Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra kybernetiky

DIPLOMOVÁ PRÁCE

Kombinace výstupů rozpoznávačů založených na odlišných principech

Plzeň, 2013 Adam Chýlek

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne Adam Chýlek

Děkuji vedoucímu diplomové práce Ing. Janu Švecovi za cenné rady, připomínky a metodické vedení práce.
Děkuji také MetaCentru za možnost využití výpočetní sítě, kterou poskytuje v rámci projektu „Velká infrastruktura CESNET“ (LM2010005).

Anotace

Tato práce se zabývá kombinací fonémových mřížek a slovních gramatik, které může být využito pro porozumění v hlasových dialozích prostřednictvím detekce klíčových slov nebo též pro vyhledávání v archivech mluvené řeči. Metody zvolené pro dosažení tohoto cíle staví na teorii vážených konečných automatů a v práci jsou uplatněny též metody strojového učení pro klasifikaci. Je diskutován vliv některých parametrů strojového učení (např. velikosti trénovací množiny) na kvalitu detekce.

Klíčová slova: konečné automaty, detekce klíčových frází, klasifikace, OpenFst

Annotation

This thesis discusses a combination of phoneme lattices and word grammars that can be used for understanding of spoken language via spoken term detection or for searching in audio archives. The methods chosen to reach this goal are based on the theory of finite state machines together with methods used in machine learning for classification. Discussed are also the effects of some of the parameters important in the field of machine learning (e.g. the size of training set) on the quality of detection.

Keywords: finite state automata, spoken term detection, classification, OpenFst

Obsah

1. Úvod

Komunikace člověka s technikou se čím dál více snaží přiblížit přirozené komunikaci mezi lidmi. Jednou z oblastí, ve které je třeba zkoumat nové možnosti a prostředky ke zlepšení kvality této komunikace, je oblast hlasových dialogů.

V oblasti hlasových dialogových systémů je důležité především rozpoznání a porozumění promluvám uživatele, přirozenost pak ovlivňuje také zajištění správné reakce na uživatelský vstup. Cílem této práce je zvolit a zanalyzovat možné kombinace dostupných prostředků za účelem vytvoření robustnějšího a rychlejšího porozumění.

Jako prostředky vhodné pro kombinaci byly zvoleny fonémové mřížky získané z řečového signálu fonémovým rozpoznávačem a gramatiky určující na slovní úrovni sémantické entity, které systém využívá jako popis informací důležitých pro porozumění mluvené řeči. V této práci bude popsána jak metoda kombinace těchto dvou výstupů pomocí konečných automatů, tak samotná detekce sémantických entit na datech získaných z reálných vstupů uživatelů.

Práce je členěna do osmi kapitol, kdy po této úvodní kapitole následuje v 2. kapitole teorie konečných automatů, které jsou stěžejními modely pro celou práci a jsou dále kombinovány s klasifikátory popsanými v 3. kapitole. Kapitola 4. zasazuje tuto práci do kontextu hlasových dialogů a následující kapitoly se věnují již vlastnímu přínosu práce právě v tomto kontextu. Popis zvoleného přístupu v kapitole 5 a jeho implementace v kapitole 6 přechází v experimenty a diskuzi vhodných parametrů v kapitole 7.

2. Konečné automaty

Konečné automaty jsou hojně využívány v oblasti rozpoznávání řeči a dialogových systémů a úzce souvisí s gramatikami a formálními jazyky. Jedná se o výpočetní model umožňující rozhodnout, zda je vstupní výraz součástí jazyka generovaného určitou gramatikou specifického typu. Pro tyto tzv. regulární gramatiky platí jistá omezení, takže před samotnou formální definicí konečných automatů stanovme co je to gramatika, jak musí gramatika vypadat a jak je definován jazyk touto gramatikou generovaný.

2.1 Gramatiky

Deterministickou gramatikou $G = (V_n, V_t, P, S)$ rozumíme množinu terminálních symbolů (terminálů) V_t, neterminálních symbolů (neterminálů) V_n, odvozovacích pravidel P a počátečního symbolu S. Pro symboly platí $V_t \cap V_n = \emptyset$. V dalším textu použijeme následující konvenci značení:

- terminály jsou značeny malými písmeny latinky $(a, b, c, \ldots \in V_t)$;
- neterminály značíme velkými písmeny latinky $(A, B, C, \ldots \in V_n)$;
- V^+ značí množinu všech neprázdných posloupností vytvořených z prvků množiny V. V^* pak označuje množinu $V^* = V^+ \cup \{\epsilon\}$, kde ϵ reprezentuje prázdný řetězec[1];
- řetězce terminálů a/nebo neterminálů jsou značeny malými písmeny řecké abecedy $(\alpha, \beta, \gamma, \ldots \in (V_n \cup V_t)^*)$, přičemž ϵ zůstává vyhrazeno pouze prázdnému řetězci.

Pravidla mají formát $\alpha \rightarrow \beta$, nebo-li řetězec α se přepíše na řetězec β. V obecném případě uvažujeme $\alpha, \beta \in (V_n \cup V_t)^*$. Pokud mají navíc pravidla pravděpodobnostní ohodnocení p_i (tedy $\alpha \xrightarrow{p_i} \beta$), hovoříme o stochastických gramatikách.

[1] V práci použité softwarové prostředky pro vykreslování konečných automatů využívají označení `<eps>`, které bude použito u obrázků

Na základě omezení symbolů u odvozovacích pravidel vzniklo více typů gramatik zařazených v tzv. Chomského hierarchii.

Definujme též, co budeme chápat jako jazyk generovaný gramatikou. Jazyk je podmnožinou množiny V_t^* všech slov nad abecedou V_t a je z gramatiky generován aplikací přípustných pravidel. Máme-li dánu gramatiku, můžeme vždy rozhodnout, zda nějaký řetězec w patří do jazyka $L(G)$ generovaného gramatikou G. Takové ověření je možné provést Turingovým strojem. Tato metoda je sice dostatečně obecná, ovšem není efektivní [17]. Jednotlivé specializované typy gramatik pak umožňují ověření jednodušími způsoby, v nejjednodušším případě konečnými automaty, za cenu snížení vyjadřovací schopnosti. Tyto gramatiky jsou dále popsány a využity v této práci.

Gramatiky typu 0 (frázové gramatiky)

Pro tyto gramatiky mají pravidla obecnou formu $\alpha \to \beta$, kde $\alpha, \beta \in (V_t \cup V_n)^*$. Na pravidla tedy nejsou kladena žádná omezení a jsou tak nejsilnějším typem gramatik s největší vyjadřovací schopností. Tyto gramatiky generují tzv. rekurzivně spočetné jazyky a o příslušnosti řetězce do daného jazyka pak může rozhodnout nějaký Turingův stroj. Jedním z problémů pak může být, že u řetězce, který do jazyka nepatří, může stroj skončit v nekonečné smyčce.

Kontextové gramatiky

Pravidla mají formu $\alpha A \beta \to \alpha \omega \beta$, kde $\alpha, \beta \in (V_t \cup V_m)^*$ a $\omega \in (V_t \cup V_m)^+$. Neterminál A je tedy nahrazován neprázdným řetězcem ω na základě svého kontextu. Dále je povoleno využít pravidlo $S \to \epsilon$. Jazyk generovaný touto gramatikou je označován za kontextový a řetězce přijímané tímto jazykem rozpoznává lineárně ohraničený Turingův stroj.

Bezkontextové gramatiky

Odstraněním kontextu v předchozím typu gramatik vznikají pravidla ve tvaru $A \to \omega$, kde opět $\omega \in (V_t \cup V_m)^+$ a taktéž můžeme použít přepis $S \to \epsilon$. Gramatika generuje bezkontextový jazyk rozpoznatelný zásobníkovým automatem. Tento

typ se využívá např. v současných hlasových dialozích v rámci standardu SRGS pro popis přípustných promluv [12].

Regulární gramatiky

Pravidla regulární gramatiky jsou omezena na tvar $A \to aB$ nebo $A \to a$, přičemž mohou obsahovat i pravidlo $S \to \epsilon$. Tyto gramatiky generují regulární jazyky rozpoznatelné konečnými automaty.

Pro praktické použití u dialogových systémů může být přípustné provést aproximaci a nahradit rekurzi neterminálu v bezkontextových gramatikách pouhým omezeným počtem opakování. Vznikne tím již regulární gramatika umožňující využít konečné automaty pro rozhodnutí o přijetí vstupního řetězce [19]. Můžeme tak využít matematické krásy a jednoduchosti konečných automatů ke zrychlení potřebných výpočtů.

2.2 Polookruhy

Před definicí vážených konečných automatů přibližme nejdříve pojem polookruh, který definuje obor hodnot pro váhy na hranách automatů a zavádí operátory použité při operacích nad váženými konečnými automaty.

Polookruh je množina $\mathbb{K}$ vybavená asociativními operacemi $\oplus$, popř. $\otimes$ s neutrálními prvky $\bar{0}$, popř. $\bar{1}$, přičemž operace $\otimes$ je dvojně distributivní vzhledem k $\oplus$ a $\bar{0}$ je nulový prvek [20]. Polookruh můžeme značit jako uspořádanou pětici $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$.

Jedním z možných ohodnocení používaných u konečných automatů je pravděpodobnostní ohodnocení. Takové ohodnocení můžeme formálně vytvořit nad pravděpodobnostním polookruhem $(\mathbb{R}, +, \cdot, 0, 1)$. Další možností je použití záporných logaritmů pravděpodobnosti logaritmickým polookruhem $(\mathbb{R} \cup \{\infty\}, \oplus, +, \infty, 0)$, kde pro operátor $\oplus$ platí $a \oplus b = -\log\left(e^{-a} + e^{-b}\right)$. V případě použití záporných logaritmů pravděpodobnosti lze využít i tzv. tropický polookruh $(\mathbb{R} \cup \{\infty\}, \oplus, +, \infty, 0)$, kde pro operátor $\oplus$ platí $a \oplus b = \min\{a, b\}$.

Jedním z důvodů zavedení logaritmů pravděpodobností je omezení plynoucí z reprezentace čísel v počítači. Při použití pravděpodobnosti, tedy čísel z intervalu

$[0; 1]$ může dojít ke zpracování tak malých čísel, že dojde k podtečení. Použití logaritmu pravděpodobnosti možnost takovéto chyby minimalizuje, zde však může dojít naopak k přetečení. Logaritmus je záporný aby bylo možné ohodnocení zapsat pomocí kladných čísel a umožňuje nad konečnými automaty využití standardních metod prohledávání grafu [20]. Nad tropickým polookruhem můžeme definovat operace jako je nejlepší cesta, proces je analogický s Viterbiho algoritmem.

2.3 Vážené konečné automaty

Konečné automaty můžeme považovat za nástroj umožňující diskrétně v čase modelovat určité systémy pomocí stavů, přičemž na základě vstupu může dojít k přechodu mezi jednotlivými stavy. Obecně můžeme využít stochastického přístupu, kdy jsou přechody ohodnoceny váhami. Takové modely pak nazýváme váženými konečnými automaty (FSM[2]). Z hlediska výstupu můžeme definovat automaty jako akceptory nebo transducery. V textu budou využity definice podle [16].

