Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE

Jan Milota

Vývoj hlasově ovládaných webových her pomocí CloudASR

Ústav formální a aplikované lingvistiky

Vedoucí bakalářské práce: Mgr. Ing. Filip Jurčíček, Ph.D.

Studijní program: Informatika

Studijní obor: Obecná Informatika

Praha 2015

Děkuji panu doktorovi Jurčíčkovi za profesionální vedení této práce a za jeho neutuchající víru v mou schopnost samostatného vývoje.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Název práce: Vývoj hlasově ovládaných webových her pomocí CloudASR

Autor: Jan Milota

Katedra: Ústav formální a aplikované lingvistiky

Vedoucí bakalářské práce: Mgr. Ing. Filip Jurčíček Ph.D., Ústav formální a aplikované lingvistiky

Abstrakt: Cílem práce je navrhnout a vyvinout software pro výuku jazyků hrou za použití webových technologií a čerstvě vznikající CloudASR knihovny. Běžný uživatel provozuje interakci se svým prohlížečem skoro výhradně prostřednictvím myši a klávesnice. Díky softwaru, který tato práce reprezentuje, má nyní uživatel příležitost zabřednout do někdy ne úplně populární výuky jazyka i za pomoci svého hlasu. Což nabízí zmíněné výuce netušené možnosti, obzvláště stran uživatelské interaktivity. Důraz byl kladen na uživatelskou přívětivost, grafickou fidelitu a na kompetitivní aspekt výuky, využívajíc Facebookovou integraci a bodové hodnotící žebříčky.

Klíčová slova: automatické rozpoznávání řeči, ASR, hry, web, HTML5, Javascript

Title: Development of speech enabled web games using CloudASR

Author: Jan Milota

Department: Institute of Formal and Applied Linguistics

Supervisor: Mgr. Ing. Filip Jurčíček Ph.D., Institute of Formal and Applied Linguistics

Abstract: The main goal of this thesis is to design and implement a piece of software for playful language learning, using web technologies and the fresh CloudASR library. A common user interacts with their web browser almost exclusively using a mouse and keyboard. Thanks to the software this thesis represents the user has an opportunity to delve into sometimes unpopular language learning process using his natural voice. This fact presents new and exciting possibilities, mainly regarding user interactivity. A lot of stress has been put to user friendliness, graphical fidelity and to the competitive aspect of language education, exploiting Facebook integration and point-scoring leader boards.

Keywords: automatic speech recognition, ASR, games, web, HTML5, Javascript

Obsah

Úvod **3**

1 Automatic Speech Recognition **5**

1.1 Co je to ASR 5

1.2 Postup rozpoznávání 5

1.3 Reprezentace a zpracování signálu 6

1.3.1 Fourierovská analýza 6

1.3.2 Cepstrální analýza 7

1.4 Zdroje omezení mluvené komunikace 8

1.5 Modelování systému 9

1.5.1 Maximum a Posteriori formulace 9

1.5.2 Jazykové modely 9

1.5.3 Akustické modely 11

1.5.4 Lexikální modely 12

1.6 Hodnocení výstupů ASR 13

2 E-learning jako přenos informace **15**

2.1 Co je to e-learning 15

2.1.1 Proč používat e-learning 16

2.1.2 Proč e-learning nepoužívat 16

2.2 Výuka a počítačové hry 17

2.3 E-learning hrou a crowdsourcing 18

2.3.1 ESP game 18

2.3.2 Duolingo 19

2.3.3 reCAPTCHA 20

2.4 TrogASR jako GWAP 20

3 Použité technologie **21**

3.1 Python 21

3.2 Flask 22

3.2.1 Flask-Assets 23

3.2.2 SQLite3 . . . 24
3.3 JavaScript . . . 24
3.3.1 jQuery . . . 25
3.3.2 transform.js . . . 25
3.4 HTML5 . . . 25
3.5 CSS3 . . . 26
3.6 CloudASR . . . 26
3.7 PyCharm . . . 27
3.8 Několik slov o webových prohlížečích . . . 28

4 Implementace **29**
4.1 Modulární kostra . . . 29
4.2 Templatovací systém . . . 30
4.3 Sprite factory a obrázkový font . . . 31
4.4 Responzivní design . . . 32
4.5 Facebook API . . . 33
4.6 Internacionalizace . . . 34
4.6.1 Zdrojové a cílové jazyky překladu . . . 36
4.7 Návrhové vzory . . . 37
4.7.1 Duck-typing . . . 37
4.7.2 Module pattern . . . 38
4.7.3 Scope cache pattern . . . 38
4.7.4 Singleton pattern . . . 38
4.7.5 Currying . . . 39
4.7.6 MVC pattern . . . 40

Závěr **41**

Seznam použité literatury **42**

Seznam použitých zkratek **46**

Přílohy **48**

Úvod

Základním komunikačním kanálem mezilidské interakce je mluvená řeč. Ačkoli trend v poslední době spěje spíše k separaci této komunikace virtuálními médii, je mluvené slovo i nadále nenahraditelnou součástí socializace a výměny informací. Proč tedy nepřenést schopnost porozumět lidské mluvě i na stroje, jakožto právě na ona separující virtuální média? Současná technologie je k tomu jistě dostačující, výpočetní výkon lidstva roste po zběsilé křivce a technologické know-how po ještě zběsilejší.

Aby tento cíl mohl být dosažen, je kritické mít dobrý ASR (Automatic Speech Recognition) systém. Těchto se pohybuje v éteru mnoho, volně dosažitelných, různé kvality a spolehlivosti. Pro specifické využití v této práci byl zvolen systém `CloudASR`[1], jehož autorem je kolega Ondřej Klejch.

`TrogASR` (Translating Online Game using Automatic Speech Recognition) je, jak možná název napovídá, online hra, zaměřující se na překlad slov a frází z jazyka do jazyka za využití automatizovaného rozpoznávání řeči.

Cílem této práce je návrh a implementace graficky přitažlivé hry pro webové prohlížeče, která poskytne uživatelům další, neotřelou, možnost, jak si vyzkoušet a zdokonalit své jazykové schopnosti. Tato hra pak bude sloužit jako kompetitivitou hnaný motor pro sběr verifikovaných dat pro použitou `CloudASR` knihovnu.

Velký důraz je kladen na grafickou stránku věci; uživatelsky neatraktivní hra nemá valnou naději na šíření mezi uživateli samotnými samovolně. Navíc se na ošklivou aplikaci nikdo nebude dívat po delší časový úsek, než je jedno sezení, a už vůbec nikdo se k takovéto aplikaci nebude vracet. Proto bylo veškeré stylování a malování a navrhování layoutů a přechodů prováděno s výraznou pílí.

Dále je důraz kladen i na zmíněnou kompetitivitu. Každý má touhu vidět své jméno někde vysoko na žebříčku, zanechávaje své přátele na tomto žebříčku daleko pod sebou. Proto byla zapojena i integrace s Facebookovým API, jež přináší kýženého výsledku poměrně nenásilnou formou. Na „Like“ a „Login“ tlačítko jsou uživatelé zvyklí a nebojí se jej extenzivně používat.

Následující text popisuje a dokumentuje vývoj této aplikace a myšlenkové po-

[1] `https://github.com/UFAL-DSG/cloud-asr/`

chody při něm stojící. První kapitola analyzuje využití ASR v e-learningu. Druhá kapitola seznámí čtenáře s využitými technologiemi a frameworky pro vývoj aplikace `trogASR`. Kapitola třetí nabídne čtenáři vhled do řešení a implementace aplikace `trogASR`, do použitých návrhových vzorů a do problematiky s nimi spojené. A konečně, kapitola čtvrtá shrne celou práci.

1. Automatic Speech Recognition

V této kapitole se čtenář seznámí s některými základními paradigmaty ASR, dozví se, jak ASR proces vypadá. Zároveň je zde představen úvodní formalismus problematiky.

Nejprve se čtenář dočte o tom, co to vlastně ASR je, poté jak ASR probíhá. Nahlédne pod pokličku základů zpracování signálu, inherentních jazykových omezení, modelování ASR systémů a nakonec i hodnocení kvality výstupu.

1.1 Co je to ASR

ASR je teoretický model, založený na pravděpodobnostních vlastnostech akustických pozorování, který formalizuje převod mluveného slova do textu. Přeneseně se takto označují i konkrétní systémy a implementace, které tohoto modelu využívají.

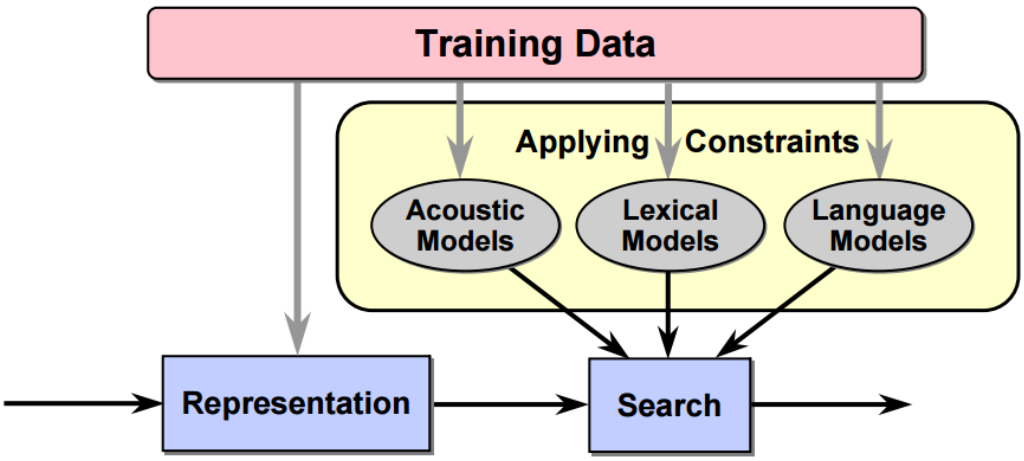
Ultimátním cílem je získat systém, který bude v reálném čase převádět přirozenou lidskou řeč na text se stoprocentní úspěšností pro všechna slova, nezávisle na zkreslení vstupních zařízení, okolním ruchu, nebo přízvuku mluvčích. Již v minulosti ASR systémy dosahovaly i kolem devadesátiprocentní úspěšnosti [3]. Nyní lze nalézt systémy s úspěšností i větší, hlavně však systémy optimalizovanější a výkonnější.

