledné struktury. Prvním z nich je gramatika G, která obsahuje seznam všech slov daného modelu. Jedná se o součást jazykového modelu na úrovni slov. Dalším z nich je lexikon L, jenž obsahuje všechna slova a sekvenci fonémů, ze kterých se skládají. Vytváří model pro každé slovo. Odpovídá tedy jazykovému modelu na úrovni fonémů. Při kompozici těchto modelů pak WFST struktura přijímá na úrovni lexikonu fonémy a ty transformuje na slova. Gramatika následně funguje jako akceptor. To znamená, že vezme výstup lexikonu a pouze akceptuje určité kombinace slov s příslušnou pravděpodobností. Následuje model označován jako C, který umožňuje použití kontextově závislých fonémů. Jak z názvu vyplývá, používají se k rozpoznávání fonémů v závislosti na okolním kontextu. Posledním modelem je model H, který za pomocí HMM umožňuje modelování jednotlivých fonémů a odpovídá akustickému modelu. Kompozicí čtyř výše uvedených modelů za využití minimalizace (označení min) a determinizace (označení det), získáme výsledný model. Tento proces je popsán vztahem,

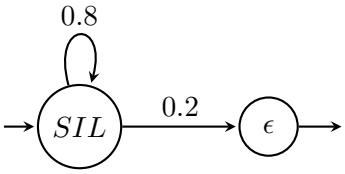
$$HCLG = (\, min\,(\, det\,(\, H' \circ min\,(\, det\,(\, C \circ min\,(\, det\,(L \circ G\,)\,)\,)\,)\,)\,)\,) \tag{6.2}$$

Uvedeným způsobem se využívá Kaldi při tvorbě modelů pro rozpoznávání řeči. Pro mé použití byl model upraven. Kaldi využívá foném jako prvek, kterým jsou tvořena slova. V mém případě budou slova tvořena písmeny. Foném tedy bude odpovídat písmenu. Gramatika G, bude stejně jako u řečových modelů, obsahovat seznam slov. V této části budou modely stejné. Rozdílné budou použité modely lexikonu L. Ten pro mé účely bude mapovat slova na písmena namísto fonémů. Kontextově závislé fonémy pro dosažení požadovaného výsledku potřebují velkou datovou sadu, ze které se mohou trénovat. Ta v době dokončování práce nebyla k dispozici, a proto model C pouze předává vstupní sekvence na výstupní, aniž by je nějak měnil. Drobný rozdíl se vyskytne v modelu H, který modeluje jednotlivé písmena. U něj je třeba implementovat možnost vynechání písmene při chybné fixaci. Zde nalezne využití výše zmíněný symbol `SIL`, který bude zahrnut do seznamu písmen a akustický model. Přeskakovaní docílím tak, že každé písmeno, vyjma symbolu `SIL`, budu modelovat jako tří-stavový levo-pravý HMM (viz obrázek 6.1). V něm bude první

stav reprezentovat konkrétní písmeno. V druhém stavu se bude vždy nacházet symbol `SIL` s vlastním dvou-stavovým levo-pravým HMM (viz obrázek 6.2) a koncový stav.



Obrázek 6.1: HMM písmena v akustickém modelu.



Obrázek 6.2: HMM symbolu SIL v akustickém modelu.

Z takto vytvořených modelů poté kompozicí podle vztahu 6.2 vytvoříme výsledný HCLG graf.

Dekódování umělých dat probíhá spuštěním skriptu `prepRand.sh`, jenž vytvoří náhodná data a provede nad nimi proces dekódování a následně evaluace. Skript se volá s jedním volitelným parametrem, určujícím násobek standardní odchylky. Skript se přitom stará pouze o získání dat a jejich převod do správného formátu. O dekódování samotné se stará vestavěný nástroj Kaldi. Pro vyhodnocování reálných dat je k dispozici skript `prepData.sh`, který vyžaduje dva parametry s cestou, kde se nalézá mapování klávesnice a samotná nahraná data. Výstup dekodéru je ve tvaru:

`function`	`<s>`	`function`	`</s>`
`beloved`	`<s>`	`beloved`	`</s>`
`warren`	`<s>`	`water`	`</s>`

Tabulka 6.3: Výstup dekodéru se vzorovým slovem a nalezenou sekvencí.