Vážené konečné akceptory

Váženým konečným akceptorem A nad polookruhem $\mathbb{K}$ rozumíme uspořádanou sedmici $A = (\Sigma, Q, E, i, F, \lambda, \rho)$, kde

- Σ je množina vstupních symbolů (abeceda),
- Q je množina stavů tohoto automatu,
- E je množina přechodů $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times \mathbb{K} \times Q$
- i je počáteční stav, $i \in Q$
- F je množina koncových stavů, $F \subseteq Q$.
- λ ohodnocení počátečního stavu a
- ρ ohodnocující funkce koncových stavů.

[2]zkratka z anglického Finite State Machines

Symbol ϵ reprezentuje prázdný řetězec.

V tomto akceptoru rozumíme přechodem nebo též hranou uspořádanou čtveřici $t = (p[t], l[t], w[t], n[t]) \in E$, kde $p[t]$ je předchozí stav, $n[t]$ cílový stav, $l[t]$ symbol příslušející hraně a $w[t]$ váha přechodu.

Dále pak definujeme cestu automatem jako posloupnost přechodů $t_1, \ldots, t_n$, kde $n[t_i] = p[t_{i+1}], i = 1, \ldots, n-1$.

Výstupem váženého konečného akceptoru je informace o přijetí či zamítnutí vstupního řetězce. Automat A přijímá vstup α, pokud $\alpha \subseteq \Sigma$ a zároveň existuje alespoň jedna cesta $\pi = t_1, \ldots, t_n$, která automat přivede z počátečního stavu i do stavu $f \in F$, tedy $\alpha = l[\pi] = l[t_1], \ldots, l[t_n]$. V případě přijetí řetězce je výstupem i váha $w[\pi] = \lambda \otimes w[t_1] \otimes \ldots \otimes w[t_n] \otimes \rho(n[t_n])$.

Vážené konečné transducery

O vážených konečných transducerech můžeme uvažovat jako o zobecnění vážených konečných akceptorů. Transducery se neomezují pouze na rozhodnutí o přijetí či zamítnutí vstupního řetězce α, ale zároveň poskytují i výstupní řetězec β.

Transducer T nad polookruhem $\mathbb{K}$ definujeme jako osmici $T = (\Sigma, \Omega, Q, E, i, F, \lambda, \rho)$, kde

- Σ je množina vstupních symbolů,
- Ω je množina výstupních symbolů,
- Q je množina stavů tohoto automatu,
- E je množina přechodů $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Omega \cup \{\epsilon\}) \times \mathbb{K} \times Q$
- i je počáteční stav, $i \in Q$
- F je množina koncových stavů, $F \subseteq Q$.
- λ ohodnocení počátečního stavu a
- ρ ohodnocující funkce koncových stavů.

Symbol ϵ opět reprezentuje prázdný řetězec.

Hrana je nyní definována jako pětice $t = (p[t], l_i[t], l_o[t], w[t], n[t])$, kde kromě předchozího stavu $p[t]$,váhy $w[t]$ a následujícího stavu $n[t]$ definovaných u akceptorů využíváme navíc $l_i[t]$ jako vstupní symbol hrany a $l_o[t]$ jako výstupní symbol hrany.

Cesta je zde opět definována stejně jako u akceptorů jako posloupnost $\pi = t_1, \ldots, t_n$, kde $n[t_i] = p[t_{i+1}], i = 1, \ldots, n-1$ a stejně tak váha dané cesty zůstává definována jako $w[\pi] = \lambda \otimes w[t_1] \otimes \ldots \otimes w[t_n] \otimes \rho(n[t_n])$. Navíc je třeba definovat vstupní řetězec (posloupnost vstupních symbolů) jako $\alpha = l_i[\pi] = l_i[t_1], \ldots, l_i[t_n]$ a také výstupní řetězec jako $\beta = l_o[\pi] = l_o[t_1], \ldots, l_o[t_n]$.

Z hlediska definice výstupní funkce hovoříme v tomto případě o automatu Mealyho typu. Formálně je výstup automatu tohoto typu k dispozici v okamžiku přijetí vstupu. Alternativou je automat Mooreův, který závisí na stavu, do kterého automat přijetím vstupu přejde a tento výstup pak setrvává do doby přijetí dalšího vstupu. Oba typy jsou vzájemně převoditelné [9]. Pokud v textu nebude uvedeno jinak, budeme pojmem vážené konečné transducery myslet právě automaty Mealyho typu.

Ekvivalence

Využijeme definice ekvivalence z [9]. Ekvivalenci vyžadujeme především při použití operací optimalizující konečné automaty jako je odstranění ϵ přechodů, determinizace a minimalizace v 2.4.

Mějme automaty $M_1 = (\Sigma, \Omega, E_1, i_1, F_1, \lambda_1, \rho_1)$ a $M_2 = (\Sigma, \Omega, E_2, i_2, F_2, \lambda_2, \rho_2)$. Uvažujme stav $q_1 \in E_1$ z automatu M_1, resp. $q_2 \in E_2$ z automatu M_2. Tyto stavy jsou ekvivalentní, pokud pro každé neprázdné slovo $v \in \Sigma^+$ platí $E_1^+(q_1, v) = E_2^+(q_2, v)$.

Dva automaty M_1 a M_2 jsou ekvivalentní pokud pro každý stav q_1 automatu M_1 existuje ekvivalentní stav q_2 automatu M_2 a zároveň pro každý stav q_2 automatu M_2 existuje ekvivalentní stav q_1 automatu M_1.

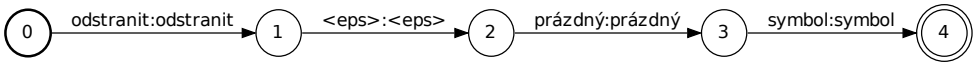
2.4 Operace nad konečnými automaty

V této práci využíváme několik operací nad konečnými automaty. Čtenáři zde přiblížíme jejich obecný popis a praktický význam. Přesný popis kroků jednotlivých operací je nad rámec teoretického základu této práce a čtenář nalezne podrobný popis v mnohých zdrojích, např. [6] nebo [9]. Pro běžné využití jsou tyto operace implementovány v použitých knihovnách OpenFst a pyfst.

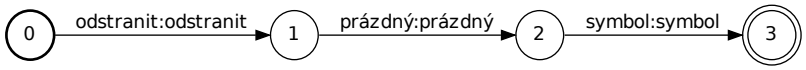
Před samotným popisem operací je vhodné přiblížit ještě formát vyobrazení konečných automatů v této práci. Pro ilustraci využijme obr. 2.1, kde vidíme, že stavy jsou značeny kružnicí, přičemž počáteční stav (0) je vyznačen tučnější čarou a pro koncové stavy (zde jediný stav 4) je přidána další soustředná kružnice. Uvnitř stavů je pak jejich číselný index, u koncového stavu případně za lomítkem i jeho váha (neuvádí se, pokud je rovna $\bar{1}$ nad daným polookruhem). Přechody jsou značeny šipkami mezi stavy, nad hranou je dvojtečkou oddělený vstupní a výstupní symbol, za lomítkem následovaný případným ohodnocením přechodu.

Odstranění ϵ přechodů

V konečných automatech můžeme vytvářet přechody které obsahují prázdné symboly. Tyto hrany zavádí do automatů neurčitost, automat přes ně může automat projít bez čtení vstupního symbolu. Přesto se hrany s prázdnými symboly používají např. pro zjednodušení vytváření konečných automatů. Odstraněním ϵ přechodů vzniká automat ekvivalentní původnímu. Ekvivalence je patrná z obrázků 2.1 a 2.2, kde snadným výčtem přijímaných řetězců zjistíme, že oba automaty přijímají stejné řetězce.



Obrázek 2.1: Automat se symbolem prázdného řetězce ϵ označeného <eps>



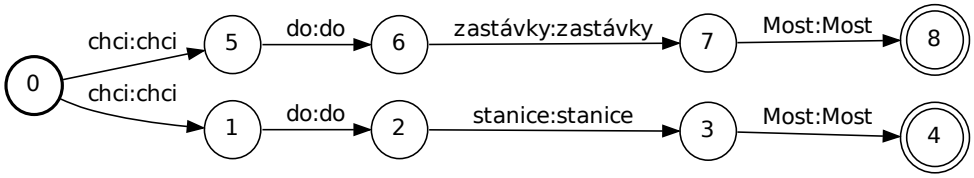
Obrázek 2.2: Automat po odstranění hran se symbolem prázdného řetězce

Determinizace

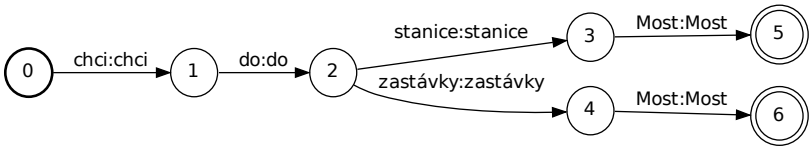
Nedeterministickým automatem rozumíme takový automat, v kterém nalezneme z jednoho stavu vycházejících více hran se stejným vstupním symbolem. Účelem determinizace je takové hrany sloučit, protože zde opět zavádí do automatu neurčitost. Výsledný automat je ekvivalentní s původním.

Příklad nedeterministického akceptoru je na obr. 2.3. Je patrné, že při vstupu začínajícím „chci do“ algoritmus musí udržovat dvě možné cesty. Deterministická verze tohoto automatu je pak na obr. 2.4.

Dodejme, že determinizace transducerů není vždy možná a při použití metod v OpenFST[3] není zaručena konečná doba běhu použitého algoritmu.



Obrázek 2.3: Nedeterministický automat



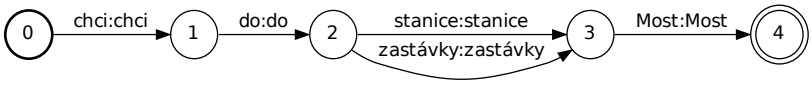
Obrázek 2.4: Determinizovaný automat z obr. 2.3

[3]Knihovna pro práci s konečnými automaty, více v části 2.5

Minimalizace

Poslední metodou optimalizace konečných automatů je minimalizace. Konečný transducer obsahující minimálně dvě různé cesty π_1, π_2, kde $l_i[\pi_1] = l_i[\pi_2] \wedge l_o[\pi_1] = l_o[\pi_2]$ označíme za neminimální. Obdobně pak konečný akceptor je neminimální když $l[\pi_1] = l[\pi_2]$.

Algoritmus minimalizace takové cesty odstraní a vytvoří minimální konečný automat ekvivalentní s původním automatem, jak je patrné z obr. 2.5. Minimalizace vyžaduje, aby byl původní automat deterministický.



Obrázek 2.5: Minimalizovaný automat z obr. 2.4

Konkatenace

Uvažujme automat U, v kterém jsou vstupní řetězce α přepsány na řetězce β s váhou a, a automat V, který vstupy γ přepíše na řetězce δ s váhou b. Konkatenací UV rozumíme automat, přepisující všechny vstupní řetězce $\alpha\gamma$ na $\beta\delta$ s vahou $w = a \otimes b$. [1]

Příkladem je zřetězení akceptorů na obr. 2.6c.

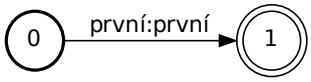
Sjednocení

Uvažujme automat U, v kterém jsou vstupní řetězce α přepsány na řetězce β s váhou a, a automat V, který vstupy γ přepíše na řetězce δ s váhou b. Sjednocením $U \cup V$ rozumíme automat, přepisující všechny vstupní řetězce α na β s vahou $w = a$ a všechny vstupní řetězce γ na δ s vahou $w = b$. [1]

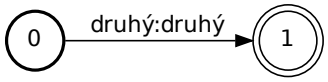
Příklad sjednocení je na obr. 2.6d.