1.2 Postup rozpoznávání

Obrázek 1.1 ukazuje, že samotný návrh systému je poměrně přímočarý.

Zásadními milníky systému jsou:

- Jak zpracovat signál?
- Jak se popasovat s omezeními?
- Jak nalézt optimální výstup?



Obrázek 1.1: *Hlavní součásti ASR systému [4]*

1.3 Reprezentace a zpracování signálu

Pro reprezentaci a zpracování signálu se běžně využívají dva druhy analýzy – analýza pomocí *discrete-time* Fourierovy transformace (DTFT) a Cepstrální analýza (CA).

1.3.1 Fourierovská analýza

Discrete-time Fourierova transformace je vztah:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{+\infty} x[n]e^{-j\omega n} \tag{1.1}$$

$$x[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n}d\omega \tag{1.2}$$

jemuž pro konvergenci postačuje podmínka:

$$\sum_{n=-\infty}^{+\infty} |x[n]| < +\infty \tag{1.3}$$

Ačkoli je $x[n]$ diskrétní, $X(e^{j\omega})$ je spojitá, 2π-periodická a pro její dualitu platí:

$$y[n] = x[n] * h[n] \tag{1.4}$$

$$Y(e^{j\omega}) = X(e^{j\omega})H(e^{j\omega}) \tag{1.5}$$

a

$$y[n] = x[n]w[n] \tag{1.6}$$

$$Y(e^{j\omega}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} W(e^{j\theta})X(e^{j(\omega-\theta)})d\theta \tag{1.7}$$

V důsledku můžeme zavést *short-time* fourierovskou analýzu (STFA – Short-Time Fourier Analysis) jako:

$$X_n(e^{j\omega}) = \sum_{m=-\infty}^{+\infty} w[n-m]x[m]e^{-j\omega m} \tag{1.8}$$

a v případě, že $X(e^{j\omega})$ representuje DTFT signálu, který pokračuje i mimo okno brané v potaz (nebo je tam konstantní nula), můžeme psát, že:

$$X_n(e^{j\omega}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} W(e^{j\theta})e^{j\theta n}X(e^{j(\omega+\theta)})d\theta \tag{1.9}$$

1.3.2 Cepstrální analýza

Původně CA vznikla jako nástroj analýzy časového rozvoje seismických signálů vzniklých zemětřeseními, či explozemi. Měla pomoci nalézt především hloubku generátoru seismických signálů analýzou odezvy těchto signálů [6].

CA využívá předpokladu, že signál je výstup lineárního časově invariantního (LTI – Linear Time-Invariant) systému; tedy že je to konvoluce vstupu a impulsové odezvy. Pokud se chceme zabývat charakterizací tohoto modelu, musíme projít procesem dekonvoluce. A CA je právě postup, využívaný pro takovouto LTI dekonvoluci [5].

Vezmeme-li v potaz pozorování, že:

$$x[n] = x_1[n] * x_2[n] \iff X(z) = X_1(z)X_2(z) \tag{1.10}$$

a vyjádříme-li komplexní logaritmus $X(z)$ jako:

$$\widehat{X}(z) = \log X(z) = \log X_1(z) + \log X_2(z) \tag{1.11}$$

pak pokud je tento logaritmus jednoznačný a $\widehat{X}(z)$ je validní z-transformace, jsou nové konvolvované signály aditivní v této nové cepstrální doméně:

$$\widehat{x}(n) = \widehat{x}_1(n) + \widehat{x}_2(n) \tag{1.12}$$

Pokud se navíc omezíme na jednotkovou kružnici $z = e^{j\omega}$, pak cepstrální transformací označujeme:

$$\widehat{X}(e^{j\omega}) = \log|X(e^{j\omega})| + j \arg X(e^{j\omega}) \tag{1.13}$$

1.4 Zdroje omezení mluvené komunikace

Při předávání informace mluvenou řečí může běžně docházet ke zkreslení řečené informace díky několika nepříjemným faktorům, které mohou (a budou) ASR systému působit nepříjemnosti. Těmto omezením se nelze vyhnout, proto jsou ASR systémy nuceny brát v potaz rozličné nepřesnosti a snažit se tyto eliminovat.

Práci už tak nepříjemnou nezlehčuje ani fakt, že omezení nejsou konstantně daná; lze je pouze taxonomizovat a obcházet.

Patří sem například:

- Akustická omezení – lidský hlasový aparát trpí, jako každý jiný komplexní systém, také svými neduhami
- Fonetická omezení – „kolemjdoucí paní“ - „kolem jdou cíp a ní“
- Fonologická omezení – například splynutí „s“ a „š“ do jednoho dlouhého fonému - „lezeš shora“
- Fonotaktická omezení – kupříkladu zjednodušení konsonantických shluků u menších dětí („uličnice“ - „ulitite“), nebo nesmyslná, rádoby přejatá slova („sympathetic“ - „sympatetický“)
- Syntaktická omezení – „jedu na výlet do Českých Budějovic“ - „Českých výlet na Budějovic jedu do“
- Sémantická omezení – „zapal sirku“ - „zapal si ruku“
- Lexikální omezení – „šuplánek“ ani „niplavý“ nejsou česká slova a nemají tedy žádnou projekci v našem vědomí

a mnohá další.

1.5 Modelování systému

V ASR procesu se stroj pokouší nalézt co možná nejlepší shodu získaného signálu (reprezentace přirozené řeči) a posloupnosti písmen (slov, vět). Abychom něčeho podobného mohli dosáhnout, musíme navrhnout dostatečně komplexní a efektivní model, který nám (a našim strojům) dá do ruky nástroj pro formulaci problému, jeho porozumění a metodologii potřebnou k jeho rozlousknutí.

1.5.1 Maximum a Posteriori formulace

Tato formulace se „pokouší nalézt nejpravděpodobnější sekvenci slov W^* při daném akustickém vstupu A“ [2]. MAP přístup [1] lze typicky vyjádřit jako:

$$W^* = \arg\max_W P(W|A) \tag{1.14}$$

Věta 1. *Bayesova věta*
Pro náhodné nezávislé jevy X a Y platí:

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} \tag{1.15}$$

Za použití 1.15 můžeme upravit pravou stranu rovnosti 1.14, čímž dostaneme vztah:

$$W^* = \arg\max_W \frac{P(A|W)P(W)}{P(A)} \tag{1.16}$$

kde $P(A|W)$ je nazývána charakteristikou akustického modelu (AM – Acoustic Model) a $P(W)$ charakteristikou jazykového modelu (LM – Language Model). Jelikož $P(A)$ je konstantní vůči této maximalizaci, můžeme se jí hladce zbavit a psát:

$$W^* = \arg\max_W P(A|W)P(W) \tag{1.17}$$

1.5.2 Jazykové modely

Modelování jazyka poskytuje nástroj, jak odlišit slova a fráze, které zní podobně, či stejně (homografy, homofony, homonyma). Kupříkladu v americké angličtině jsou věty „recognize speech“ a „wreck a nice beach“ sémanticky diametrálně odlišné, foneticky jsou to však věty téměř homofonní.

Jelikož vyhledávání je v drtivé většině případů pouze jednosměrné, můžeme pravděpodobnost posloupnosti slov $W = w_1, \ldots, w_n = w_i^n$ vyjádřit pomocí řetízkového pravidla a napsat jako:

$$P(w_i^n) = \prod_{i=1}^{n} P(w_i | w_1^{i-1}) \tag{1.18}$$

Protože sekvence W může mít jednoduše příliš mnoho prvků a navíc pravděpodobnost $P(w_i)$ nemusí nutně záviset na úplně celé historii $h_i = w_1^{i-1}$, slučujeme h_i do tříd ekvivalence $\Phi(h_i)$, čímž dospíváme k:

$$P(w_i^n) \approx \prod_{i=1}^{n} P(w_i | \Phi(h_i)) \tag{1.19}$$

Jak možno nahlédnout, pro kvalitní odhad je kritické určit dobré zobrazení Φ. Čím lepší návrh tohoto zobrazení, tím lepší informaci získáme o w_i, využívaje historie $\Phi(h_i)$.

Zde nastupují na scénu *n-gram* modely [7][8], které v tomto případě zavádějí $n-1$ předchozích slov pro reprezentaci historie, tj. $\Phi(h_i) = w_{i-(n-1)}^{i-1}$, čímž získáme:

$$P(w_i^n) \approx \prod_{i=1}^{n} P(w_i | w_{i-(n-1)}^{i-1}) \tag{1.20}$$

Například uvážíme-li *bigram* (*n-gram*, $n = 2$), pravděpodobnost věty „České Budějovice jsou krásné město“, tj. posloupnosti slov:

$$W = Ceske, Budejovice, jsou, krasne, mesto$$

lze vyjádřit jako:

$$\begin{aligned} P(W) = {} & P(Ceske| <start>)P(Budejovice|Ceske) \\ & P(jsou|Budejovice)P(krasne|jsou) \\ & P(mesto|krasne)P(<konec> |mesto) \end{aligned}$$

Pravděpodobnosti $P(w_i | w_{i-(n-1)}^{i-1})$ můžeme dále odečíst z natrénovaných dat vztahem:

$$P(w_i | w_{i-(n-1)}^{i-1}) = \frac{c(w_{i-(n-1)}^{i})}{c(w_{i-(n-1)}^{i-1})} \tag{1.21}$$

kde $c(\xi_p^q)$ představuje četnost výskytů posloupnosti ξ_p^q.

Kvůli možné nepřiměřené míře řídkosti dat je nasnadě uvážit možnost, že dělitel $c(w_{i-(n-1)}^{i-1})$ bude nulový, tedy že posloupnost ještě nebyla natrénována.

Pro eliminaci těchto faktorů lze využít například některý z vyhlazovacích algoritmů (Jelinek-Mercer [9], Katz [10], Good-Turing [11]), nebo drobně pozměnit paradigma a přizvat na pomoc neuronové sítě [12].