První sloupec značí název sekvence a tedy i slovo, které bylo opravdu napsáno. V dalším sloupci následuje znak <s> pro začátek sekvence. Následuje dekódované slovo a znak </s> pro konec sekvence. Stejný formát výstupu mají jak data nahraná uměle, tak data získaná ze skutečného trackeru. Skript `prepRandom.sh`, který se o dekódování stará, provádí rovnou vyhodnocování výstupních sekvencí z dekodéru, kdy pouze spočte, kolik vzorků bylo určeno špatně a kolik dobře. Na základě toho stanoví procentuální přesnost daného měření.

Kapitola 7

Experimenty

Pro ověření možností nástroje byla provedena sada experimentů. Původním záměrem bylo otestovat nástroj na reálných datech nahraných na vlastním trackeru. Ty se mi ale nepodařilo získat, kvůli chybě kalibrace nástroje, a proto jsem se rozhodl jej nepoužít. Nicméně nástroj pro jejich rozpoznávání byl vytvořen a v případě otestování funkčnosti může být při dalších experimentech použit. Experimenty byly prováděny nad uměle vytvořenou sadou. Vyhodnocení možností nástroje u jednotlivých experimentů pak spočívalo ve srovnání poměru správně a špatně rozpoznaných sekvencí. Pro experimenty byly vytvořeny 4 různé testovací sady. Výsledky experimentů jsou k nalezení v tabulce 7.1.

Experimenty jsem pro uměle vytvořená data rozdělil do několika kroků, ve kterých jsem se zaměřil na některé vlastnosti, které lze pozorovat u dat reálných:

- Uměle získaná data bez přidaných vlastností.
- Sekvence simulujících přesun pohledu mezi písmeny.
- Sekvence pro chybné fixace.
- Sekvence značících chybné fixace a sekvencí simulujících přesun pohledu.

V prvním experimentu jsem vygeneroval náhodná data pomocí nástroje `randFst`. Ty jsem bez dalších úprav předal dekodéru a provedl vyhodnocení. Tento experiment ve své podstatě simuluje super-uživatele, který přesun mezi písmeny provádí tak rychle, že jej nahrávací zařízení nezaznamená. Zároveň vždy najde požadované písmeno ihned. Data pro tento experiment obsahovala pouze několik fixací (přibližně 1 až 6) na požadovaném písmenu, popřípadě v jeho blízkém okolí díky šumu. Experiment s touto datovou sadou pouze sloužil k základní verifikaci modelu.

Experiment následující vycházel z datové sady vytvořené v prvním experimentu. Navíc obsahoval pouze fixace vytvořené při přesunu pohledu na další písmeno. Vytváření fixací spočívalo v nalezení cesty mezi předposlední a poslední zapsanou klávesou. Principem bylo odečtení x-ových a y-ových souřadnic písmen a získání vzdáleností. Následně se pomocí generátoru pseudonáhodných čísel s rovnoměrným rozložením vygenerovalo celé číslo z intervalu $< 0, 3 >$, které značilo počet kroků, ve kterých dojde k zaznamenání hodnot na této cestě. Souřadnice fixací se podařilo získat tak, že jsem vzdálenosti na obou osách x, y pouze vydělil získaným počtem kroků a odpovídající hodnoty zanesl do výsledné sekvence jako další informaci. V tomto experimentu také byla implementována vlastnost, kdy uživatel při přesunu pohledu na požadované písmeno neví kde se nachází, a tak jej hledá. Toho bylo

dosaženo vygenerováním celého čísla pomocí rovnoměrného rozložení z intervalu $< 0, N >$, kde $2 <= N <= 4$. Přičemž hodnota N je závislá na vzdálenosti kláves, mezi kterými uživatel přesouvá pohled. Platí, že čím větší je vzdálenost, tím déle bude symbol hledat a tím více vytvoří chybných fixací. Rozsah intervalů byl přibližně stanoven na základě zkušeností při vývoji vlastního Eye Trackeru a tím, jak rychle nástroj zvládne zaznamenávat pozici pohledu.