Kompozice

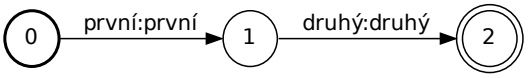
Kompozice $T = R \circ S$ transducerů R, S vytváří automat, který přepisuje řetězce u na řetězce w s tím, že všechny u z automatu R vytvářející výstupní řetězec v



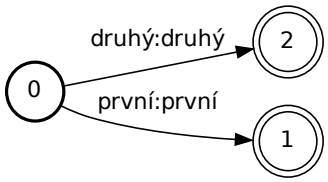
(a) Konečný automat A_1



(b) Konečný automat A_2



(c) Konkatenace $A_1 + A_2$



(d) Sjednocení $A_1 \cup A_2$

Obrázek 2.6: Ukázka operací konkatenace a sjednocení dvojice automatů

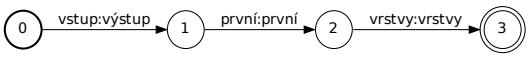
spáruje s výstupními řetězci w automatu S vytvořenými vstupem v. Váha takto spárovaných hran je určena $\otimes$-násobením vah příslušných cest v automatech R a S.

Příklad kompozice transducerů je na obr. 2.7. První transducer T_1 (obr. 2.7a) přepisuje řetězec „vstup první vrstvy“ na „výstup první vrstvy“. Druhý tranducer T_2 (obr. 2.7b) pak přepisuje řetězec „výstup první vrstvy“ na „výstup druhé vrstvy“. Kompozice $T_1 \circ T_2$ pak podle definice přepisuje řetězec „vstup první vrstvy“ na „výstup druhé vrstvy“.

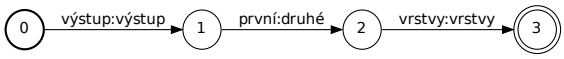
Pro operaci kompozice je možné využívat líné implementace, která kompozici provádí pouze u cest, které jsou nějakou další operací vyžadovány.

Knihovny OpenFST nabízí za určitých podmínek zjednodušení automatů R a S využitím tzv. ρ a $\sigma-$matcherů. Přes tyto prostředky může dojít k úpravě symbolů aktuálně zpracovávaných hran během kompozice automatů.

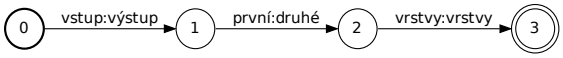
Konkrétně σ-matcher zavádí speciální symbol σ, při jehož použití na hraně automatu dochází při kompozici k určení shody s libovolným symbolem druhého automatu. ρ-matcher pak zavádí speciální symbol ρ, který při kompozici zastupuje všechny symboly, které se nenachází na jakékoliv jiné hraně jdoucí ze stejného stavu z kterého vychází právě hrana s ρ symbolem. V obou případech dojde k nahrazení speciální symbolu ρ či σ. [1]



(a) Konečný transducer T_1

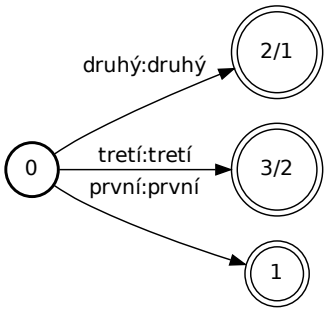


(b) Konečný transducer T_2

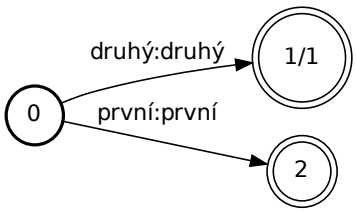


(c) Kompozice $T_1 \circ T_2$

Obrázek 2.7: Ukázka operace kompozice



(a) Výchozí transducer T_p



(b) Výsledek prořezání T_p s prahem $p = 1$

Obrázek 2.8: Ukázka prořezání konečného automatu

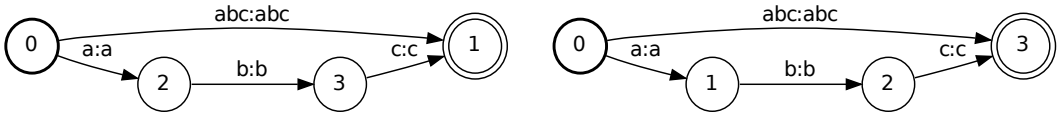
Prořezání

Prořezáním s prahem p vzniká konečný automat, který obsahuje pouze cesty s váhou maximálně $p\otimes$ větší, než je váha nejlepší (1-best) cesty původním automatem.

Příklad prořezání s prahem $p = 1$ je na obr. 2.8. Nejlepší hypotézou je zde řetězec „první“ s váhou $w_1 = 0$ nad tropickým polookruhem. Operace $\otimes$ u tropického polookruhu představuje součet, čili v automatu zůstanou jen cesty s váhou $w \leq w_1 + p$, tedy s váhou $w \leq 1$.

Topologické seřazení

Topologické seřazení můžeme provádět na acyklických automatech. Výsledkem je (v případě reprezentace uzlů číselnými indexy) přečíslování uzlů tak, že pro každou hranu $t = (p, l_i, l_o, w, n)$ platí, že uzel p má nižší index, než uzel n. [1]



(a) Výchozí konečný automat T_s (b) Výsledek seřazení stavů T_s

Obrázek 2.9: Topologické seřazení stavů automatu

Příklad topologického seřazení stavů je na obr. 2.9.

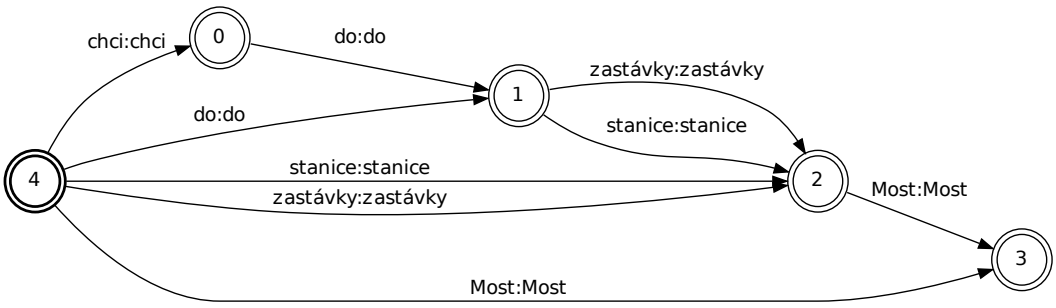
Faktorové automaty

Faktory řetězce α nad abecedou Σ rozumíme podle [7] množinu

$$F(\alpha) = \{\beta \in \Sigma^* | \exists \delta, \gamma \in \Sigma^*, \alpha = \delta\beta\gamma\}$$

Faktorovým automatem $F(\alpha)$ pak označíme takový minimální konečný akceptor, který přijímá právě množinu faktorů řetězce α.

Pokud máme konečný akceptor A, pak můžeme faktorový automat $F(A)$ definovat jako minimální konečný akceptor přijímající právě jen řetězce faktorů všech řetězců, které akceptor A přijímá.



Obrázek 2.10: Automat z obr. 2.5 převedený na faktorový automat

2.5 Programové prostředky pro práci s konečnými automaty

V praktické části této práce budeme využívat k reprezentaci konečných automatů a k provádění operací s těmito automaty knihovnu OpenFst [1].Tato knihovna je napsaná v jazyce C++ a pro použití v jazyce Python je na katedře kybernetiky vyvíjen nástroj pyfst, který umožňuje provázání metod a tříd z OpenFst s Pythonem pomocí knihoven Boost.

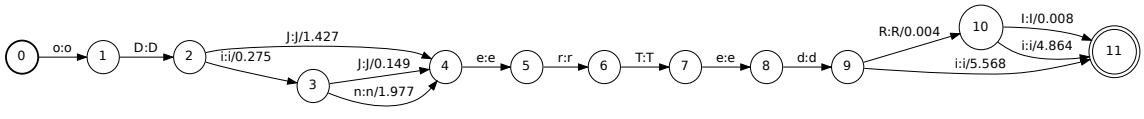
2.6 Mřížky

Výstupem systémů rozpoznávání řeči, případně systémů detekce klíčových slov, může být vážený konečný automat obsahující N nejlepších hypotéz. Takový automat nazveme mřížkou.

Mřížka reprezentuje N nejlepších posloupností fonémů, slov či sémantických entit, které rozpoznávač ve vstupu nalezl. Jednotlivé hrany obsahují právě symboly fonémů (resp. slov, či sémantických entit) a stavy pak uchovávají časové zarovnání. Tyto mřížky jsou pro uchování hypotéz efektivnější, než prostý seznam N nejlepších hypotéz. [20] S výhodou také můžeme využít operací s konečnými automaty pro další zpracování.

Dodejme zde, že se symboly fonémů liší od běžné abecedy, v mřížkách se tedy setkáme s abecedou $P = \{$A, B, C, D, E, F, I, J, M, N, O, Q, R, S, T, U, Y, Z, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, x, y, z$\}$.

Příklad mřížky reprezentované jako konečný transducer je na obr. 2.11.



Obrázek 2.11: Fonémová mřížka

Formát MLF

Jedním z formátů pro textovou definici mřížek je formát využívaný v souborech štítků HTK Label Files. Předpokládá se, že jednotlivé štítky reprezentují transkripce segmentů řeči v nahrávce. Tyto Label soubory mohou být sdruženy do Master Label File (MLF). [25]

Každý řádek souboru štítků má formát

`[start  [end] ] name [score] { auxname [auxscore] } [comment]`

přičemž `start` je čas začátku, `end` čas konce, `name` samotný štítek, `score` ohodnocení štítku, `auxname` obdoba tagu - výstupního řetězce - ze SRGS gramatik.

V samotném MLF pak po hlavičce `#!MLF!#` následuje vždy cesta k původnímu Label souboru a samotný obsah tohoto souboru zakončený tečkou pro všechny zahrnuté soubory, jak je vidět na následujícím příkladu tohoto souboru:

```
#!MLF!#
"*/CD03~Part1~04-000427-080612-1_007.rec"
100000 1800000 a 1.000000
1800000 2800000 n 1.000000
2800000 5700000 o 1.000000
5700000 11900000 # 0.150600
.
"*/CD03~Part1~04-000427-075424-1_017.rec"
100000 1400000 a 0.924020
1400000 2500000 k 0.932197
2500000 4300000 i 0.995992
4300000 6100000 a 1.000000
6100000 7000000 n 1.000000
7000000 9100000 o 1.000000
.
```

Tento příklad ukazuje první nejlepší posloupnosti fonémů ze dvou mřížek. Časy jsou zde v řádech 10^{-4} ms. V příkladu též můžeme pozorovat symbol, který do fonémové abecedy nepatří - symbol `#` zde reprezentuje symbol prázdného řetězce.

3. Klasifikace

Klasifikace je jedním z druhů strojového učení. Cílem klasifikace je zařazení vstupního vektoru příznaků x do jedné z K tříd C_k, kde $k = 1, \ldots, K$. Podle průběhu trénování klasifikátorů pak také hovoříme o učících se systémech s učitelem a bez učitele. V rámci této práce budeme využívat pouze učení s učitelem, kdy je ve fázi trénování vstupem systému dvojice (x,t), přičemž x je vektor příznaků $x = (x_1, \ldots, x_N) \in R^N$ a t je vektor který určuje příslušnost do k-té třídy $t = (t_1, \ldots, t_k, \ldots, t_K)$ a $t_k \in \{0, 1\}$. Tato sekce čerpá definice z [4].