Buď jak buď, *bigramy* stále zachovávají svou pozici v modelování jazyka — lze je jednoduše zakomponovat do Viterbiho vyhledávání [13] — a *trigramy* (*n-gram*, $n = 3$) pokračují v bytí dominantním modelem.

1.5.3 Akustické modely

Modelováním akustiky můžeme naopak získat nástroj pro zachycení a vypořádání se například s akustickým šumem, se zkreslením vstupního zařízení, či s netradičním přízvukem mluvčího.

Akustický model statisticky reprezentuje zvuky, jenž poskládány dohromady tvoří vyřčené slovo. Každé takovéto statistické reprezentaci může být přiřazena nálepka ji reprezentující (běžně foném). Aby model mohl poskytovat rozumné výsledky, je potřeba jej nejprve natrénovat na nějakém jazykovém korpusu, například pro tento účel lze využít Baum-Welchova trénovacího algoritmu [15].

Skryté Markovovy modely (HMMs – Hidden Markov Models) [14] jsou jednou z možných interpretací akustického modelu (jiné interpretace mohou zahrnovat například neuronové sítě [16], nebo *dynamic time warping* [17]).

Při použití HMMs jeden HMM reprezentuje každý foném, slova vznikají konkatenací menších HMMs, věty na oplátku konkatenací těchto HMMs a tak dále.

Proč zrovna „skrytý“ Markovův model? Tento název vychází z definice „skrytého“ Markovova procesu (HMP – Hidden Markov Process), který popíšeme za pomoci kolegů Alibaby a Rychlonožky a jejich uren.

Vezměme myšlenkový experiment, kde v místnosti za zavřenými dveřmi žije Alibaba se třemi svými urnami u_1, u_2, u_3. Každá urna obsahuje známou množinu kuliček $\{k_{u_i}\}_1^n$, kde platí $(\forall u \in \{u_1, u_2, u_3\})(\forall j \in \{1, \ldots, n\})(k_{uj} \in \{c_1, \ldots, c_n\})$, tedy každá kulička z každé urny je jedné z barev c. Alibaba bude nyní ďábelským způsobem tahat kuličky z uren, a sice tak, že l-tou kuličku K_l vytáhne náhodně a pouze s ohledem na K_{l-1}. Tomuto procesu se říká Markovův proces. Jelikož

Alibaba na běhání nikdy nebyl, dá taženou K_l Rychlonožkovi, který vyběhne ven dveřmi a složí K_l k nohám nezávislého pozorovatele, hned vedle kuličky K_{l-1} do řady.

Pozorovatel sice zná složení kuliček v urnách, ale nemá ani tušení, co se za dveřmi děje — Markovův proces je skrytý — takže může pouze smutně sledovat sekvenci kuliček, rodící se mu pod nohama. Po $1 < m < n$ tazích a Rychlonožkovo sprintech bude mít pozorovatel v zorném poli posloupnost kuliček K_1^m. Problém ale nastává, chce-li pozorovatel určit z jaké urny byla tažena K_m. Ani v případě, že $(\exists i \in \{1, 2, 3\})(K_1^m \subseteq \{k_{u_i}\}_1^m)$ si totiž nemůže být jist.

Nicméně, pozorovatel se může alespoň pokusit odhadnout pravděpodobnost, že K_m byla z urny i, což se právě HMMs pokouší zachytit.

1.5.4 Lexikální modely

Lexikální modely popisují vztah mezi jednotkami akustickými (fonémy – části slova mluveného, popsané akustickými modely) a lexikálními (lexémy – množina všech tvarů určitého slova, nebo jeho části).

Pro ilustraci lexému vezměme slova „ředkvička“ a „ředkvičky“; mají rozdílné gramatické a sémantické významy (singulár a plurál téhož), lexikální význam je ale stejný (malý, načervenalý, jedlý kořen).

Jednou z výzev navrhování ASR systému je popsat tyto vztahy pro jazyk, kde lexikální příznaky nejsou zcela zřejmé. Jak ale [18] uvádí, pro standardní stochastické HMM ASR systémy existuje lexikální model deterministický, tj. existuje bijektivní zobrazení mezi trénovanými HMMs a lexémy.

Pro jazyky obtížné, s ne zcela formálními lexikálními příznaky, lze lexikální model odvodit z některého lexikálního modelu již známého – aplikací známého modelu na nový jazyk a jeho postupnou adaptací.

Zavedeme-li $\Theta = \{\Theta_A, \Theta_L\}$ množinu parametrů akustických a jazykových modelů, kde $\Theta_A = \{\theta_a, \theta_p r, \theta_l\}$ jsou parametry akustického modelu, respektive

lexikonu, respektive lexikálního modelu, můžeme s využitím 1.14 a 1.15 rozvést:

$$W^* = \arg\max_W P(W|A,\Theta) \tag{1.22}$$

$$= \arg\max_W \frac{P(A|W,\Theta_A)P(W|\Theta_L)}{P(A|\Theta)} \tag{1.23}$$

$$= \arg\max_W P(A|W,\Theta_A)P(W|\Theta_L) \tag{1.24}$$

Hledání nejpravděpodobnější sekvence W^* pro akustický vstup A pak můžeme díky lexikálnímu modelování transformovat na problém hledání nejpravděpodobnější posloupnosti stavů Q^*:

$$Q^* = \arg\max_Q P(Q,A|\Theta) \tag{1.25}$$

$$= \arg\max_Q \prod_{t=1}^{T} p(a_t|q_t = l^i,\Theta_A)P(q_t = l^i|q_{t-1} = l^i,\Theta) \tag{1.26}$$

$$= \arg\max_Q \sum_{t=1}^{T} (\log p(a_t|q_t = l^i,\Theta_A) + \log P(q_t = l^i|q_{t-1} = l^i,\Theta)) \tag{1.27}$$

kde T je celkový počet uvažovaných oken, $Q = \{q_1,\ldots,q_t\}$ jde přes všechny posloupnosti možných HMM stavů, $q_t \in \{l^1,\ldots,l^i,\ldots,l^I\}$ a I značí počet lexémů. Obvykle je výraz $\log p(a_t|q_t = l^i,\Theta_A)$ označován jako *lokální emisní skóre* (LES – Local Emission Score) a výraz $\log P(q_t = l^i|q_{t-1} = l^i,\Theta)$ jako *přechodové skóre* (TS – Transition Score).

1.6 Hodnocení výstupů ASR

Zásadním problémem výstupu ASR systémů je, že rozpoznaná sekvence slov W^* může mít rozdílnou délku od referenční sekvence slov (buďto známé, nebo neznámé, každopádně ale té domněle správné).

Míra chybovosti slov (WER – Word Error Rate) je metrika založená na Levenshteinově (editační) vzdálenosti (LD – Levenshtein Distance) [19].

Formálně je LD mezi dvěma textovými řetězci a, b dána jako $ld_{a,b}(|a|, |b|)$:

$$ld_{a,b}(0,0) = 0 \tag{1.28}$$

$$ld_{a,b}(i,0) = i \tag{1.29}$$

$$ld_{a,b}(0,j) = j \tag{1.30}$$

$$ld_{a,b}(i,j) = \min \begin{cases} ld_{a,b}(i-1,j-1) + 0, & \text{pro } a_i = b_j \\ ld_{a,b}(i-1,j-1) + 1 & \text{(nahrazení)} \\ ld_{a,b}(i,j-1) + 1 & \text{(vložení)} \\ ld_{a,b}(i-1,j) + 1 & \text{(vypuštění)} \end{cases} \tag{1.31}$$

kde $1 \leq i \leq |a|, 1 \leq j \leq |b|$.

WER je pak odvozena z LD, pracujíc na úrovni slov místo na úrovni menších jednotek:

$$WER = \frac{S + D + I}{N} \tag{1.32}$$

kde

- S je počet nahrazení slova slovem
- D je počet vypuštěných slov
- I je počet vložení nových slov
- N je počet slov v referenční sekvenci

2. E-learning jako přenos informace

V této kapitole se čtenář seznámí s několika základními definicemi pojmu e-learning, dozví se, jak se běžně na tento jev nahlíží. Seznámí se s některými pro a proti nasazení e-learningu jakožto výukového média a média pro získávání a předávání informace.

Dále bude čtenáři poskytnut vhled do současného trendu využití e-learningu v herním průmyslu a naopak, využití herního průmyslu v procesu e-learningu.

Čtenář se také dozví o vzniku fenoménu GWAP (Game With a Purpose), jeho praktickém užití a některých příkladech zapracování do života běžného uživatele webu.

Nakonec bude zmíněn vztah GWAP modelu a hry `trogASR`.

2.1 Co je to e-learning

Definovat e-learning (EL) se může zdát úkolem poněkud obtížným, jelikož žádná jedna konkrétní definice nepostihne všechny oblasti a aspekty EL. Proto je zde uvedeno několik definic a ponecháno na čtenáři, nechť si vybere tu pro něj nejpříhodnější.

- „EL je výuka s využitím výpočetní techniky a internetu“ [20].
- „EL je způsob provozu výuky, tréninku, či edukačního programu pomocí elektronických prostředků. EL zahrnuje využití počítače, nebo jiného elektronického zařízení (například mobilního telefonu), jako média pro poskytnutí tréninkového, edukativního, či učebního materiálu“ [21].
- „EL je takový typ učení, při němž získávání a používání znalostí je distribuováno a usnadňováno elektronickými zařízeními“ [22].

2.1.1 Proč používat e-learning

Jak [23] podotýká, tradiční studenti jsou při procesu EL více spontánní a otevření. Většinou si jsou schopni nalézt vlastní postup faktorování výuky, který je pro ně nejméně násilný a může být tedy výrazně efektivnější, než-li způsoby výuky v kamenné třídě s figurou absolutního supervizora – učitele.

EL přináší další výhody oproti klasickým výukovým metodám. Díky dostupnosti virtuálních médií mohou žáci sedět na druhém konci světa a studovat kurzy expertů, se kterými by se jen těžce osobně setkávali.