Přidání chybných sekvencí spočívalo v náhodném vygenerování několika souřadnic na klávesnici a vytvoření fixací v okolí vygenerovaného bodu. Protože takto získané fixace umožňují, aby se uživatel podíval několikrát do stejného místa (vygenerované body mohou být blízko sebe), bylo potřeba vymyslet lepší způsob, jakým můžu reálné řešení aproximovat. Rozhodl jsem se proto vytvořit dvourozměrnou matici, ve které odpovídá každý prvek jednomu pixelu klávesnice a jeho hodnota je ve výchozím stavu nastavena na 0. Skript poté náhodně vybere souřadnici na klávesnici a prohledá její okolí. V případě, že se v okolí vyskytuje požadované písmeno, které uživatel chce zapsat, tak jej vybere a matici resetuje. Pokud ne, tak do matice zapíše odezvu Gaussovy funkce pro rozdělení hustoty pravděpodobnosti pro každý bod (prvek matice), kde střed rozdělení leží na vygenerované souřadnici a standardní odchylka byla experimentálně přibližně určena jako vzdálenost 1,5 klávesy (uživatel periferně vidí a rozeznává klávesy přibližně v tomto rozsahu). V případě, že další vygenerovaná souřadnice bude v místě, kde již byla dříve zapsána nějaká odezva (hodnota větší než 0), jsou podle velikosti odezvy vygenerovány 0-3 fixace v jejím okolí. Platí, že čím menší odezva v daném bodě je, tím více fixací se vygeneruje. Tento přístup tak umožním simulaci pohledu reálného uživatele, který se většinou nepodívá 2-krát na stejné místo a zároveň zohledňuje periferní vidění.

Poslední experiment zahrnoval spojení všech předchozích, nejlépe tedy odpovídal reálným datům.

Datová sada	**Experiment**	**Správně**	**Chybně**	**Přesnost**
1. sada	1. Bez přidaných vlastností	4	24	**14.29 %**
	2. Přesun mezi symboly	1	27	**3.57 %**
	3. Chybné fixace	0	28	**0 %**
	4. Kombinace výše uvedených	0	28	**0 %**
2. sada	1. Bez přidaných vlastností	4	40	**9.09 %**
	2. Přesun mezi symboly	5	39	**11.36 %**
	3. Chybné fixace	0	44	**0 %**
	4. Kombinace výše uvedených	0	44	**0 %**
3. sada	1. Bez přidaných vlastností	2	11	**15.38 %**
	2. Přesun mezi symboly	1	12	**7.69 %**
	3. Chybné fixace	0	13	**0 %**
	4. Kombinace výše uvedených	0	3	**0 %**

Tabulka 7.1: Výsledné přesnosti nástroje pro jednotlivé experimenty.

Výsledky uvedené v tabule 7.1 ukazují, že nástroj není schopen rozpoznávání dat s reálnými vlastnostmi. Z podrobnějších výsledků vyplývá, že největším problémem bylo zmíněné přeskakování chybných fixací a to i přesto, že symbol `SIL` měl pravděpodobnost výskytu několikanásobně vyšší, než jiné slova a písmena. Typický příklad chyby, kterou nástroj produkoval, je možno vidět v tabulce 7.2, kde bylo chybně označeno slovo `myself`. Klávesy,

které jsou blízko sebe v kombinaci s velkým počtem chybných fixací měly za následek vybrání více slov ležících v blízkosti chybných fixací. Pro nasazení a reálné používání tak nástroj v této podobě není vhodný.