V rámci této práce budeme uvažovat binární klasifikaci, kde počet tříd $K = 2$ a můžeme tedy zjednodušit $t \in \{0, 1\}$. Pro klasifikaci budou využity klasifikátory využívající logistické regrese a rozhodovacích stromů.

3.1 Logistická regrese

Logistická regrese je jedním z lineárních modelů pro klasifikaci.

Úkolem trénování klasifikátoru za použití lineárních modelů je stanovit takovou hranici, která rozdělí D dimenzionální prostor vstupu x $(D-1)$ dimenzionální nadrovinou. Logistická regrese využívá nelineárních transformací vstupu $\phi(x)$ tak, aby bylo možné (v původním prostoru lineárně neseparabilní) třídy rozdělit za použití lineárních modelů.

V případě logistické regrese hovoříme taktéž o diskriminativním modelu. Vycházíme-li z Bayesova vztahu

$$p(C_k|\phi) = \frac{p(\phi|C_k)p(C_k)}{p(\phi)}$$

pak diskriminativní model odhaduje přímo pravděpodobnost $p(C_k|\phi)$, zatímco generativní modely odhadují pravděpodobnost $p(\phi|C_k)$ při znalosti pravděpodobnosti $p(C_k)$ a teprve z nich je určena posteriorní pravděpodobnost $p(C_k|\phi)$.

Posteriorní pravděpodobnost u binární klasifikace můžeme také zapsat jako

$$p(C_1|\phi) = y(\phi) = \sigma(w^T\phi),$$

kde využíváme sigmoidální funkci

$$\sigma(a) = \frac{1}{1 + \exp(-a)}.$$

Dodejme, že $p(C_2|\phi) = 1 - p(C_1|\phi)$.

Parametry modelu za využití logistické regrese určujeme použitím principu maximální věrohodnosti s využitím věrohodnostní funkce

$$p(t|w) = \prod_{n=1}^{N} y_n^{t_n} \{1 - y_n\}^{1-t_n},$$

kde pro trénovací dvojice (ϕ_n, t_n) uvažujeme $t_n \in \{0, 1\}, \phi_n = \phi(x_n), y_n = p(C_1|\phi_n)$ a $t = (t_1, \ldots, t_n)$

Ztrátovou funkci pak definujeme jako

$$E(w) = -\ln p(t|w) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}$$

s gradientem

$$\bigtriangledown E(w) = \sum_{n=1}^{N} (y_n - t_n)\phi_n.$$

Pro trénování tak můžeme využít gradientních optimalizačních metod.

3.2 Rozhodovací stromy

Problém klasifikace je možné řešit i binárními rozhodovacími stromy. V každém uzlu je vybrán jeden příznak x_i z D dimenzionálního vektoru příznaků $\mathbf{x}_n$ a stanoven práh ϕ_m podle kterého dochází k větvení. Rozvětvením pak vznikají dva D-rozměrné intervaly, které mohou být dále nezávisle děleny, přičemž dále neděleným listům je přiřazena cílová třída. Dodejme, že i zde máme kromě množiny vstupních vektorů $\{\mathbf{x_1}, \ldots, \mathbf{x}_N\}$ k dispozici k nim příslušnou informaci učitele $\{t_1, \ldots, t_N\}$.

Důležitý je tedy správný výběr příznaků podle kterých se rozhoduje, ale také zastavující podmínka, aby nedošlo k přetrénování klasifikátoru. Běžnou praxí je sestavení většího stromu, který je následně prořezán. Práh prořezání je určen na základě kritéria vyrovnávající chybu vůči komplexitě modelu [4].

Definujme strom $T \subset T_0$, kde T_0 je výchozí strom pro prořezávání. Každý list tohoto stromu reprezentuje prostor R_τ mající N_τ referenčních bodů, kde τ je

index listů $\tau = 1, \ldots, |T|$. Pokud označíme $p_{\tau k}$ jako poměr referenčních bodů v regionu R_τ přiřazených do třídy k, kde $k = 1, \ldots, K$, pak pro výpočet kritéria prořezávání nejprve definujeme křížovou entropii

$$Q\tau(T) = \sum_{k=1}^{K} p_{\tau k} \ln p_{\tau k}$$

případně tzv. Gini index

$$Q\tau(T) = \sum_{k=1}^{K} p_{\tau k}(1 - p_{\tau k})$$

a hodnotu kritéria prořezávání stanovíme jako

$$C(T) = \sum_{\tau=1}^{|T|} Q_\tau(T) + \lambda|T|$$

s parametrem λ umožňujícím regularizaci. Hodnota parametru je stanovena křížovou validací.

V této práci využíváme modifikace rozhodovacích stromů ve formě extrémně znáhodněných stromů (Extra-Tree, ET). Tento přístup rozdělí trénovací množinu na N podmnožin a pro každou z těchto podmnožin pak vytvoří rozhodovací strom, v němž dává důraz na náhodný výběr příznaků x_n použitých pro větvení uzlu stromu. Rozhodnutí o příslušnosti vektoru příznaků do třídy je realizováno hlasováním jednotlivých stromů. Odhad posteriorní pravděpodobnosti je pak vypočítán na základě toho, kolik stromů rozhodlo pro příslušnost daného vektoru do třídy a kolik proti ní.

Parametr s jakým je přistupováno k randomizaci je možné vhodně zvolit a v krajním případě vytvoří zcela náhodný strom nezávisle na informaci od učitele. Podrobný popis algoritmu a především diskuze nad vhodností výchozího parametru pro randomizaci je nad rámec této práce a čtenář je najde v [11]. Tento přístup kromě přesnosti vyniká především efektivitou výpočtů.

3.3 Programové prostředky pro klasifikaci

Pro práci s klasifikátory využíváme knihovnu `scikit-learn`, která integruje prostředky pro strojové učení do jazyka Python s využitím dalších vědeckých modulů `numpy` a `scipy` [18].

Oba popsané typy klasifikátorů jsou ve `scikit-learn` implementovány a to nad knihovnou `LIBLINEAR` [10] v případě logistické regrese a na základě již zmiňované práce [11] pro extrémně znáhodněné stromy.

4. Hlasové dialogové systémy

Systémy umožňující hlasovou komunikaci člověka s počítačem nazýváme hlasovými dialogovými systémy (HDS). Kromě rozpoznání a syntézy řeči je zde zapotřebí také porozumění obsahu promluv uživatele. Navíc je třeba efektivně informace od uživatele zjišťovat a pokud možno i ověřovat. Toho se docílí vhodnou volbou řízení HDS.

4.1 Řízení dialogových systémů

V současné době se využívá dialogů v nichž se v tzv. výměnách střídá jako řečník uživatel a systém. Změna řečníka přichází po určité době ticha. Jako nový přístup se objevil tzv. inkrementální dialogový systém, který má přiblížit komunikaci přirozenému dialogu mezi lidmi. Tento přístup nerozlišuje výměny, resp. umožňuje systému okamžitě přerušit promluvu uživatele a dotázat se na doplňující informace, popř. provést určitou akci před skončením promluvy. [21] U obou přístupů je též možnost, aby uživatel přerušil promluvu systému.

Samotné řízení HDS můžeme rozdělit z pohledu jeho aktivity na tři kategorie: dialogy s iniciativou systému, dialogy s iniciativou uživatele a dialogy se smíšenou iniciativou.

První kategorie předpokládá, že systém pokládá přímé otázky uživateli a očekává od něj stručnou odpověď či povel na základě stanoveného plánu. U systémů s iniciativou uživatele jsou otázky a požadavky předkládány uživatelem a tedy bez předem známého plánu. Smíšenou iniciativou pak rozumíme situaci, kdy je systém schopen zpracovat delší odpovědi uživatele, který zasahuje do plánu dialogu tím, že některé otázky systému předjímá.

Z hlediska struktury systému a z ní plynoucího průběhu dialogu a uložení informací od uživatele rozlišujeme systémy s konečným počtem stavů, systémy se strukturou rámců a systémy s využitím agentů.

U systému s konečným počtem stavů definujeme mezi stavy přechody, které určují možné průchody dialogem. Jejich struktura je pevně zvolena při vytváření systému a posun z jednoho stavu do druhého není možný, dokud uživatel nepro-

vede k danému přechodu příslušející akci. Může jít o odpověď na otázku systému nebo nějaký povel, jako třeba „nápověda“. Tato struktura je vhodná u HDS s iniciativou systému. Uživatel je nucen chovat se podle plánu dialogu.

Struktura rámců využívá rámce, které obsahují tzv. sloty. Rozlišit pak můžeme sloty, které jsou potřebné pro splnění daného rámce a sloty, které obsahují jen doplňující informace neovlivňující splnění rámce. Úkolem systému je tyto sloty naplnit informací od uživatele. K naplnění slotů může dojít v libovolném pořadí a to i v jedné výměně. Struktura rámců je vhodná především u HDS s iniciativou smíšenou nebo s iniciativou uživatele. [20]

4.2 Porozumění

Pro přirozený dialog člověka s počítačem je třeba jistý způsobem z promluvy získat klíčové informace, které nám uživatel poskytuje. K tomu slouží právě moduly porozumění. [3]

V rámci porozumění můžeme definovat tzv. sémantické entity. Jedná se o pojmenování určitých konceptů (ať už jsou konkrétní jako např. vlakové stanice, požadavek zjištění odjezdu vlaků nebo i obecné např. číslice), jejichž nalezení v promluvě je pro řízení a průběh dialogu důležité. Každá takováto entita má svůj název (tzv. tag) a může být přiřazena jak k celým promluvám, tak i časově zarovnaná k jejich částem. U časově zarovnaných entit pak může řízení HDS brát v potaz např. jak jdou entity po sobě či jaký je mezi nimi časový rozdíl. V tomto případě můžeme hovořit i o sémantických mřížkách.

Jednou z možností detekce časově zarovnaných entit je vytvoření pravděpodobnostního modelu podobného jako je v rozpoznávání řeči model jazykový. Pokud hledáme nejlepší posloupnost $\tau(w_{1,n})$ tagů $t_{1,n}$ v promluvě z n slov $w_{1,n}$, pak podle [3] využijeme $\tau(w_{1_n}) = \arg\max_{t_{1,n}} P(t_{1,n}, w_{1,n})$.

Pravděpodobnosti výskytu konkrétní posloupnosti sémantických entit při dané promluvě jsou pak získány např. trénováním z dostatečně velkého korpusu, v kterém jsou entity označeny. Podobně jako u jazykových modelů i zde je možné redukovat historii předcházejících sémantických entit a slov. Zde však narážíme na problémy slov OOV, které nemáme ve slovníku (Out of Vocabulary) - ty nejsme

schopni rozpoznat.

K porozumění mluvené řeči můžeme přistoupit jako k problému detekce klíčových slov [23]. Tuto detekci můžeme provádět nad slovy, nicméně ke správnému rozpoznání slov je zapotřebí kvalitní jazykový model. Za použití fonémové mřížky můžeme vytvořit mřížku sémantickou. Připomeňme, že fonémový rozpoznávač vyžaduje pouze fonémový slovník, který je výrazně menší než lexikální slovník a je možný jeho zápis úplným výčtem fonémů. Odpadá tak také problém OOV slov.