EL skrz webové rozhraní přináší o to lepší možnost rozšiřitelnosti; web jako takový využívá v dnešní době naprostá většina civilizovaného světa. Má tedy smysl využívat právě toto médium jako to dominantní.

EL je dostupný v dnešní době každému a kdykoli; odpadá závislost na fyzickém kontaktu dvou osob pro průběh procesu předání informace. Tento fakt značně hraje do noty celkovému dnešnímu trendu, jenž míří směrem globalizace a postupného odpadávání fyzické komunikace jako takové.

EL může být masový. Běžný vyučující je schopen reálně zvládnout třídu třiceti žáků. S dobře navrženým EL systémem mohou najednou pracovat miliony lidí. A to při vytížení toho samého jednoho učitele, který daný EL kurz navrhl a realizoval.

2.1.2 Proč e-learning nepoužívat

EL jako koncept má mnoho výhod, avšak je pln i inherentních nevýhod, plynoucích povětšinou ze způsobů nasazení a podání.

Jakkoli velký může mít EL přínos pro předání informace, může mít dopad na kvalitu informace samotné. Vzděláváme-li se prostřednictvím médií a autorů, jejichž identita je skryta, informace, získaná tímto způsobem, nemusí být nutně validní. Nemusíme mít kontrolu nad zdrojem informace, nemáme páky, jak informaci verifikovat. Přesto bývá informaci, předávané virtuálně, přikládán jaksi absolutní význam.

Dalším dopadem je separace vyučujícího a vyučovaného. Nemůžeme si dostatečně jasně vymezit názor na zdroj, který nám EL program připravil. Tento údaj se může zdát nepotřebný. Je však nasnadě si uvědomit, že ač věříme svému úsud-

ku bezmezně, šikovnou manipulací s údaji o původu zdroje lze naše podvědomí naprogramovat na zvýšenou / sníženou hladinu zájmu o poskytovaný materiál.

EL může zaviňovat i úpadek socializační. Studenti nejsou nuceni opouštět zázemí své domény a interagovat se svými vrstevníky, spolustudenty a lektory fyzicky. Stačí jim otevřít rozhraní svého konkrétního média a komunikovat jeho prostřednictvím.

2.2 Výuka a počítačové hry

„Počítačové hry mi zničily život.
Ještě, že mám další dva.“

— autor neznámý

Téma výuky prostřednictvím počítačových her je dnes a denně omílanou parafrází.

Na jednu stranu, plejáda edukativních her a hříček nabízí pohled na interakci výuky a herního průmyslu jako na proces oboustranně výhodný. Student — hráč — je vtažen do interaktivního prostředí počítačové hry, kde může objevovat a osahat si koncepty, které v reálném světě mohou být náročné na vysvětlení, nebo v něm dokonce ani nemusí existovat [24]. Herní vývojář naopak získává platícího zákazníka.

Na druhou stranu, student — hráč — může být vystaven vlivům nežádoucím; pro svůj věk, svou psychickou vyspělost, či pro svou víru a formování osobnosti. Násilí, nepřiměřená erotika, nespisovný a vulgární slovník mohou zapříčinit zkreslení náhledu na svět, obzvlášť u jedinců nedospělých, s ne zcela zformovanou osobností.

Jak ale [25] uvádí, negativní sociální dopady na hráče nezapříčiňuje často hra samotná, nýbrž prostředí, ve kterém se hráč při hraní pohybuje. Negativistické rodinné zázemí, domácí násilí, nepohoda, toto jsou stresory, které, dokresleny vulgární počítačovou hrou, mohou propuknout až v agresivní chování hráče, jeho depresivní sklony, popřípadě v inklinaci k násilí. Jak [25] také uvádí, i násilná počítačová hra může mít pozitivní vliv na zdravého jedince a poskytnout mu nástroj k formování základních kognitivních procesů, rychlého uvažování a přístupu

k řešení problémů.

S čím se často setkáváme u hráčů, hrajících počítačové hry v jazyce jiném, než je mateřský, bývá až nečekaný nárůst jazykových schopností. Tento proces u zarputilých hráčů může vést až k částečnému, nebo i úplnému, bilingvismu. Hráč má totiž zájem sám sebe vzdělávat v jazyce své oblíbené hry. Pokud nějaké frázi v daném jazyce nerozumí, rád a dobrovolně si ji vyhledá ve slovníku, aby byl schopen ve hře pokračovat.

2.3 E-learning hrou a crowdsourcing

Crowdsourcing (CS), též někdy *wisdom of the crowds* (moudrost davů), je označení pro způsob dělby práce, kde se pro řešení problému místo zaměstnance, či kontraktora, využije blíže nespecifikovaná skupina lidí (*crowd* – dav) v rámci všeobecné výzvy.

CS se často využívá například při vývoji nové technologie (testování), zdokonalování algoritmu (komparace výsledků), či pro zachycení, analýzu a třídění velkého množství dat.

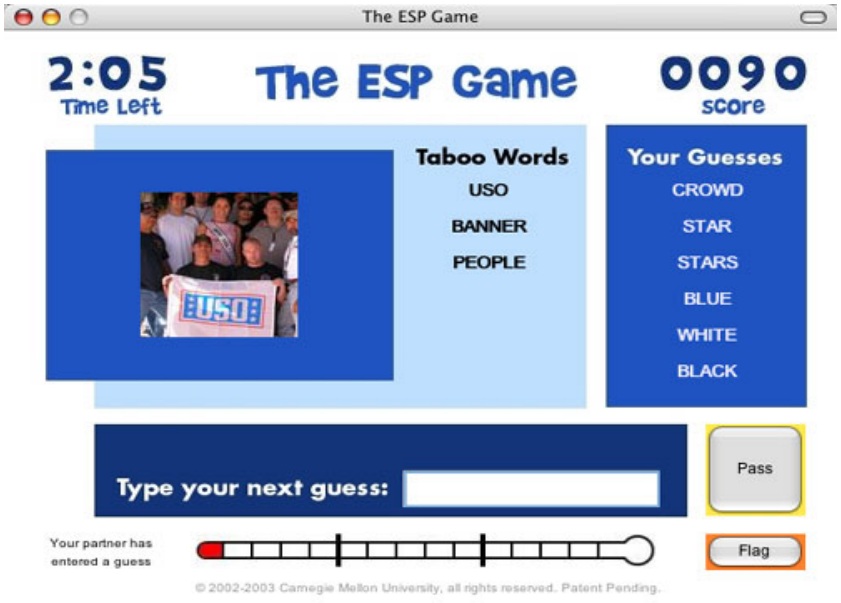
„Každý rok, lidé na celém světě stráví miliardy hodin hraním počítačových her. Co kdyby mohl tento vynaložený čas a energie být soustředěny do užitečné práce? Co kdyby lidé, hrající počítačové hry, mohli bezděky zároveň řešit rozsáhlé a náročné problémy?“ [26]

Díky této otázce vzniká pojem *hra s účelem* (GWAP – Game With a Purpose). Jedná se, jak název napovídá, o počítačovou hru, která mimo pobavení hráče zároveň plní jiný, taktéž smysluplný účel. Lze zde vidět krásné použití CS a EL v praxi. Uživatelé se nevědomky stávají součásti procesu CS a ještě se při tom baví a vzdělávají.

2.3.1 ESP game

Pojem GWAP se poprvé začal objevovat v souvislosti s `ESP game` [27]. `ESP game` (později GIL – Google Image Labeler) je hra vyvinutá pro popasování se s problémem vytváření komplexních metadat obrázků.

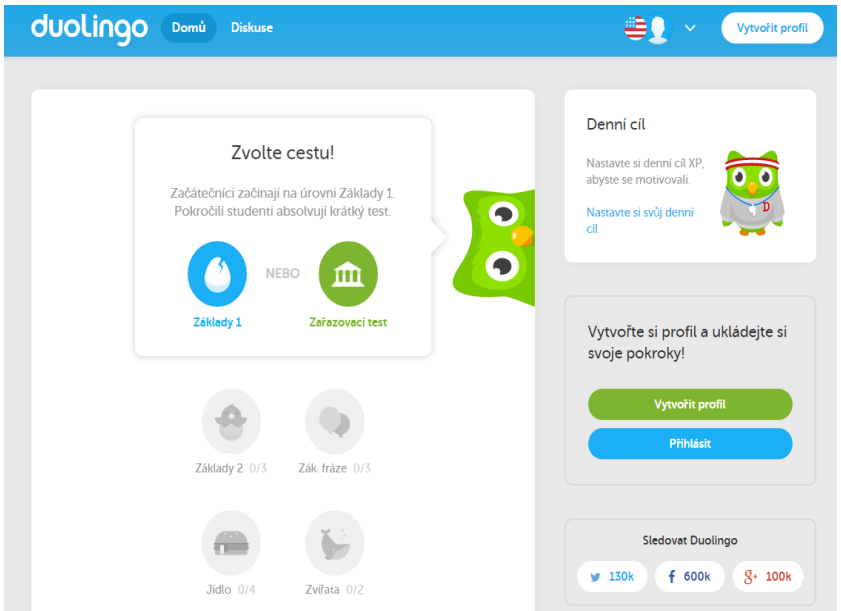
Původně měla `ESP game` řešit úkol, který tou dobou stroje nezvládaly, to jest



Obrázek 2.1: *Ukázka ESP game*

rozpoznávání obrázků. Díky tomu, že autoři zabalili tento úkol do jednoduché hry, mohli využít ohromného potenciálu zahálejících uživatelů webu.

2.3.2 Duolingo



Obrázek 2.2: *Ukázka Duolingo*

`Duolingo`[1] je populární webová hra pro výuku jazyka. Umožňuje například

[1] `https://www.duolingo.com/`

soutěžit s přáteli v jazykové zdatnosti prostřednictvím sítě Facebook.

Nalezneme zde desítky jazykových kurzů, v současnosti i kurzy angličtiny pro české mluvčí. V rámci výuky jsou zde překládány texty partnerských společností, které za překlady platí. Studenti pak provádí samotný překlad a data posléze i verifikují.