`myself`	`<s>`	`have`	`without`	`</s>`

Tabulka 7.2: Příklad chybně vyhodnocené sekvence.

Zmíněné chyby by bylo možné odstranit několika způsoby. Všechny jsou ale založeny na dostatečně velké datové sadě, která bude obsahovat data nahraná pomocí trackeru. S takovou sadou by bylo možné provést trénování modelu a stanovení všech pravděpodobností mnohem citlivěji a přesněji. Model natrénovaný na takové sadě by pak mohl vykazovat lepší výsledky. Na dobré a velké datové sadě by pak bylo možné například použít kontextový model, který je schopen zohlednit i pravděpodobnosti výskytu fixací, jak jdou za sebou.

I když nebylo nahrávání s reálnými daty uskutečněno, byl připraven skript `./prepData.sh` pro jejich vyhodnocování. Nahraná data byla získána tak, že v každém kroku byly na nahrávacím zařízení změřeny souřadnice pohledu zvlášť pro levé a pravé oko a byly zaneseny do Kartézské soustavy souřadnic. Jelikož v bakalářské práci neuvažuji uživatele s očními vadami, při kterých je narušena kooperace očí (např. strabismus), můžu směr pohledu spočítat jako aritmetický průměr pravého a levého oka pomocí vztahů,

$$x = \frac{x_{left} + x_{right}}{2}$$

$$y = \frac{y_{left} + y_{right}}{2}$$

Uvedenou korekcí jsem získal vektor uspořádaných dvojic (x, y), jež už může nástroj klasickým způsobem zpracovávat.

Kapitola 8

Závěr

Cílem bakalářské práce bylo sestavení nástroje, který by umožňoval psaní pohledem pomocí metody Swype. Nástroj měl být zpracován na základě poznatků uveřejněných v článku *The Potential of Dwell-Free Eye-Typing for Fast Assistive Gaze Communication* [8]. Specifickým znakem nástroje a také jeho hlavní výhodou při použití metody Swype je to, že nepotřebuje pro zapsání znaku nutnost zaměření (dwell-free).

Jelikož v době vzniku bakalářské práce neexistoval žádný podobný nástroj využívající metody Swype, bylo zapotřebí přijít s vlastním přístupem. Ten byl v této práci založen na nástrojích a strukturách, které se využívají při zpracování řeči. To umožňovalo použití postupů a modelů, které se při řešení řečových úloh využívají a jejich modifikací dostat plnohodnotný jazykový model odpovídající požadavkům.

K tomu byl využit toolkit Kaldi využívající WFST. Na jeho základě se podařilo sestavit model, který je schopen nést informace potřebné pro psaní pohledem. K práci s modelem byla vytvořena sada skriptů, umožňujících provádění specifických úloh, které se od běžných postupů, jakými toolkit disponuje liší. Na závěr byla vytvořena relativně přesná datová sada, která alespoň zčásti umožňovala simulování dat reálných.

Experimenty, na kterých byla funkce nástroje ověřována, byly prováděny, jak už bylo uvedeno, na uměle vytvořených datech. Jejich výsledky ovšem nebyly uspokojivé. V případě experimentů bez přidaných vlastností se podařilo získat alespoň malou úspěšnost, ale i ta je pro reálné použití nedostačující. Data s touto přesností totiž nebude běžný uživatel schopen generovat. Data s uměle přidanými vlastnostmi, která měla aproximovat data reálná lepším způsobem, dosahovala výsledků mnohem horších nebo žádných. I přesto, že byl model ve své podobě co nejlépe nastaven, nedařilo se přeskakovat písmena v takové míře, která byla potřeba. Dekodér pak namísto izolovaných slov nacházel především sekvence slov a to i přesto, že pravděpodobnosti pro dekódování izolovaných slov byly nastaveny dostatečně vysoko.