Z takovýchto fonémových mřížek je pak možné najít nejlepší hypotézu. Problematika překryvů při zarovnání různých sémantických entit je řešena např. v [13].

5. Detekce klíčových frází (Spoken term detection)

Problematika porozumění řeči je velice blízká problematice detekce klíčových frází. V případě porozumění řeči je vstupem jediná promluva a nad ní se realizují komplexní dotazy, např. hledáme různé časy, položky z databáze. Oproti tomu máme při detekci klíčových frází rozsáhlou množinu vstupních promluv a nad ní realizujeme dotazy často tvořené jedním slovem nebo jednou frází.

Detekce klíčových frází pomocí vážených konečných transducerů umožňuje efektivně implementovat oba tyto přístupy. Soustřeďme se nyní na vývoj metody, která umožní v obou těchto úlohách efektivně kombinovat informaci na slovní a fonémové úrovni. V případě porozumění mluvené řeči je slovní úroveň reprezentována slovní gramatikou popisující sémantické entity. U detekce klíčových frází je naopak nutné efektivně určit, jaké segmenty vstupních promluv je vhodné indexovat na slovní úrovni a jaké na fonémové.

Proto byl navržen algoritmus, který umožňuje kombinovat rozpoznávání vstupních promluv na fonémové úrovni s dotazy na úrovni slovní.

Detekcí klíčových slov se můžeme zabývat nejen v oblasti hlasových dialogů, ale též v oblasti indexace a vyhledávání v řečových archivech. Na katedře kybernetiky Západočeské univerzity již byly implementovány některé metody indexace za pomoci vážených konečných transducerů a to jak nad slovní, tak fonémovou mřížkou v práci [22].

Dalším problémem detekce klíčových frází z pohledu zpracování sémantických mřížek vhodným řešením překryvů různých sémantických entit se zabývá práce [14].

6. Navržená metoda

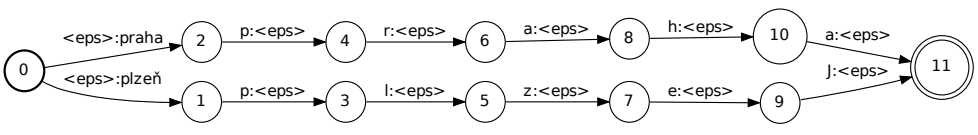
V této části popíšeme vstupní data, metodu jejich zpracování a nakonec také samotnou metodu detekce a formát výstupu.

6.1 Slovník fonetických přepisů

Pro potřeby kombinace fonémové mřížky a lexikálních realizací sémantických entit využíváme jako mezičlánek slovník fonetických přepisů. Slovník byl vygenerován prostřednictvím fonetických pravidel používaných na katedře kybernetiky. Obsahuje tabulátorem oddělenou dvojici hodnot a to přepisované slovo a mezerami oddělenou posloupnost fonémů, která je jeho přepisem. Například pro přepis slova Plzeň najdeme ve slovníku řádek

```
plzeň  p l z e J
```

V následujících odstavcích budeme slovník používat především při kompozici konečných transducerů, proto je třeba ho převést do tohoto formátu. Vhodnou ukázkou je opět slovo „Plzeň“ na obr. 6.1 v dvojici se slovem „Praha“. Vidíme, že přepisované slovo je výstupním symbolem jedné z hran, zatímco ostatní hrany mají jako výstupní symbol vždy symbol prázdného řetězce ϵ. Jako vstupní symboly hran jsou pak samotné fonémy, které při průchodu tímto automatem odpovídají fonetickému přepisu. Další přepisy slov se přidávají paralelně a vychází opět z počátečního stavu. V dalším textu budeme tento automat pro fonetický přepisů označovat jako transducer D.



Obrázek 6.1: Automat pro fonémový přepis slov „Plzeň“ a „Praha“

6.2 Referenční data

Data sloužící k vytvoření informací učitele při fázi trénování byla k dispozici ve formě slovního přepisu zvukových nahrávek provedeného anotátory s přiřazenou identifikací k příslušné fonémové mřížce a dalšími informacemi. Tento přepis je převeden pro každou mřížku do konečného automatu R_w, který přijímá právě jen posloupnost slov odpovídající přepisu. Využijeme též automatu pro fonetický přepis D podle části 6.1. Kompozicí slovníku a slovního přepisu získáváme mřížku $R_f = D \circ R_w$ se všemi fonetickými přepisy referenční promluvy. Tuto mřížku využijeme pro pozdější generování příznaků pro klasifikátor.

```
00012; present_info; TO(STATION); do plešnic; do c_station2;
CD03~Part1~04-000506-145926-1_006
```

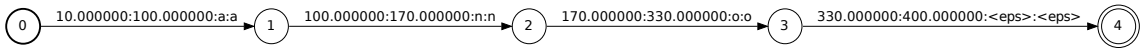
6.3 Nejlepší časově zarovnané hypotézy z fonémové mřížky

Jedním z výstupů fonémového rozpoznávače je první nejlepší (1-best) hypotéza jako posloupnost fonémů v souborech formátu MLF spolu s časovým zarovnáním (čas začátku a konce daného fonému v řečovém signálu), která odpovídá akustickému signálu s největší pravděpodobností. Z této posloupností je pro každou hypotézu sestaven konečný transducer L_{best}. Do vstupních symbolů na hranách transduceru je zakódováno jak časové zarovnání s časem začátku a konce, tak i samotný znak fonému. To je provedeno převedením obou časů na řetězec a spojení těchto řetězců do jednoho přes znak, který v původní abecedě není - v našem případě přes dvojtečku. Výstupním symbolem je pak samotný foném. Průchodem tímto transducerem získáme z výstupních symbolů právě původní posloupnost fonémů.

Ukázka transduceru vytvořeného z MLF s nejlepší hypotézou je na obr. 6.2.

6.4 Zpracování fonémové mřížky

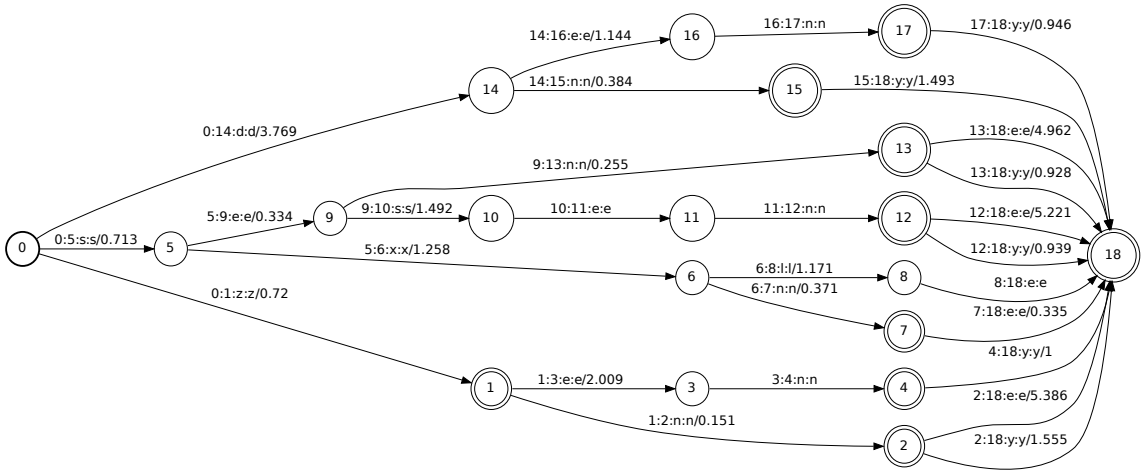
Fonémová mřížka je dalším, informačně bohatším, výstupem fonémového rozpoznávače. Obsahuje více nejlepších hypotéz (počet závisí na prořezávání u



Obrázek 6.2: Automat vytvořený z MLF souboru s časovým zarovnáním zakódovaným na vstupní symboly hran

fonémového rozpoznávače). K dispozici byly mřížky již ve formátu váženého konečného akceptoru. V tomto akceptoru jsou na hranách symboly fonémů ohodnocené pravděpodobností, s kterou rozpoznávač odhaduje výskyt daného fonému v řečovém signálu. Ve fonémové mřížce chybí informace o zarovnání, které je však vyžadováno pro další zpracování. Můžeme však využít topologického seřazení konečného akceptoru (popsáno v části 2.4) k zarovnání jednotlivých fonémů - toto zarovnání nicméně neodpovídá zarovnání časovému. Samotná časová informace však není pro zpracování důležitá, neboť nám postačují údaje o relativním uspořádání sémantických entit, které získáme i touto aproximaci.

Obdobně jako u nejlepší hypotézy i zde vytvoříme konečný transducer L, v kterém zakódujeme informaci o aproximovaném počátečním a koncovém čase fonému do vstupních symbolů hran. Výstupní symboly opět obsahují pouze příslušný foném. Ukázka takto upravené fonémové mřížky ve formě transduceru je na obr. 6.3.

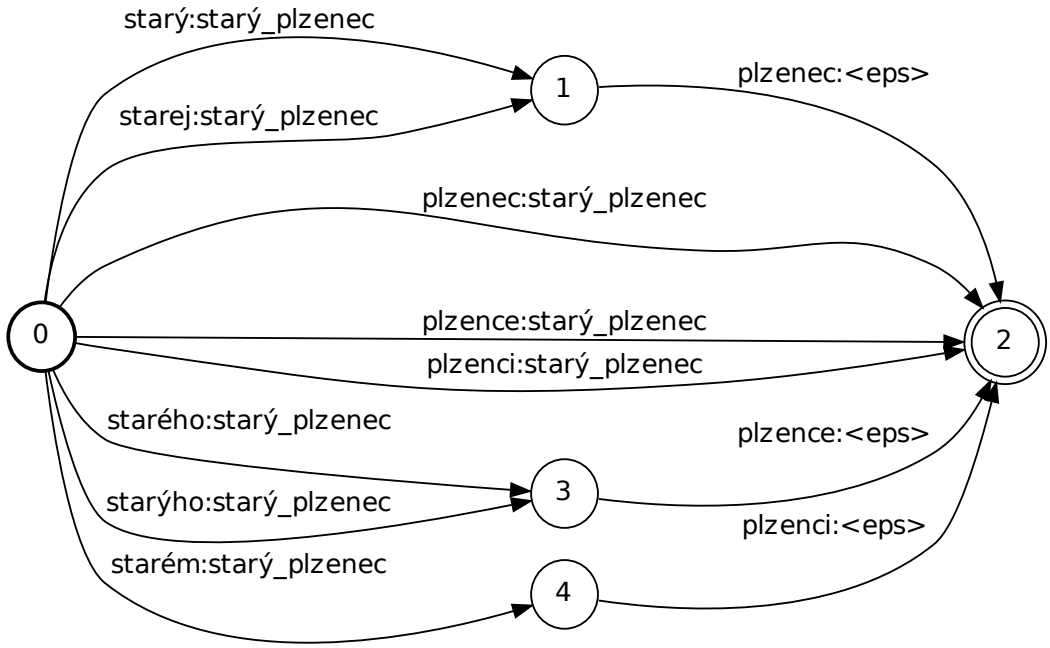


Obrázek 6.3: Mřížka s časovým zarovnáním zakódovaným na vstupních symbolech hran

6.5 Zpracování sémantických entit

Hledání uskutečňujeme na základě sémantický entit. Tyto entity jsou definovány svými lexikálními realizacemi a to buď prostým výčtem dvojic „entita-realizace“ či gramatikou. Bylo třeba vytvořit takový výstup, který by umožnil určovat sémantické entity kompozicí s fonémovou mřížkou. K sestavení výstupu využijeme opět konečných automatů a to konkrétně kompozici několika „vrstev“.