2.3.3 reCAPTCHA

Obrázek 2.3: *Ukázka reCAPTCHA*

`reCAPTCHA`[2] [28] není tak úplně hrou. Jedná se o dialog mezi uživatelem a počítačem, který má zjistit, zda uživatel není také stroj. Zároveň jsou tímto procesem získávána data o prezentovaných obrázcích a jejich obsahu.

Díky `reCAPTCHA` procesu se podařilo například digitalizovat celý archiv *New York Times*. Zároveň je takto podporován vývoj softwaru pro strojové rozpoznávání textu, který je na oplátku používán pro prolomení `reCAPTCHA` kontroly.

2.4 TrogASR jako GWAP

`TrogASR` aplikace, kterou tato práce reprezentuje, je bezesporu typickou GWAP. Jedná se o webovou hru s účelem jazykové výuky za pomoci soupeření o místa na žebříčku, čímž se nejvíce podobá hře `Duolingo` ze zmíněných.

Na jazykovou výuku jde ale z jiného úhlu. Na rozdíl od `Duolingo` nevyžaduje psaný uživatelský vstup, nýbrž vstup hlasový. Cílem tohoto návrhového kroku je poskytnout uživateli větší pocit „zatažení“ do hry a inteligentnější interakce.

Stran smyslu hry `trogASR` lze uvést stránku EL — uživatel projevuje a zdokonaluje své jazykové znalosti — a stránku CS — uživatel bezděčně poskytuje verifikovaná data `CloudASR` knihovně.

[2] `https://www.google.com/recaptcha/intro/index.html`

3. Použité technologie

V této kapitole se čtenář seznámí s technologiemi, frameworky a prostředími, využitými pro vývoj aplikace `trogASR`. U každého z nich se navíc ve stručnosti dozví o díle zapojení daného nástroje do aplikace.

Všechny nástroje jsou buď *open-source*, *freeware*, nebo je pro ně volně dostupná akademická licence.

3.1 Python

`Python`[1] je *open source* vysokoúrovňový skriptovací programovací jazyk. Nabízí širokou škálu programátorských paradigmat, jako například OOP (objektově orientované programování), imperativní, procedurální, nebo funkcionální přístup k programování.

Tento jazyk v osmdesátých letech dvacátého století navrhl Guido van Rossum. V roce 1989 započal vývoj. Od té doby popularita `Python`u vzrostla natolik, že se stal základní součástí většiny distribucí systému Linux. Lze jej však velmi snadno instalovat i do jiných operačních systémů.

Hlavními návrhovými milníky a posléze i výhodami `Python`u jsou jeho orientace na čistotu a efektivitu psaní kódu, na velkou modularitu a na výraznou uživatelskou přívětivost. Některými dokonce `Python` bývá považován za nejvhodnější programovací jazyk pro začátečníky.

V aplikaci `trogASR` bylo `Python`u využito pro tvorbu *back-end*u, tj. pro tvorbu prostředí aplikačního serveru za pomoci MVC (Model View Controller) návrhového vzoru a pro operování s databází.

[1] `https://www.python.org/`

3.2 Flask

Flask[2] je *microframework* napsaný v Pythonu, založený na knihovně Werkzeug[3] a na templatovacím systému Jinja2[4].

Označení *microframework* nese Flask proto, že se jedná vskutku o framework minimalistický. Základní distribuce obsahuje pouze nástroje nezbytně nutné pro běh webové aplikace. Velký důraz je však kladen na modularitu tohoto frameworku; uživatel má tedy možnost doinstalovat si do svého prostředí právě ta rozšíření, která potřebuje.

Obrázek 3.1 ukazuje, že instalace Flasku je velice snadná. Stačí mít funkční prostředí Pythonu a o zbytek se již může postarat nástroj easy_install a pip. Pro zprovoznění základní webové aplikace poté stačí už jen sedm řádek kódu.

```
> <python_home>/Scripts/easy_install pip
> pip install Flask

----------

#hello.py
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'

if __name__ == '__main__':
    app.run()

----------

> python hello.py
  * Running on http://127.0.0.1:5000/
```

Obrázek 3.1: *Příklad instalace frameworku Flask na MS Windows, kódu modulu "hello" a spuštění webového serveru.*

[2] http://flask.pocoo.org/
[3] http://werkzeug.pocoo.org/
[4] http://jinja.pocoo.org/

Frameworku `Flask` bylo v aplikaci `trogASR` využito především proto, že tento framework je velmi jednoduchý na použití; naprosto oproštěn od přebytečných součástí, které by bylo třeba konfigurovat, ačkoli by je aplikace nevyužívala. Zároveň `Flask` oplývá širokou komunitou uživatelů, schopných poradit s problémem, a exhaustivní dokumentací.

3.2.1 Flask-Assets

```
> pip install Flask-Assets

----------

#hello.py
from flask.ext.assets import Environment, Bundle

# ... Flask app initialization and setup

assets = Environment(app)
bundles = {
    'all-css': Bundle(
        'css/style.css', 'css/font.css',
        output='gen/all.css', filters='cssmin'),
    'all-js': Bundle(
        'js/app.js', 'js/i18n.js',
        output='gen/all.js', filters='jsmin')
}
assets.register(bundles)

# Flask app run ...

----------

<!-- index.html -->
{% assets 'all-css' %}
    <link rel="stylesheet" href="{{ ASSET_URL }}"/>
{% endassets %}
{% assets 'all-js' %}
    <script type="text/javascript" src="{{ ASSET_URL }}"></script>
{% endassets %}
```

Obrázek 3.2: *Příklad instalace balíčku Flask-Assets na MS Windows, zapojení do modulu "hello" a ukázka použití v Jinja2 HTML templatu.*

`Flask-Assets`[5] je rozšiřovací balíček pro `Flask` framework, který řeší správu a minifikaci statických skriptů a stylů.

Jak ukazuje obrázek 3.2, instalace je opět triviální, používání balíčku pak velmi intuitivní a praktické.

Rozšíření `Flask-Assets` lze také jednoduše konfigurovat. Například spouštíme-li aplikaci v *debug* módu, můžeme knihovně říci, aby poskytovala nesloučené a neminifikované zdroje.

Tato knihovna je v aplikaci `trogASR` využívána hlavně z důvodu pohodlnosti správy statických zdrojů.

3.2.2 SQLite3

Pro správu databáze byl zvolen *light-weight* framework `SQLite3`. Tento přichází již zahrnut jako součást instalace frameworku `Flask`.

Aplikace `trogASR` využívá tento framework kvůli jeho jednoduchosti. V budoucnosti je jednoduše možné jej nahradit některým z pokročilejších frameworků.

3.3 JavaScript

`JavaScript` (JS) je multiplatformní, objektově orientovaný, interpretovaný skriptovací jazyk. Jeho autorem je Brendan Eich ze společnosti Netscape.

JS sice svou syntaxí patří do rodiny jazyků `C/C++/Java`, s posledním zmíněným jazykem však krom slova `Java` v názvu a podobné syntaxe nemá nic společného. Volba názvu má čistě marketingové důvody – těžila právě z oblíbenosti jazyka `Java`. Standardizovaným jménem JS je podle ISO (International Organization for Standardization) `ECMAScript`.

JS je dnes již tak rozšířen, že jeho interpretaci podporuje drtivá většina současných webových prohlížečů. Nejčastějším využitím JS je totiž manipulace s WWW stránkou a ovládání všemožných prvků GUI (Graphic User Interface).

Protože aplikace `trogASR` je velmi dynamická a odesílat data při každé uživatelské akci na server by způsobovalo uživateli nemalou frustraci odezvou, je

[5] `http://flask-assets.readthedocs.org/`

JS zhusta využíván. Pomocí JS je řešena naprostá většina animací, přechodů a dynamických prvků aplikace.

3.3.1 jQuery

`jQuery`[6] je malý, rychlý a možnostmi bohatý JS framework. Umožňuje uživateli zefektivnit proces psaní kódu, zjednodušit manipulaci s DOM (Document Object Model) a obalit a zpříjemnit využití technologie AJAX (Asynchronous JavaScript and XML).

Pro jeho jednoduchost, flexibilitu a ohromující uživatelskou komunitu byl `jQuery` framework zvolen jako základní kámen celé aplikace `trogASR`.

3.3.2 transform.js

`transform.js`[7] je minimalistický `jQuery` plugin, umožňující animaci CSS (Cascading Style Sheets – kaskádové styly) vlastností *transform*.

Jednoduchým zapojením a použitím získal tento plugin své místo v aplikaci `trogASR`.

3.4 HTML5

HTML (HyperText Markup Language – hypertextový značkovací jazyk) verze 5 je nejnovější iterací HTML jazyka. Přináší oproti svému předchůdci, verzi 4.01, nové sémantické prvky, vyřazuje zastaralé konstrukty z jazyka a opravuje řadu chyb.

HTML5 jazyka je v aplikaci `trogASR` využito pouze poskrovnu; je nasazen pouze jako rámec aplikace, do kterého se většina obsahu generuje dynamicky s použitím JS.

Zároveň některých nových možností HTML5 využívá knihovna `CloudASR` (jmenovitě API pro záznam zvuku).

[6] `https://jquery.com/`

[7] `http://louisremi.github.io/jquery.transform.js/`

3.5 CSS3

CSS technologie umožňuje programátorům a návrhářům oddělit strukturu a obsah HTML stránky od jejího vzhledu. CSS jazyk popisuje vztahy mezi elementy ve stránce a způsoby zobrazení. Díky tomuto popisu je možno vytvořit dobře znovupoužitelné nastavení, které je transparentně přenositelné mezi prvky nejen v jedné stránce, ale i mezi prohlížeči a platformami. Autorem konceptu CSS byl Håkon Wium Lie a standardizační organizace `W3C`[8].

Kaskádové styly verze 3 přináší na trh přehršel nových, užitečných konceptů. Mezi tyto patří například příchod *@media* pravidla pro zachycení a podmínění stylů podle využitého zobrazovacího zařízení, nebo zavedení *transform* a *transition* vlastností elementů pro usnadnění pozicování a animace.