I když se nástroj pro psaní podařilo za pomoci řečových modelů vytvořit, k jeho reálnému nasazení by bylo zapotřebí dalších modifikací a rozšíření. Hlavním nedostatkem nástroje je špatné přeskakování chybných fixací, které i v případě nastavení velmi vysokých přechodových pravděpodobností nedosahují požadovaných výsledků. Jako možné řešení, které ale v rámci práce nebylo implementováno, je použití kontextových závislostí mezi jednotlivými písmeny. K tomu, abychom mohli k uvedenému řešení přistoupit by bylo zapotřebí velkého množství reálných dat, na kterých by bylo možné kontexty natrénovat. Dalším přístupem, který by mohl pomoci zvýšit úspěšnost nástroje, je trénování modelu taktéž na základě reálných dat. Tím bychom mohli model ideálně nastavit oproti současnému způsobu, kdy jsou pravděpodobnosti v modelu nastavovány ručně na základě experimentů. Obě

řešení ale vyžadují dostatečně velkou datovou sadu. Ta měla být dodána externě, nicméně i v době dokončování práce nebyla k dispozici.

Literatura

[1] Quadriplegic Sets New Guinness World Records Achievement for Fastest Hands-Free Typing. 3 January 2011.
URL http://www.swype.com/wp-content/uploads/Swype_Press_Release_7.pdf

[2] Croft, W. B.; Lafferty, J.: *Language Modeling for Information Retrieval*. Norwell, MA, USA: Kluwer Academic Publishers, 2003, ISBN 1402012160.

[3] Do, C. B.: The Multivariate Gaussian Distribution. 2008.

[4] Duchowski, A.: *Eye Tracking Methodology: Theory and Practice*. Methods in molecular biology, Springer-Verlag London Limited, 2007, ISBN 9781846286094.

[5] Dymarski, P.: Hidden Markov Models, Theory and Applications. *InTech Open Access Publishers*, 2011.
URL http://pure.rhul.ac.uk/portal/files/1446635/Hidden_Markov_Models_Theory_and_Applications.pdf

[6] Gales, M.; Young, S.: The application of hidden Markov models in speech recognition. *Found. Trends Signal Process.*, ročník 1, č. 3, Leden 2007: s. 195–304, ISSN 1932-8346, doi:10.1561/2000000004.
URL http://dx.doi.org/10.1561/2000000004

[7] Jacob, R. J. K.; Karn, K. S.: Commentary on Section 4. Eye tracking in human-computer interaction and usability research: Ready to deliver the promises.
URL http://www.cs.tufts.edu/~jacob/papers/ecem.pdf

[8] Kristensson, P. O.; Vertanen, K.: The potential of dwell-free eye-typing for fast assistive gaze communication. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '12, New York, NY, USA: ACM, 2012, ISBN 978-1-4503-1221-9, s. 241–244, doi:10.1145/2168556.2168605.
URL http://pokristensson.com/pubs/KristenssonVertanenETRA2012.pdf

[9] Kushler, C.: AAC: Using a Reduced Keyboard. 1998.

[10] Majaranta, P.; Ahola, U.-K.; Špakov, O.: Fast gaze typing with an adjustable dwell time. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, New York, NY, USA: ACM, 2009, ISBN 978-1-60558-246-7, s. 357–360, doi:10.1145/1518701.1518758.
URL http://doi.acm.org/10.1145/1518701.1518758

[11] Majaranta, P.; Räihä, K.-J.: Twenty years of eye typing: systems and design issues. In *Proceedings of the 2002 symposium on Eye tracking research & applications*,

ETRA '02, New York, NY, USA: ACM, 2002, ISBN 1-58113-467-3, s. 15–22, doi:10.1145/507072.507076.
URL http://doi.acm.org/10.1145/507072.507076