V prvé řadě je třeba uchovat informaci k jaké entitě daná lexikální realizace patří. Toho docílíme vytvořením transduceru S_i (na obr. 6.4 vidíme příklad sjednocení několika takovýchto transducerů), kde je název sémantické entity jako výstupní symbol hrany transduceru a k ní příslušná realizace je vstupním symbolem této hrany. Pokud je lexikální realizace víceslovná, jsou slova umisťována na po sobě jdoucí hrany, přičemž příslušná sémantická entita je vstupním symbolem pouze jedné z této hran a ostatní hrany mají jako vstupní symbol prázdný řetězec (ϵ).



Obrázek 6.4: Reprezentace hledané sémantické entity `starý_plzenec` s lexikální realizací ve formě konečného transduceru

Sjednocením $S = \bigcup_{i=1,\ldots,N} S_i$ dostáváme požadovaný transducer S lexikálních

realizací sémantických entit.

Další možností načtení sémantických entit a jejich realizací jsou gramatiky převedené do formy konečného transduceru (použitím nástrojů z části 2.1).

Čtenář snadno nahlédne, že pro kombinaci s fonémovou mřížkou jako je ta na obr. 6.3 je tento automat nevhodný, neboť by kompozicí vznikl prázdný automat. Je třeba ještě další vrstvy, a to již popsaného konečného transduceru D, který zajišťuje fonetický přepis realizací sémantických entit - vytváří tedy fonetické realizace. Kompozicí automatů $T = D \circ S$ pak získáváme všechny přípustné posloupnosti fonémů a k nim příslušnou entitu.

6.6 Přístup přesné shody

Po vytvoření automatů L a T můžeme přistoupit k samotné detekci klíčových slov prostou kompozicí $R = F(L) \circ T$. Konečný automat R pak obsahuje všechny posloupnosti fonémů, které přesně odpovídají hledaným entitám.

Jedná se o přístup, který vyžaduje, aby výstup rozpoznávače obsahoval bezchybný fonémový přepis hledaných slov. Z výsledků experimentů bude patrné, že fonémový rozpoznávač neposkytuje dostatečně kvalitní výstup a bylo třeba využít jiného přístupu, který by dokázal vzít v potaz i možné chyby fonémového rozpoznávače.

6.7 Využití editační vzdálenosti

Editační vzdálenost

Pro dva řetězce můžeme definovat editační vzdálenost jako nejmenší počet úprav nad znaky těchto řetězců jejich záměnou, vložením či smazáním. V této práci využijeme Levenshteinovu vzdálenost. Formálně můžeme podle [24] definovat tuto

vzdálenost $\mathrm{lev}_{a,b}(|a|,|b|)$ mezi slovy a, b jako

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & , \text{ pokud } \min(i,j) = 0 \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j)+1 \\ \mathrm{lev}_{a,b}(i,j-1)+1 \\ \mathrm{lev}_{a,b}(i-1,j-1)+[a_i \neq b_j] \end{cases} & , \text{ jinak} \end{cases}$$

Konečný automat pro výpočet Levenschteinovy vzdálenosti

Pro nalezení této vzdálenosti můžeme využít i konečných automatů. Základem je automat $E = (\Sigma, \Sigma, \{s\}, T, s, \{s\}, 0, 0)$ s jediným stavem s nad tropickým polookruhem. [2]

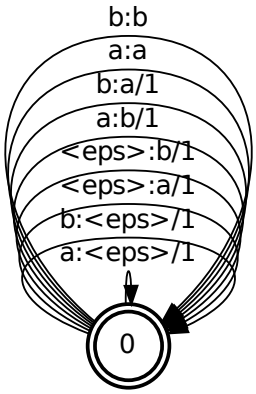
Na jeho hranách pak stanovíme penalizace jednotlivých operací:

- vložení jako hrany $(s, \epsilon, \alpha, 1, s)$, tedy přepis prázdného řetězce na symbol $\alpha \in \Sigma$ s váhou 1.
- smazání jako přechody $(s, \alpha, \epsilon, 1, s)$
- substituce jako přechody $(s, \alpha, \beta, 1, s)$, kde $\alpha, \beta \in \Sigma$, přičemž $\beta \neq \alpha$.
- ponechání fonému jako $(s, \alpha, \alpha, 0, s)$

Obecně pro editační vzdálenost mezi dvěma transducery $A_1 = (\Sigma_1, \Omega_1, Q_1, T_1, i_1, F_1, \lambda_1, \rho_1)$ a $A_2 = (\Sigma_2, \Omega_2, Q_2, T_2, i_2, F_2, \lambda_2, \rho_2)$ musíme zaručit, že množina symbolů $\Sigma = \Sigma_1 \cup \Sigma_2$. Pro porovnání na úrovni fonémové mřížky pak postačí vyjít z informace, že $\Sigma = \Sigma_1 = \Sigma_2 = P$, kde P je fonémová abeceda zmíněná v 4.2.

Příklad automatu pro určení editační vzdálenosti nad abecedou $P_{ab} = \{a, b\}$ je vidět na obr. 6.5.

Kompozicí $R = L \circ E \circ T$ můžeme získat vzdálenost fonémové mřížky L od fonetického realizace hledaných entit T. Příklad automatu $R_{ab} = Ł_{ab} \circ E \circ T$, kde automat L_{ab} představuje jednoduchou mřížku nad abecedou $P_{ab} = \{a, b\}$ přijímající řetězce z množiny $\{abb, aab\}$ a automat T_{ab} přijímá pouze řetězec bbb je na obr. 6.6. Nejkratší cestou je pak $abb \rightarrow bbb$ s váhou 1.



Obrázek 6.5: Automat umožňující stanovit editační vzdálenost mezi dvěma automaty nad abecedou P_{ab}

Je třeba si uvědomit, že výsledný automat R obsahuje všechny možné vzdálenosti, nikoliv jen tu nejmenší. Tu získáme hledáním nejkratší cesty tímto automatem podle 2.4.

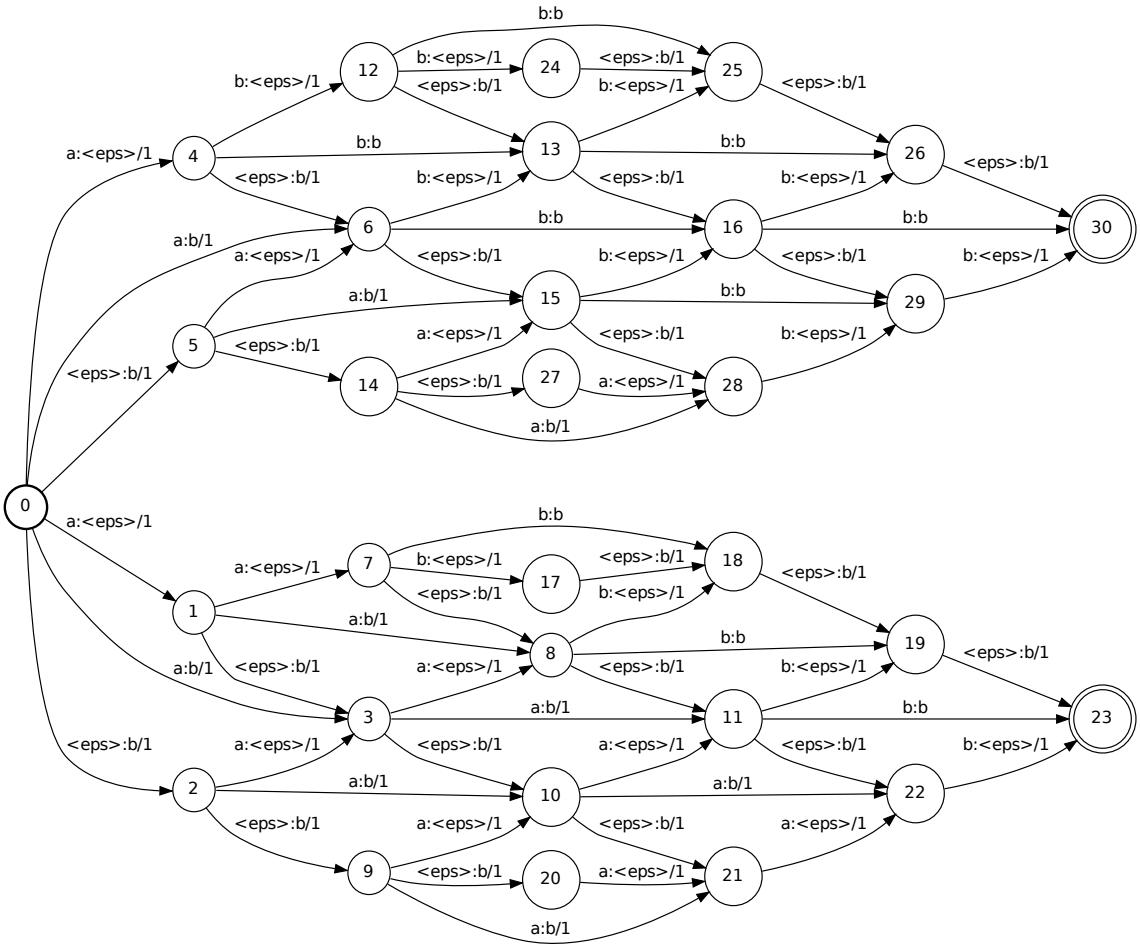
V [2] je taktéž zmíněno, že automat obsahuje $(|P| + 1)^2 - 1$ hran, přičemž rozdělením automatu určitým způsobem na dva jednostavové automaty $E_1(\Sigma_1, \Sigma_1, \{s_1\}, T_1, s_1, \{s_1\}, 0, 0)$ a $E_2 = (\Sigma_2, \Sigma_2, \{s_2\}, T_2, s_2, \{s_2\}, 0, 0)$ nad tropickým polookruhem je možné docílit počtu hran $4|P|$. Kompozice těchto dvou automatů je ekvivalentní automatu E. První komponentou E_1 je jednostavový automat složený z hran využívajícího následujícího principu:

- vložení jako přechod $(s_1, \epsilon, \texttt{<ins>}, 0.5, s_1)$, tedy přepis prázdného řetězce na symbol `<ins>` s váhou 0.5
- smazání jako přechody $(s_1, \alpha, \texttt{<del>}, 0.5, s_1)$
- substituce jako přechody $(s_1, \alpha, \texttt{<del>}, 0.5, s_1)$
- ponechání fonému jako $(s_1, \alpha, \alpha, 0, s_1), \forall \alpha \in \Sigma_1$

přičemž `<sub>`,`<del>`,`<ins>` jsou speciální symboly přidané k tabulce symbolů.