Aplikace `trogASR` využívá CSS3 velmi extenzivně. CSS pravidla definují vzhled všech prvků, které uživatel vidí, a dokonce i některých animovaných přechodů.

3.6 CloudASR

`CloudASR`[9] je softwarová platforma a veřejná ASR služba. Za jejím vznikem stojí kolega Ondřej Klejch, který tuto platformu vyvinul v rámci své diplomové práce na MFF.

Hlavními rysy `CloudASR` platformy jsou především

- automatické škálování podle zatížení
- jednoduchost nasazení a provozu
- cloudová nátura
- *state-of-the-art* inkrementální rozpoznávání řeči
- kompatibilita s *Google Speech API*
- rozšiřitelnost o nové ASR enginy

[8] `www.w3.org/`

[9] `http://www.cloudasr.com/`

CloudASR také poskytuje JS knihovnu, jejíž zapojení si aplikace trogASR z titulu práce žádala. Této knihovny je využito právě pro transformaci hlasového vstupu na výstup textový. Její zavedení demonstruje obrázek 3.3.

Obrázek 3.4 pak ukazuje, jak vypadají JSON (JavaScript Object Notation) objekty, které JS knihovna dodává na konci samotného rozpoznávání.

```
<!-- index.html -->
<script src="http://www.cloudasr.com/static/js/socket.io.js"/>
<script src="http://www.cloudasr.com/static/js/Recorder.js"/>
<script
    src="http://www.cloudasr.com/static/js/SpeechRecognition.js"/>

----------

// game.js
var speechRecognition = new SpeechRecognition(
    'http://api.cloudasr.com:80'
);
speechRecognition.onResult = function(res) {/* process result */};
speechRecognition.onStart = function() {/* handle start event */};
speechRecognition.onEnd = function() {/* handle end event */};
speechRecognition.onError = function(err) {/* handle error */};

$('#recordButton').on('click', function(){
    if(speechRecognition.isRecording){
        speechRecognition.stop();
    }
    else{
       speechRecognition.start('en-wiki');
    }
});
```

Obrázek 3.3: *Ukázka zapojení CloudASR JS knihovny.*

3.7 PyCharm

PyCharm[10] je vývojové prostředí, vyvíjené a udržované společností JetBrains. Je určeno převážně pro vývoj aplikací v jazyce Python, oplývá však i velmi sofistikovaným prostředím pro vývoj v JS.

[10] https://www.jetbrains.com/pycharm/

```
{
    'result': [{
        'alternative': [{
          'confidence': 0.5549500584602356,
          'transcript': 'WHERE AM I NOW'
        }, {
          'confidence': 0.14846260845661163,
          'transcript': 'WHERE AM I NOWADAYS'
        }],
        'final': true
    }],
    'result_index': 0
}
```

Obrázek 3.4: *Příklad finálního výsledku rozpoznávání.*

JS podpora sahá až na úroveň nativního zapojení všemožných frameworků, jako jsou například právě `jQuery`, ale i `ExtJS`, nebo `Node.js`.

Obecně je práce v tomto vývojovém nástroji velmi příjemná. Kontextová inteligentní nápověda, možnosti refactoringu a projektově-přehledové nástroje hledají na dnešním trhu kompetici jen těžce.

3.8 Několik slov o webových prohlížečích

V současné době je trh s webovými prohlížeči poměrně rozmanitý. Běžný uživatel má na výběr nepřeberné množství různých řešení, více i méně rozšířených a použitelných.

Bohužel, ani po několika letech existence norem HTML5 a CSS3 nejsou tyto v naprosté většině zmíněných prohlížečů podporovány, nebo jsou podporovány jen částečně, nebo chybně.

Proto aplikace `trogASR` necílí na multiplatformnost. Bude fungovat pouze v prohlížečích, které dostatečně podporují čisté HTML5 a CSS3. Těmito jsou k dnešnímu dni pouze prohlížeče `Google Chrome` (verze 41) a `Opera Next` (verze 29).

4. Implementace

V této kapitole se čtenář dočte o implementaci aplikace `trogASR`. Bude obeznámen s pilíři aplikace, nahlédne na vývoj aplikace jako na řešení komplexní, i po částech.

Nejprve se čtenář dočte o struktuře aplikace — o modularitě, designu, znovupoužití kódu — při čemž se seznámí s myšlenkovými pochody, stojícími za návrhem implementace.

Dále se čtenář dozví o interoperabilitě s Facebook API (Application Programming Interface – aplikační programátorské rozhraní) a o důležité součásti — internacionalizaci aplikace.

V závěru je pak věnován prostor úhrnu některých použitých návrhových vzorů, principu jejich fungování a projekci do aplikace.

4.1 Modulární kostra

Aplikace `trogASR`, jakkoli neobsahuje žádné extrémně složité algoritmické řešení, byla navrhována a vyvíjena s ohledem na rozšiřitelnost a modularitu.

```
// app.js
var APP = APP || {};
(function($, window, app) {
    var registerComponent = function(cmp) {
        // Add the cmp to the process chain
    };
    var reset = function() {
        // Call reset() on every registered component
    };
    var initialize = function() {
        // Call initComponent() on every registered component
    };
    $(window).load(initialize);
    app.registerComponent = registerComponent;
    app.reset = reset;
})(jQuery, window, APP);
```

Obrázek 4.1: *Ukázka části kódu hlavní komponenty.*

```
// i18n.js
(function($, ns) {
    var i18n = {
        initComponent: function() {/* prepare stuff */},
        reset: function() {/* reset stuff */}
    };
    ns.I18N = ns.registerComponent(i18n);
})(jQuery, APP);
```

Obrázek 4.2: *Ukázka části internacionalizační komponenty.*

Každá komponenta systému musí být registrována v komponentě hlavní, aplikační, a musí splňovat základní implementační požadavky, aby vůbec systémovou komponentou být mohla.

Jelikož v JS neexistuje princip *interface*, muselo být se smluvním typováním nakládáno jinak. Zvoleným přístupem je návrhový vzor *duck-typing* (viz sekce 4.7.1).

Prvním krokem je běh JS skriptu, kdy prohlížeč skript interpretuje, jak je vložen do stránky. V tuto chvíli se na prvním místě v pořadí nachází skript inicializující hlavní aplikační komponentu. V této se pak nachází mechanismus pro registraci ostatních komponent.

Každá registrovaná komponenta musí splňovat několik základních vlastností – musí zpřístupňovat metodu `initComponent()` a metodu `reset()`. První zmíněná metoda je volána na každé komponentě v handleru `window.onload`, zatímco druhá zmíněná metoda je volána při nutnosti znovuvytvoření statických prvků komponenty (například při změně lokále, či při změně velikosti okna). Část implementace aplikační komponenty demonstruje obrázek 4.1. Část implementace běžné komponenty pak zobrazuje obrázek 4.2.

4.2 Templatovací systém

Pro pohodlnost používání a zvětšení znovupoužitelnosti kódu byl do aplikace `trogASR` implementován systém na vytváření a processing HTML templatů. Tohoto systému celá aplikace využívá ve chvíli, potřebuje-li inicializovat jakoukoli komponentu vícekrát (položky menu, tlačítka, ...).

Každý takovýto template má formu HTML kódu, mixovaného s možným JS kódem; vymezeným konstruktem `<% <JS_code> %>`, nebo `<%= <JS_variable> %>`.

```
<!-- facebookWidget.html -->
<script type="text/html" id="tpl_fbWidget">
    <div id="<%=id%>" class="<%=widgetType%>"
    <% for(var name in data) { %>
        data-<%=name%>="<%=data[name]%>"
    <% } %> >
    </div>
</script>

----------

// facebook.js
var htmlBuffer = APP.TEMPLATES.process('tpl_fbWidget', {
        widgetType: widgets.LIKE.type,
        data: widgets.LIKE.data
    }
);
htmlBuffer += APP.TEMPLATES.process('tpl_fbWidget', {
        widgetType: widgets.LOGIN.type,
        data: widgets.LOGIN.data
    }
);
$('#fbHolder').append(htmlBuffer);
```

Obrázek 4.3: *Ukázka "facebook widget" templatu a jeho použití.*

Všechny tyto templaty jsou uzavřeny v `<script type="text/html"></script>` elementech, kterým interpretující prohlížeč nepřikládá žádnou zvláštní váhu, pouze je vkládá do stránky. Příklad demonstruje obrázek 4.3.

4.3 Sprite factory a obrázkový font

Pro účely zvýšení grafické fidelity byla navržena oku lahodící barevná paleta a ji využívající obrázkový font. Tento font je používán pro všechny texty, kde to má smysl, a slova v tomto fontu jsou generována dynamicky pomocí komponenty *sprite factory*.

Sprite factory komponenta se stará o vytváření dynamických grafických prvků podle současného nastavení jazyka a velikosti zobrazovacího média. Pro tento proces využívá taktéž templatovací systém (sekce 4.2).

Obrázek 4.4: *Položka menu "Play"/"Hrát" pro anglické, respektive české jazykové nastavení.*

Slova obrázkového fontu jsou tvořena ze *spritů* jednotlivých písmen, uzavřených do obalovacího `<div>` elementu. Velikost písmen a slov je dána responzivním nastavením aplikace (více v sekci 4.4). Každé písmeno je reprezentováno vlastním obrázkem, pro nějž existuje CSS styl, který jej zapojí jako pozadí příslušného elementu. Vzhled takto vygenerovaného dynamického slova ukazuje obrázek 4.4.

4.4 Responzivní design

Protože aplikace `trogASR` se má líbit a být uživatelsky oblíbená, vyvstala nutnost možnosti aplikaci zobrazovat na co možná nejširším spektru zobrazovacích médií. Ačkoli aplikace podporuje jenom specifickou podsekci webových prohlížečů, podporované prohlížeče lze nalézt i na jiných platformách, než jen PC (PC – Personal Computer) — například mobilní zařízení s operačním systémem Android. Zároveň je třeba vzít v potaz i fakt, že zobrazovací média uživatelů běžných PC se liší, někdy i výrazně.

Proto je aplikace `trogASR` navržena čistě dynamicky. Veškeré stylování je činěno pomocí procentuálních poměrů velikostí elementů vůči jejich nadřazeným elementům.