[12] Mohri, M.: Weighted Finite-State Transducer Algorithms. An Overview. In *Formal Languages and Applications*, Springer, 2004, s. 551–563.
URL http://www.cs.nyu.edu/~mohri/pub/fla.pdf

[13] Mohri, M.; Pereira, F. C.; Riley, M.: Speech recognition with weighted finite-state transducers. *Handbook on Speech Processing and Speech Communication*, 2008.
URL http://www.cs.nyu.edu/~mohri/pub/hbka.pdf

[14] Nuance: Swype Press Kits - Nuance.
URL http://www.nuance.com/company/news-room/press-kits/swype-press-kits/index.htm

[15] Oparin, I.: *Language Models for Automatic Speech Recognition of Inflectional Languages*. Dizertační práce, University of West Bohemia, 2008.
URL http://perso.limsi.fr/Individu/oparin/Thesis.pdf

[16] Ostrach, T. R.: Typing Speed: How Fast is Average: 4,000 typing scores statistically analyzed and interpreted. *Five Star Staffing, Inc., Orlando, FL Retrieved September*, ročník 18, 1997: str. 2010.

[17] Poole, A.; Ball, L. J.: Eye tracking in human-computer interaction and usability research: Current status and future. In *Prospects?, Chapter in C. Ghaoui (Ed.): Encyclopedia of Human-Computer Interaction. Pennsylvania: Idea Group, Inc*, 2005.
URL http://www.alexpoole.info/blog/wp-content/uploads/2010/02/PooleBall-EyeTracking.pdf

[18] Povey, D.; Ghoshal, A.; Boulianne, G.; aj.: The Kaldi Speech Recognition Toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, IEEE Signal Processing Society, Prosinec 2011, iEEE Catalog No.: CFP11SRW-USB.

[19] Reynolds, D.: Gaussian mixture models. *Encyclopedia of Biometric Recognition*, ročník 2, č. 17.36, 2008: s. 14–68.
URL http://www.ll.mit.edu/mission/communications/ist/publications/0802_Reynolds_Biometrics-GMM.pdf

[20] San Agustin, J.; Skovsgaard, H.; Hansen, J. P.; aj.: Low-cost gaze interaction: ready to deliver the promises. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '09, New York, NY, USA: ACM, 2009, ISBN 978-1-60558-247-4, s. 4453–4458, doi:10.1145/1520340.1520682.
URL http://doi.acm.org/10.1145/1520340.1520682

[21] San Agustin, J.; Skovsgaard, H.; Mollenbach, E.; aj.: Evaluation of a low-cost open-source gaze tracker. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, ETRA '10, New York, NY, USA: ACM, 2010, ISBN 978-1-60558-994-7, s. 77–80, doi:10.1145/1743666.1743685.
URL http://doi.acm.org/10.1145/1743666.1743685

[22] Tuisku, O.; Majaranta, P.; Isokoski, P.; aj.: Now Dasher! Dash away!: longitudinal study of fast text entry by Eye Gaze. In *Proceedings of the 2008 symposium on Eye tracking research & applications*, ETRA '08, New York, NY, USA: ACM, 2008, ISBN 978-1-59593-982-1, s. 19–26, doi:10.1145/1344471.1344476.
URL http://doi.acm.org/10.1145/1344471.1344476

[23] Ward, D. J.: Adaptive computer interfaces. *Cambridge, University of Cambridge*, 2001.
URL http://www.inference.phy.cam.ac.uk/djw30/papers/thesis.pdf

Příloha A

Obsah CD

Adresářová struktura, která je obsažena na CD je následující:

- `text`
  - `bp.pdf` – bakalářská práce ve formátu pdf
  - `src` – adresář se zdrojovými soubory bakalářské práce
  - `poster` – adresář s plakátem
- `program`
  - `scripts` – adresář obsahující pomocné skripty
  - `dataset` – složka, která obsahuje použitý korpus, mapování klávesnice a vzorové nahrané data, které nebyly použity
  - `data` – adresář pro vytvořené struktury a obsahující předlohu pro některé struktury
  - `makeFST.sh` – skript, který se stará o vytvoření automatu
  - `path.sh` – skript pro nastavení požadovaných cest a proměnných
  - `prepRandom.sh` – skript pro spuštění testů nad umělými daty
  - `prepData.sh` – skript použitý k zpracování reálných nahraných dat
- `gazeKeyboard` – adresář obsahující soubory, které byly použity při vývoji Gaze Trackeru

Příloha B

Plakat

Klávesnice pomocí pohledu

Bakalářská práce

AUTOR: Jakub Sznapka

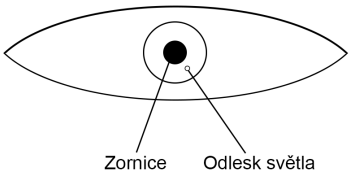
VEDOUCÍ: Ing. Michal Hradiš

VUT FIT 2013

Gaze Tracker

Zařízení, které slouží k snímání pohledu uživatele. Může být buď přímo aplikované na uživatele (oko, obličej), nebo umístěné externě s výhledem na uživatelovo oko.

V praxi se nejčastěji používá snímání za využití kamery, která snímá pohyby oka uživatele. Aby bylo možné eliminovat nežádoucí pohyby hlavy, je oko nasvíceno Infračerveným světlem a při vyhodnocování se počítá vzájmená poloha odlesku světla na rohovce a středu zornice.



Swype

Swype je moderní metoda a zároveň aplikace, která se využívá pro psaní na zařízeních s dotykovými displeji. Při psaní pomocí této metody se vychází z plné QWERTY klávesnice. Psaní textu probíhá tak, že uživatel prstem přejíždí mezi symboly a nezvedá prst. Zvednutí prstu je označeno jako konec slova. Trajektorie pohybu je vyhodnocena a je nabídnuto nejpravděpodobnější slovo. V případě, že uživatel dané slovo nechtěl napsat, mu je k dispozici seznam dalších méně pravděpodobných slov. Pokud ani zde nenajde kýžený výraz, je zde možnost napsat slovo klasickým způsobem, a tím jej přidat do slovníku.

Psaní slova *quick* pomocí metody Swype. Zdroj: (www.swype.com)

Weighted Finite State Transducers

Pro jazykové modelování se využívá různých typů automatů. Speciálním případem jsou Weighted Finite-State Transducers (WFST). Jedná se o automaty, které na každém přechodě obsahují mimo vstupního symbolu také symbol výstupní. Ty se hojně využívají při rozpoznávání zvuku, obrazu, ale také textu. WFST umožňují mapování různých typů informací v rámci jednoho modelu. Lze pomocí nich využít jednoduchou a přirozenou reprezentaci HMM modelů, kontextově závislých modelů, gramatik a dalších.

HCLG = (min(det(H' o min(det(C o min(det(L o G))))))

Kompozice modelů za pomoci minimalizace a determinizace.

Model *G* odpovídá gramatice, která modeluje slova. Lexikon *L* umožňuje modelování jednotlivých slov založených na písmenech. Kontextově závislé modelování písmen pomocí modelu *C* nebylo realizováno. Model *H* umožňuje modelování jednotlivých písmen.

Výsledky

Experimenty s nástrojem byly prováděny nad uměle vytvořenou sadou. Vyhodnocení možností nástroje u jednotlivých experimentů pak spočívalo ve srovnání poměru správně a špatně nalezených sekvencí. Pro experimenty byly vytvořeny 4 různé testovací sady.

- data bez přidaných vlastností **- data s chybnými fixacemi**

- data se simulací přesunu pohledu **- data s chybnými fiaxcemi a přesunem**

Experiment	Přesnost
Bez přidaných vlastností	**12,32 %**
Přesun pohledu	**5,6 %**
Chybné fixace	**0 %**
Kombinace předchozích	**0 %**