Druhý jednostavový automat E_2 pak využívá hrany vytvořené podle principu

- vložení jako přechody $(s_2, \texttt{<ins>}, \beta, 0.5, s_2)$, tedy přepis prázdného řetězce na řetězec $\beta \in \Sigma_2$ s váhou 0.5.
- smazání jako přechod $(s_2, \texttt{<del>}, \epsilon, 0.5, s_2)$



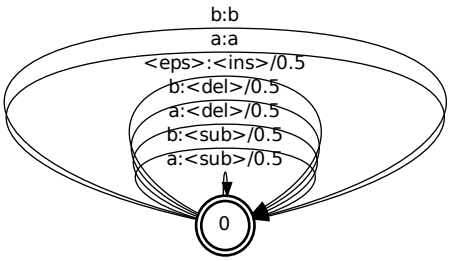
Obrázek 6.6: Příklad automatu, u nějž váha nejlepší cesty určuje editační vzdálenost řetězců „aab“, „abb“ od řetězce „bbb“

- substituce jako přechod $(s_2, \texttt{<sub>}, \beta, 0.5, s_2)$
- ponechání fonému jako $(s_2, \beta, \beta, 0, s_2), \forall \beta \in \Sigma_2$

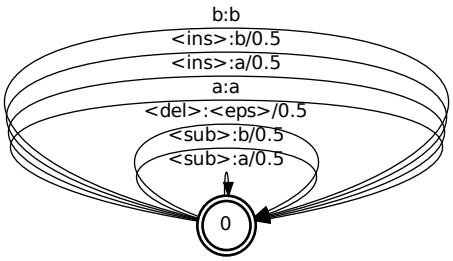
Pokud se i zde omezíme na fonetickou abecedu P a tedy $\Sigma_1 = \Sigma_2 = P \cup \{\texttt{<ins>}, \texttt{<sub>}, \texttt{<del>}\}$, můžeme kompozicí získat vzdálenost fonémové mřížky A od hledaného fonetického přepisu entit B jako $R_2 = A \circ E_1 \circ E_2 \circ B$.

Příklad automatů E_1 a E_2 nad abecedou $P_{ab} = \{a, b\}$ je na obr. 6.7a a 6.7b. Výsledek kompozice $R_{ab} = E_1 \circ E_2$ je na obr. 6.7c.

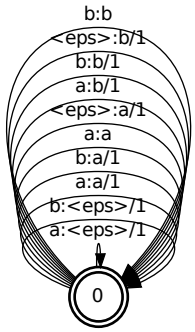
I přes redukci počtu hran však může neomezený počet vložení, smazání a nahrazení způsobit výrazné zpomalení výpočtů. Je vhodné přistoupit k omezení



(a) Konečný automat E_1



(b) Konečný automat E_2



(c) Kompozice $E_1 \circ E_2$. Zde zjevně ekvivalentní s automatem na obr. 6.5

Obrázek 6.7: Příklad dvojice transducerů pro určení editační vzdálenosti řetězců dvou automatů nad abecedou P_{ab}

počtu možných vložení a smazání.

Omezení maximálního počtu operací

Uvažujme, že chceme povolit maximálně n smazání či inzercí. Byly implementovány dva postupy, jak tohoto omezení docílit a to buď omezením počtu „s možností opakování“ a nebo „bez opakování“. Možností opakování rozumíme případ, kdy po maximálně n smazáních či vložení musí dojít k jedné ze zbývajících dvou operací (substituce nebo ponechání znaku), aby bylo opět možné použít dalších maximálně n mazání nebo vložení znaků. Pokud opakování nepovolíme, dojde k maximálně n smazání či vložení pouze v jedné souvislé části řetězce. Diskuze vlivu parametru n na výsledky detekce klíčových slov je v části 7.

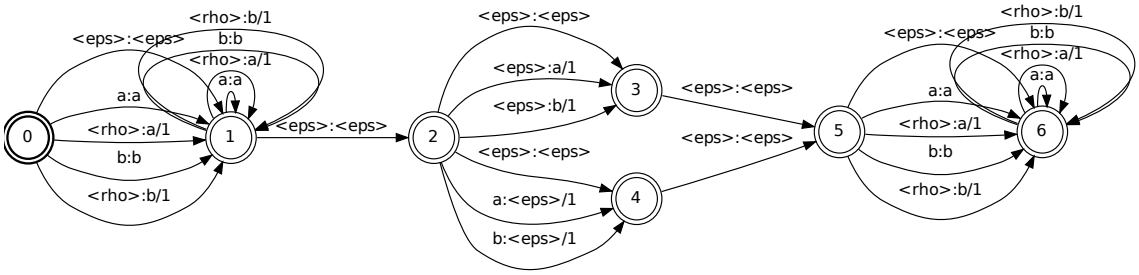
Opět můžeme rozdělit automat pro editační vzdálenost na dvě komponenty. První z nich je konečný transducer $E_{di} = (P, P, \{s_1, \ldots, s_{n+1}\}, T, s_0, \{s_{n+1}\}, 0, 0)$ zajišťující až n smazání či vložení. Tento automat má $n + 1$ stavů a mezi dvěma stavy $s_i, s_{i+1}, i = 1, \ldots, n$ je vždy $2|P|$ přechodů $t_i \in T$, z nichž je $|P|$ přechodů typu $(s_i, \epsilon, a, 1, s_{i+1}), \forall a \in P$ (vložení) a $|P|$ přechodů $(s_i, a, \epsilon, 1, s_{i+1}), \forall a \in P$ (smazání). Po sestavení těchto hran je automat faktorizován.

Druhou komponentou je nad tropickým polookruhem vytvořený dvoustavový automat $E_s = (P_\rho, P_\rho, \{s_0, s_1\}, T, s_0, \{s_0, s_1\}, 0, 0)$, kde $P_\rho = P \cup \{\rho\}$. Mezi stavy s_0 a s_1 jsou hrany $t_i \in T$ ve tvaru $(s_0, a, \rho, 1, s_1), \forall a \in P$ (záměna znaků) a $(s_0, a, a, 1, s_1), \forall a \in P$ (shoda znaků). Připomeňme, že ρ je symbol zastupující libovolný symbol který není na žádné hraně vycházející ze stejného stavu s_0 (popsán v části 2.4). Stejné operace jsou pak ve smyčce na stavu s_1, tedy $(s_1, a, \rho, 1, s_1), \forall a \in P$ a $(s_1, a, a, 1, s_1), \forall a \in P$. V případě, že nepožadujeme možnost opakování vložení či smazání, přidáváme i hranu $(s_0, \epsilon, \epsilon, 0, s_1)$, aby bylo možné provést tyto dvě operace i na prvním a posledním znaku porovnávaných řetězců.

Automat L pro určení Levenshteinovy vzdálenosti s omezeným počtem operací smazání či vložení bez opakování pak vytvoříme konkatenací $L = E_s + E_{di} + E_s$ a všechny stavy budeme považovat za koncové. Příkladem je automat sestavený pro symboly z P_{ab} a $n = 1$ na obr. 6.8.

Pro možnost opakování provedeme konkatenaci $L = E_{di} + E_s$ s tím, že přidáme

hranu se symbolem prázdného řetězce ϵ mezi jeho poslední stav a počáteční stav.



Obrázek 6.8: Automat vhodný pro určení editační vzdálenosti mezi dvěma automaty nad abecedou P_{ab} s omezením na 1 smazání či vložení

Na rozdíl od předchozích automatů pro určení Levenshteinovy vzdálenosti, kde stačila běžná kompozice tří konečných automatů, je u tohoto typu potřeba použít ρ-kompozici (viz 2.4).

6.8 Využití klasifikátorů

V algoritmu detekce jsou též použity klasifikátory. Zde popíšeme jaké příznaky pro klasifikaci využíváme, jak probíhá trénování a jak je využita klasifikace.

Při implementaci bylo rozděleno trénování klasifikátorů na dvě části - přípravu dat, jejímž výstupem jsou vektory příznaků a informace učitele, a pak samotné vytvoření a trénování klasifikátoru.

Zvolené příznaky

Vektor příznaků má dimenzi 91 a vychází z informací o hledaných sémantických entitách a pozorovaných posloupností fonémů. První tři příznaky jsou celočíselné a obsahují délku pozorovaného řetězce, editační vzdálenost od fonémové realizace hledané entity a také počet fonémů shodných s touto realizací. Dále obsahuje informaci o začátku a konci řetězce (zarovnání v podobě dvou desetinných čísel). Následují binární příznaky, které postupně pro všechny symboly fonetické abecedy určují, zda se daný foném v pozorovaném řetězci vyskytuje (1) či nikoliv (0). Po nich přidáváme do vektoru příznaků opět postupně pro všechny fonémy z

abecedy celočíselnou informaci o tom, zda byl foném obsažen jak v pozorovaném řetězci, tak ve fonetické realizaci sémantické entity (1), zda byl nahrazen, smazán, či vložen (-1) nebo zda nebyl ani v jednom řetězci (0).

Příprava dat

Příprava dat předpokládá existenci trénovací sady mřížek, k nim příslušný seznam hledaných entit, jejich lexikální realizace a slovník fonetických přepisů.

Načteme seznam hledaných entit a vytvoříme z nich konečný automat S postupem popsaným v 6.5. Převodem fonémového slovníku podle 6.1 vytvoříme konečný transducer D a získáváme fonetickou realizaci hledaných entit Q kompozicí $Q = D \circ S$. Tento automat dále upravíme a to přidáním automatu pro zjištění editační vzdálenosti kompozicí $Q_E = E \circ Q$ s automatem E z části 6.7.

Pro každou mřížku z trénovací sady pak provádíme následující:

- Načteme mřížku L_M, příp. první nejlepší hypotézu podle 6.4 resp. 6.3
- Z mřížky L_M sestavíme faktorový automat $L = F(L_M)$
- Provedeme ρ-kompozici $R = L \circ Q_E$. Takto vzniklý automat můžeme prořezat libovolným prahem - vhodná volba prahu je diskutována v kapitole 7.
- Postupným procházením automatu R získáváme údaje potřebné pro vytvoření vektoru příznaků pro klasifikátor.

  Ze vstupních symbolů procházených hran dekódujeme časy začátku a konce fonémů a také samotné fonémy. Z vah pak určíme editační vzdálenost od fonetických realizací hledaných sémantických entit.

  Na základě stanovených hranic minimální délky a minimální relativní shody rozhodujeme o tom, zda bude posloupnost nalezených fonémů brána jako pozitivní příklad (v případě dodržení minimální délky a minimální relativní shody) či jako příklad negativní.
- Vygenerovaný vektor příznaků spolu s informací učitele ukládáme do souboru pro pozdější použití při trénování

Trénování

Pro trénování klasifikátoru předpokládáme pouze existenci souboru příznaků a informací učitele vytvořenou během přípravy dat. Po načtení tohoto souboru může být provedeno decimování - omezení počtu negativní příkladů pro trénování tím, že jich z trénovací sady náhodně vyřadíme $D\%$. V části experimentů bude ukázáno, jaký vliv má míra decimování D na přesnost detekce, resp. výstupy klasifikátoru.

Samotné trénování probíhá s využitím knihovny scikit-lean [18]. Použity byly klasifikátory popsané v kapitole 3. V kapitole 7 jsou diskutovány výsledky pro jednotlivé klasifikátory a jejich parametry.

Načtení velkého množství příznaků je náročné na paměť a proto byly především pro trénování využity clustery spadající pod MetaCentrum. Výstupem této části je natrénovaný klasifikátor.

6.9 Detekce

Vstupem pro algoritmus detekce je seznam hledaných sémantických entit s jejich lexikálními reprezentacemi, seznam mřížek nad kterými provádíme detekci a slovník fonetických přepisů. Jeho výstupem je informace o sémantických entitách nalezených v promluvě. Kromě samotné entity je na výstupu také časové zarovnání a odhad posteriorní pravděpodobnosti správného zařazení této entity. V následujících odstavcích bude popsáno, jak k tomuto výstupu dospět.