V místech, kde je procentuální vyjádření nemožné (šířky okrajů, velikosti *box-shadow* vlastností), jsou nasazeny jednotky `vw` a `vh`. Tyto jednotky vyjadřují jedno procento šířky zobrazované plochy (`1vw`), respektive jedno procento její výšky (`1vh`).

```
<!-- index.html -->
<div id="letter-metrics"></div>

----------

/* font.css */
#letter-metrics { width: 3%; display: none; }

----------

// spriteFactory.js
var letterSize;
var initComponent = function() {
    letterSize = $(#letter-metrics).width();
};
```

Obrázek 4.5: *Ukázka propagace dynamické velikosti písmen obrázkového fontu.*

Dynamická velikost písmen obrázkového fontu je zařízena pomocí skrytého `<div>` elementu, který je nastylován na tři procenta šířky zobrazovacího média. V inicializaci komponenty *sprite factory* je posléze tato skutečnost zaznamenána a aplikována na jednotlivá písmena jako jejich výška a šířka (písmena jsou vždy čtvercová). Rozměr písmen a slov je dále možno upravovat posláním multiplikačního parametru velikosti do příslušných metod. Tento proces demonstruje obrázek 4.5.

Obecně je doporučený minimální poměr stran 1:1, maximální pak 2:1. Optimální poměry jsou 16:9, nebo 4:3. Doporučené minimální rozlišení je *720p HD* (High Definition), optimální *1080p FHD* (Full HD), maximální *4k UHD* (Ultra HD).

4.5 Facebook API

Hra `trogASR` by také měla být kompetitivní. Proto byla zvolena integrace s facebookovým API. Tato zajišťuje, že chce-li uživatel soupeřit s ostatními, má jednoduchou možnost, jak se do aplikace přihlásit.

V případě, že uživatel přihlašovací možnosti nevyužije, má i tak možnost hru hrát, nemůže se však umístit na žebříčcích. Aplikace jako taková na uživatelovu „zeď“ žádné příspěvky neodesílá — uživatel může tuto volbu učinit sám pomocí

„Like“, nebo „Share“ tlačítek, které se mu taktéž implicitně nabízí. Není proto důvod, aby se uživatel do hry nepřihlásil. Navíc, na facebooková tlačítka jsou lidé poměrně zvyklí a nebojí se jich použít.

Pro zapojení facebookových tlačítek do webové stránky nabízí Facebook poměrně jednoduché a přímočaré řešení[1]. Stačí do stránky umístit `<div>` element, přiřadit mu příslušnou CSS třídu, zadefinovat příslušné *data* atributy a natáhnout do stránky skript, který všechny tyto elementy přepíše elementy *iframe*. Do těchto pak načte svá data a zobrazí je. Příkladem nastavení facebookových *widgetů* budiž obrázek 4.3.

Velkou nevýhodou facebookových *widgetů* je ale poměrně malá konfigurovatelnost. Obzvlášť co se týče nastavení velikostí tlačítek, neposkytuje API — krom předdefinovaných *small*, *medium*, *large* rozměrů — žádné rozšiřující možnosti. Což v důsledku působí problém s ohledem na responzivní design aplikace. Je-li zobrazovací médium moc malé, tlačítka začnou „vytékat“ z přiřazených kontejnerů. Je-li naopak moc velké, tlačítka jsou titěrně malá.

4.6 Internacionalizace

Anglická zkratka I18N (Internationalization) je běžně používána pro označení částí systému, které jsou schopny přizpůsobit své chování (nebo toto přizpůsobení zajistit) dle volby uživatelského lokále. Aplikace `trogASR` tento koncept přejímá a integruje.

Každý zobrazovaný text, který má být internacionalizován, musí mít svůj lokalizační klíč. Pro každý klíč pak musí existovat překlady do všech podporovaných jazyků. Na překlady textů je možno přistupovat buďto skrze statickou proměnnou `t` modulu I18N, kam se příslušné texty uloží vždy při změně lokále, nebo přes dynamickou `getText([locale])` metodu, která na vstupu požaduje lokále, pro nějž se má text získat. V případě vynechání tohoto parametru metoda na klíči vrátí text pro současně nastavené lokále. Takže například text pro tlačítko *play* lze získat buďto ze statického kontextu jako `APP.I18N.t.MENU_PLAY`, nebo dynamicky jako `APP.I18N.keys.MENU_PLAY.getText([locale])`.

[1] `https://developers.facebook.com/docs/javascript`

V současné době jsou v základu podporovány dva zobrazovací jazyky — angličtina a čeština. Nastavení jazyka přepne kompletně celou aplikaci a vynutí tedy její reset.

Zároveň je možno poměrně jednoduše v několika málo krocích přidat další zobrazovací jazyk. Je pro to potřeba:

- zavést nové lokále do souboru `i18n.js` do příslušného výčtu s příslušnými zkratkami (*en*, *cs*, *klingon*, ...)
- přidat překlad pro každý lokalizační klíč v témže souboru
- poskytnout styly pro tlačítko na výběr nového lokále (viz obrázek 4.6)
- pokud nový jazyk má ve své abecedě nějaké speciální znaky, je nutno poskytnout i tyto (viz obrázek 4.7)

O zbytek se postará jádro modulu I18N.

```
/* style.css */
#localeSelector .localeSelectorItem.klingon {
    background: url('img/ico/klingon.png') no-repeat right top;
    background-size: contain;
}
```

Obrázek 4.6: *Příklad přidání stylu pro ikonu výběru jazyka „klingon“.*

```
// spriteFactory.js
var letterMap = { '❤': 'H1' /*, ...*/ };

---------

/* font.css */
.letter-H1 {
    background: url('img/font/klingon/H1.png') no-repeat right top;
}
```

Obrázek 4.7: *Příklad přidání specifického znaku do abecedy nového jazyka a nastavení nového prvku obrázkového fontu.*

4.6.1 Zdrojové a cílové jazyky překladu

Jelikož se hra `trogASR` zaměřuje na překlad slov a frází z jazyka do jazyka, lze pochopitelně nastavit i dvojice zdrojový a cílový jazyk. Zdrojový jazyk je ten, ve kterém se uživateli budou slova a fráze zobrazovat, cílový naopak ten, ve kterém uživatel slova a fráze musí vyřknout.

Platforma `CloudASR` v současnosti rozumí pouze češtině a angličtině, proto cílovým jazykem musí být vždy jeden z nich. Nic však nebrání v zavedení nového zdrojového jazyka. Pro jeho přidání je potřeba:

- pokud zdrojový jazyk obsahuje speciální znaky, nutno přidat znak do obrázkového fontu (viz obrázek 4.7)
- zaregistrovat dvojici zdrojový - cílový jazyk v příslušném výčtu v modulu *options* (viz obrázek 4.8)
- přidat lokalizační klíč a překlady pro zobrazení výběru dvojice jazyků v nastavení do modulu I18N
- poskytnout slovník v CSV (Comma Separated Values – hodnoty oddělené čárkami) formátu, kde první slovo na každé řádce je slovo ve zdrojovém jazyce, ostatní hodnoty na řádce jsou významy slova v jazyce cílovém (`vergh;car;vehicle`)
- naimportovat tento slovník do databáze za využití připravené funkcionality (viz obrázek 4.9)

```
// options.js
var languages = {
  KLINGON_EN: {
      id: 'KLINGON_EN',
      getText: APP.I18N.keys.OPTIONS_LANGUAGE_KLINGON_EN.getText,
      source: 'klingon',
      target: 'en'
  },
  //...
};
```

Obrázek 4.8: *Příklad zavedení dvojice „klingon“ - „en“ do modulu options.*

```
> python
>>> from db.import_dict import import_dict
>>> import_dict('/dictionaries/klingon_en.csv', 'klingon', 'en')
```

Obrázek 4.9: *Příklad importu slovníku „klingon“ → „en“.*

4.7 Návrhové vzory

V aplikaci `trogASR` je využita celá škála návrhových vzorů. Převážně byly tyto zapojovány pro zpřehlednění hektického a chaotického světa JS. Některé samospádem vyplývají z konceptu zpracování, některé jeden ze druhého. Následuje výčet a popis některých z nich.

4.7.1 Duck-typing

> *„Pokud vidím ptáka, který chodí jako kachna, plave jako kachna a kváká jako kachna, tak o tomto ptáku tvrdím, že je to kachna.“*
>
> — James Whitcomb Riley

Duck-typing (DT) je metoda dynamického typování objektů, při které zkoumáme objekt z hlediska jeho vlastností a ne z hlediska implementovaných rozhraní a předků. Tato metoda se chová podobně, jako kdybychom nějaké rozhraní vskutku implementovali, nemusíme tak však činit; stačí nám implementovat metody, které rozhraní udává.

Zmíněný epigraf tedy není stoprocentně spojitelný s principem DT. Po drobné úpravě raději tvrdíme, že „pokud vidím něco, co umí chodit, kvákat a plavat, pak věřím tomu, že je to kachna“.

Důležité je pozorování, že stačí vidět „něco“, a že poté „věříme“, že to je daného typu. Kdybychom pomocí DT například hledali mezi dostupnými třídu `Enemy` jako implementora metody `engage()`, mohli bychom se velmi divit, kdybychom jako první našli třídu `Girlfriend`.

Jelikož JS je jazyk, který nezná koncept rozhraní, DT je jediná možnost, jak typovost zajistit. `TrogASR` aplikace tuto možnost využívá při skládání modulár-

ního systému.

4.7.2 Module pattern

Module pattern je vzor určený pro organizaci JS aplikace. Zapouzdřuje vznik jednotlivých modulů do volání anonymní funkce, kde modul může vytvořit svou vnitřní strukturu. Ven pak vysouvá pouze API, které je s to poskytnout.

Tímto postupem třímáme schopnost dostat do lokálního rámce globální proměnné, které by JS interpret jinak musel složitěji vyhledávat v celém kódu. Jednoduše je pošleme jako parametry tovární funkce. Máme takto postaráno o přehlednější strukturu a jsme vyzbrojeni alespoň částečnou ochranou proti injekci potenciálně škodlivého kódu.

Aplikace `trogASR` — její prezentační část — je na tomto vzoru celá postavena.

4.7.3 Scope cache pattern

Jelikož JS je jazyk s funkcionálními prvky, kde proměnná běžně může být typu funkce, musí znát princip *scope* (rámec). Vnořené funkce mají vždy přehled o svém rámci a o v něm dostupných proměnných. Proto můžeme elegantně vytvářet statické funkce, které dokážou mít přístup do dynamického obsahu je obklopujícího.

Tím, že zavádíme a vydáváme funkce, které s sebou nesou informaci o stavu svého tvůrce, lze obejít jindy nevyhnutelnou nutnost statického předávání konfigurační informace, nebo nutnost posílání extrémního množství funkčních argumentů.

Tohoto vzoru aplikace `trogASR` využívá poměrně nemálo; všude, kde lze. Exemplárním případem budiž generování statických `getText()` metod na lokalizačních klíčích v I18N modulu, které se vždy při svém volání zařizují podle dynamické konfigurace modulu.

4.7.4 Singleton pattern

Název *singleton pattern* vychází z matematické definice, která jako *singleton* označuje množinu s právě jedním prvkem. V informatice se termínem *singleton* ozna-

čuje třída, jež se instancuje vždy maximálně jednou.

Ve světě JS je tento vzor poměrně obvyklý. Často jsou zde využívány globální proměnné, kterými jsou objekty zrovna typu *singleton*. Aplikace `trogASR` v tomto ohledu není výjimkou a vzor využívá též. Aplikační moduly jsou všechny tohoto typu.

4.7.5 Currying

Currying, česky také curryfikace, se jako termín poprvé objevuje v jazyce `Haskell` a získal svůj název (stejně jako jazyk) podle amerického matematika a logika Haskella Curryho.

Currifikace demonstruje základní paradigma funkcionálního programování. Jedná se o částečnou aplikaci funkce na parametry, kde výsledkem je opět funkce, která přijímá parametry dosud neaplikované. Obrázek 4.10 ukazuje kontrast curryfikace v Haskellu a v JS.

```
-- Haskell
ghc> let add x y = x + y    -- add :: (Num a) => a -> a -> a
ghc> add 2 3
5
ghc> let addTwo = add 2
ghc> addTwo 3
5

----------

// JS
var add = function(x, y) { return x + y; };
add(2, 3); // -> 5

var addTwo = (function(x) {
    return function(y) { return add(x, y); };
})(2);
addTwo(3); // -> 5
```

Obrázek 4.10: *Příklad curryfikace v Haskellu a v JS.*

Curryfikace je v aplikaci `trogASR` využito při zpracování templatů, kdy se nejprve vyrábí funkce, která template podle vstupu částečně upraví, a teprve

poté se tato aplikuje na příchozí data.

4.7.6 MVC pattern

MVC (Model-View-Controller) je jediný návrhový vzor, který byl využit v aplikaci `trogASR` na *back-end*u (serverové části). Není však pravidlem, že by MVC vzor měl být pouze tam. Například *front-end* framework `ExtJS` od verze 5.0 přináší možnost nasazení MVC i na prezentační vrstvě.

MVC vzor diktuje rozdělení aplikace na tři části — datový *model*, uživatelské rozhraní (*view*) a řídící logiku (*controller*) — tak, aby modifikace kterékoli této části měla minimální vliv na části ostatní. Vpravdě se nejedná ani tak o vzor návrhový, jako spíše o vzor architektonický (někdy také agregační). Týká se totiž výrazněji architektury aplikace, než-li klasické vzory návrhové.

`Flask` *microframework* tento vzor implicitně podporuje. Proto bylo jeho nasazení více, než žádoucí. Vzor je navíc podpořen specifickou strukturalizací zdrojového kódu.

Závěr

Seznam použité literatury

[1] Bahl, L. R., Jelinek, F., Mercer, R. L.
A maximum likelihood approach to continuous speech recognition.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 1983

[2] Deoras, A., Filimonov, D., Harper, M., Jelinek, F.
Model combination for Speech Recognition using Empirical Bayes Risk minimization.
IEEE Spoken Language Technology Workshop (SLT), 2010

[3] Bahl, L., Baker, J., Cohen, P., Dixon, N.R., Jelinek, F., Mercer, R.L., Silverman, H.F.
Preliminary results on the performance of a system for the automatic recognition of continuous speech.
Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '76, 1976

[4] Glass, James, Zue, Victor
6.345 Automatic Speech Recognition.
Massachusetts Institute of Technology: MIT OpenCourseWare, Spring 2003.
License: Creative Commons BY-NC-SA

[5] Tohkura, Y.
A Weighted Cepstral Distance Measure for Speech Recognition.
IEEE Trans. ASSP, Vol. ASSP-35, No. 10, 1414-1422, 1987

[6] Bogert, B.P., Healy, M.J.R., Tukey, J.W.
The frequency alanysis of time series for echoes: cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking.
Time Series Analysis, M. Rosenblatt, Ed. New York: Wiley, ch.15, 1963

[7] Byeongkyu, Ko, Dongjin, Choi, Chang, Choi, Junho, Choi, Pankoo, Kim
Document Classification through Building Specified N-Gram.

Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Sixth International Conference 2012

[8] Masataki, H., Sagisaka, Y., Hisaki, K., Kawahara, T.
Task adaptation using MAP estimation in N-gram language modeling.
Acoustics, Speech, and Signal Processing (vol.2), IEEE International Conference 1997

[9] Jelinek, F., Mercer, R.
Interpolated estimation of Markov source parameters from sparse data.
Proceedings of Workshop on Pattern Recognition in Practice, pg.381-397, 1980

[10] Katz, S.
Estimation of probabilities from sparse data for the language model component of a speech recognizer.
Acoustics, Speech and Signal Processing, IEEE Transactions, 1987

[11] Church, K., Gale, W.
A Comparison of the Enhanced Good-Turing and Deleted Estimation Methods for Estimating Probabilities of English Bigrams.
Computer Speech & Language, 1991

[12] Chien-Lin, Huang, Hori, C., Kashioka, H.
Semantic inference based on neural probabilistic language modeling for speech indexing.
Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference 2013

[13] Goblirsch, D.M.
Viterbi beam search with layered bigrams.
Spoken Language, Fourth International Conference, 1996

[14] Poritz, A.
Hidden Markov models: a guided tour.
Acoustics, Speech, and Signal Processing, International Conference, 1988

[15] Han, Shu, Hetherington, I.L., Glass, J.
Baum-Welch training for segment-based speech recognition.
Automatic Speech Recognition and Understanding, IEEE Workshop 2003

[16] Kingsbury, B.
Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling.
Acoustics, Speech and Signal Processing, IEEE International Conference 2009

[17] Tarar, S.
Speech analysis: Desktop items activation using Dynamic time warping algorithm.
Computer Science and Information Technology (ICCSIT), 3rd IEEE International Conference 2010

[18] Rasipuram, Ramya, Magimai.-Doss, Mathew,
Acoustic and Lexical Resource Constrained ASR using Language-Independent Acoustic Model and Language-Dependent Probabilistic Lexical Model.
Idiap Research Report, 02-2014

[19] Schluter, R., Nussbaum-Thom, M., Ney, H.
On the Relationship Between Bayes Risk and Word Error Rate in ASR.
Audio, Speech, and Language Processing, IEEE Transactions, 2010

[20] Korviny, Petr
Moodle (nejen) na OPF.
OPF, 2005

[21] Stockley, Derek
http://www.derekstockley.com.au/elearning-definition.html, 2003

[22] Průcha, Jan, Walterová, Eliška, Mareš, Jiří
Pedagogický slovník.
Pedagogický slovník s.66, Praha, Portál, 2003

[23] Fongling, Fu, Sheng-Chin, Yu

The Games in E Learning Improve the Performance.

Information Technology Based Higher Education and Training, 7th International Conference, 2006

[24] Graven, O.H., MacKinnon, L.

Exploitation of games and virtual environments for e-learning.

Information Technology Based Higher Education and Training, 7th International Conference, 2006

[25] Xue-min, Zhang, Mao, Li, Bin, Yang, Liu, Chang

Violent Components and Interactive Mode of Computer Video Game on Player's Negative Social Effect.

Intelligent Information Technology Application, Third International Symposium, 2009

[26] von Ahn, Luis

Games With A Purpose.

Institute of Electrical and Electronics Engineers Computer Magazine, vol. 39, no. 6, 2006

[27] von Ahn, Luis, Dabbish, Laura

Labeling images with a computer game.

In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 319-326, 2004

[28] von Ahn, Luis, Maurer, Benjamin, McMillen, Colin, Abraham, David, Blum, Manuel

reCAPTCHA: Human-Based Character Recognition via Web Security Measures.

Science, vol. 321, no 5895, pp. 1465-1468, 2008

Seznam použitých zkratek

AJAX	Asynchronous JavaScript and XML
AM	Acoustic Model
API	Application Programming Interface
ASR	Automatic Speech Recognition
CA	Cepstral Analysis
CS	Crowdsourcing
CSS	Cascading Style Sheets
CSV	Comma Separated Values
DOM	Document Object Model
DT	Duck-Typing
DTFT	Discrete-Time Fourier Transform
EL	E-Learning
FHD	Full HD
GIL	Google Image Labeler
GUI	Graphic User Interface
GWAP	Game With a Purpose
HD	High Definition
HMM	Hidden Markov Model
HMP	Hidden Markov Process
HTML	HyperText Markup Language
I18N	Internationalization
ISO	International Organization for Standardization
JS	JavaScript
JSON	JavaScript Object Notation
LD	Levenshtein Distance
LES	Local Emission Score
LM	Language Model
LTI	Linear Time-Invariant
MAP	Maximum a Posteriori
MVC	Model View Controller

MVC	Model-View-Controller
OOP	Object Oriented Programming
PC	Personal Computer
STFA	Short-Time Fourier Analysis
TrogASR	Translating Online Game using Automatic Speech Recognition
TS	Transition Score
UHD	Ultra HD
WER	Word Error Rate

Přílohy