Nejprve načteme data, která se po dobu detekce měnit nebudou:

- Ze slovníku fonetických přepisů vytvoříme konečný transducer D pro fonetickou transkripci slov postupem popsaným v části 6.1.
- Načteme do podoby konečného transduceru Q_W hledané sémantické entity a jejich lexikální reprezentaci podle 6.5.
- Získáme fonetickou reprezentaci sémantických entit v podobě automatu $Q = D \circ Q_W$.
- Nakonec vytvoříme automat Q_E kompozicí $Q_E = E \circ Q$, kde E je automat popsaný v 6.7 umožnující určení editační vzdálenosti.

Dodejme, že se hledané sémantické entity použité při detekci obecně mohou lišit od těch, které byly použity pro trénování. V části 7 je diskutován vliv těchto změn na výsledky detekce.

Postupně procházíme seznam mřížek nad kterými požadujeme detekci a každou mřížku převedeme podle 6.3 (pokud jde o mřížku obsahující pouze první nejlepší hypotézu) či 6.4 (pro úplnou mřížku) na konečný automat L. Kompozicí s automatem Q_E získáváme kombinaci výstupů jak fonetického rozpoznávače, tak sémantického stromu a to spolu se zarovnáním a editační vzdáleností zakódovanou v transduceru $R = L \circ Q_E$. Tento transducer má na vstupních symbolech hran zakódovány fonémy s jejich časy začátku a konce v promluvě. Výstupní symboly pak obsahují informaci o sémantické entitě.

Postupným procházením všech přípustných cest automatu R získáváme z vstupních symbolů příznaky pro klasifikátor popsané v 6.8. Vektor těchto příznaků předáme klasifikátoru, který vrací informaci o tom, zda posloupnost fonémů na této cestě patří do skupiny hledaných sémantických entit a s jakou pravděpodobností.

Ze struktury konečných automatů, které sloužily jako součást kompozic, je dané, že výstupem jakékoliv přípustné cesty v automatu bude vždy jen jedna sémantická entita. Pokud klasifikátor určí, že posloupnost fonémů reprezentuje hledané slovo, pak výstupem je právě sémantická entita a příslušná pravděpodobnost z klasifikátoru.

Překryvy časů

Protože však automat R obecně není deterministický, tak může nastat situace, kdy se na výstupu objeví stejná sémantická entita ať už s různou pravděpodobností určenou klasifikátorem nebo s různými časy začátku a konce.

Označme tedy jednu sémantickou entitu jako s_1 s pravděpodobností p_1, její čas začátku t_1 a čas konce t_2. Dále uvažujme entitu s_2 s pravděpodobností p_2 časem začátku $t_1^{'}$ a časem konce $t_2^{'}$, přičemž $p_1 \leq p_1^{'} \wedge p_2 \geq p_1^{'}$. Pokud $s_1 \neq s_2$, pak obě entity zapíšeme na výstup, neboť je nežádoucí přijít o jednu z možných správně nalezených entit. V případě $s_1 = s_2$ pak na výstup vypíšeme s_1, pokud

$p_1 \geq p_2$, jinak s_2. Řešením je tedy vybrat takovou sémantickou entitu ze všech shodných časově se překrývajících entit, která má největší pravděpodobnost. Pokud je shodných překrývajících se entit více, pak tuto redukci provádíme dokud nejsou všechny shodné překrývající se entity odstraněny.

6.10 Formát výstupu

Výstupem je mřížka v HTK formátu popsaném v 2.6 s časově zarovnanými sémantickými entitami spolu s odhadem pravděpodobnosti z klasifikátoru.

```
#!MLF!#
CD03~Part1~04-000506-143418-1_001
10000 70000 desná:desný 0.007771 0.007771 1
.
CD03~Part1~04-000506-143554-1_001
60000 90000 žatec:žatce 0.033144 0.033144 1
60000 90000 tatce:tatce 0.036276 0.036276 1
60000 90000 stará_paka:pace 0.032850 0.032850 1
60000 90000 nová_paka:pace 0.032850 0.032850 1
.
```

Pro vyhodnocení experimentů je k dispozici i slovník mřížek uložený ve formátu čitelném pro Python s použitím modulu `pprint`. Tento slovník neobsahuje časové zarovnání, pouze sémantické entity příslušející k mřížce a jejich odhadované pravděpodobnosti. Výstup pro vyhledávání ve dvou mřížkách pak vypadá následovně:

```
{u'CD03~Part1~04-000506-143418-1_001':
    {u'station:desn\xe1': 0.0077711546569219038},
 u'CD03~Part1~04-000506-143554-1_001':
    {u'station:star\xe1_paka,nov\xe1_paka': 0.032850238299504286,
     u'station:tatce': 0.036275715356831849,
     u'station:\u017eatec': 0.033144073852512647}
}
```

7. Experimenty

Implementovaný algoritmus představený v části 6.9 vyžaduje určení několika parametrů, jejichž hodnota se výrazně podílí na přesnosti detekce sémantických entit. Kromě jejich experimentálního určení je také vhodné diskutovat jaké je přijatelné množství dat pro trénování klasifikátoru či zda je nutné využívat mřížky, nebo je dostačující použít první nejlepší hypotézu.

Algoritmus byl implementován v jazyce Python s využitím knihovny pyfst a OpenFst pro práci s konečnými automaty a scikit-learn pro práci s klasifikátory.

7.1 Metodika vyhodnocení (ROC, DR, ...)

Pro stanovení metodiky vyhodnocení nejprve definujeme ROC[1] křivky, detekční schopnosti DR[2] a míry falešných poplachů FPR[3]. [5]

V tabulce 7.1 jsou definována čísla, které reprezentují možnosti výsledků klasifikace. Pod pojmem „pozitivní pozorování“ rozumějme, že se hledaná entita v promluvě nachází. Pojem „pozitivní predikce“ pak značí, že je výstupem klasifikátoru informace, že se hledaná entita v promluvě nachází. Jejich opakem je pak pozorování či predikce „negativní“.

	Pozitivní pozorování p	Negativní pozorování n
Pozitivní predikce p'	Správný výsledek tp	Falešný poplach fp
Negativní predikce n'	Chybějící výsledek fn	Správný výsledek tn

Tabulka 7.1: Možné výsledky klasifikace a jejich značení

Analýzou výsledků a přiřazením hodnot podle tabulky můžeme vygenerovat některý z ukazatelů kvality klasifikace:

- míra detekce: DR $= \frac{tp}{p}$
- míra falešných poplachů FPR $= \frac{fp}{n}$

[1] angl.: Receiver Operating Characteristic Curve
[2] angl.: Detection Rate
[3] angl.: False Positive Rate

Připomeňme, že binární klasifikaci můžeme získat pro příznakový vektor x taktéž porovnáním skóre $d(x)$ s určitým prahem ϕ. Změnami prahu získáváme pracovní body, v kterých určujeme míry DR a FPR. Křivku vyjadřující závislost DR na FPR pak označíme jako ROC křivku.

Protože však počet počet negativních pozorování může být nekonečný (počet sémantických entit nemusí být omezen), zavedeme normalizovanou četnost falešných poplachů $FPR_N = \frac{FP}{N}$, kde N je celkový počet jednotek, na které falešné poplachy vztahujeme (v případě této práce se jedná o promluvy). Použitím veličiny FPR_N získáváme modifikovanou ROC křivku vyjadřující závislost DR na FPR_N při spojitě se měnícím prahu ϕ.

Pro porovnání ROC křivek využijeme velikost plochy pod křivkou

$$AUC = \int_0^1 DR(FPR_N)\mathrm{d}FPR_N$$

7.2 Použitá data

Experimenty byly provedeny s využitím dat z nádražní infolinky s odjezdy a příjezdy vlaků [15]. Jednalo se o promluvy uživatele i operátora s tím, že k dispozici byly fonémové mřížky spolu se slovními přepisy vytvořenými anotátory.

Z těchto dat byly vytvořeny dvě trénovací sady - menší sada, kterou budeme značit `train_h`, obsahovala 570 mřížek, ve větší sadě `train_t` bylo 5240 mřížek. Testovací sada nad kterou proběhlo vyhodnocení obsahovala 1439 mřížek.

Kromě mřížek byl použit seznam 2806 nádražních stanic v 1., 2., 4. a 6. pádě. Názvy stanic byly určeny jako sémantické entity s tím, že jejich lexikální realizace byly právě všechny vyskloňované varianty jejich skutečného názvu - celkem 21539 dvojic „entita-realizace“.

U některých stanic byly kromě pádů i jejich zkrácené formy, např. sémantická entita `Plzeň hlavní nádraží` měla kromě realizací `Plzeň hlavní nádraží`, `Plzně hlavního nádraží`, `Plzni hlavní nádraží` též realizace `Plzeň`, `Plzně`, `Plzni`.

Podobně pak byly určeny sémantické entity 9 typů vlaků spolu s jejich lexikálními realizacemi. Využit byl také slovník fonetických přepisů.

Jako základní výsledek detekce (tzv. baseline), podle kterého je možné dis-

kutovat případné zlepšení v experimentech, nám slouží výsledek detekce, která byla provedená na základě přesné shody fonetické realizace sémantických s posloupností fonémů ve fonémové mřížce (viz 6.6). Tento typ detekce dosáhl ve sledovaném parametru AUC hodnoty 0.458 pro nádražní stanice a AUC= 0.601 pro typy vlaků.

7.3 Hledání optimálních parametrů

Pro nalezení optimálních parametrů ze spojitého prostoru bylo použito vyčerpávající hledání „grid search“ na vhodně zvolených konečných množinách hodnot, které jednotlivé parametry mohou nabývat.

Hledané parametry jsou:

- $P \in \{1, 2, 3\}$ - práh prořezávání automatů v průběhu hledání i trénování
- $C \in \{1, 2, 3, 4\}$ - maximální povolený počet vložení či smazání při určování editační vzdálenosti mezi fonémovou mřížkou a fonémovými realizacemi entit
- $D \in \{0.1, 0.5, 0.7, 0.9\}$ - míra decimování negativních příkladů při trénování
- $M \in \{0.3, 0.5, 0.7, 0.9\}$ - minimální relativní shoda fonémů nalezeného řetězce s hledanou fonetickou realizací sémantické entity
- $L \in \{1, 2, 3, 4\}$ - minimální počet fonémů nalezeného řetězce
- $R \in \{0, 1\}$ - příznak, zda připouštíme opakování vložení a smazání při určování editační vzdáleností (1) či nikoliv (0)
- $T \in \{0, 1\}$ - příznak, zda jako vstupní data požadujeme fonémovou mřížku (1) nebo pouze k ní příslušnou první nejlepší hypotézu (0)

Pro stanovení těchto parametrů byly hledány sémantické entity nádražních stanic nad množinou testovacích dat. V případě budoucích experimentů by bylo vhodné také vyzkoušet hledání optimálních parametrů na jiných, neviděných datech, aby bylo zajištěno dostatečné zobecnění parametrů.

Logistická regrese

Optimální kombinaci parametrů při klasifikaci s využitím logistické regrese (LR) představuje $P = 1, M = 0.7, C = 1, L = 4, D = 0.9, R = 1, T = 1$.

V tabulce 7.2 vidíme jakého rozdílu ve sledovaném parametru AUC je dosaženo při použití fonémové mřížky oproti prosté nejlepší hypotéze.

Rozhodovací stromy

Pro tento typ klasifikátoru byla nalezena následující kombinace parametrů: $P = 2, M = 0.7, C = 2, L = 4, D = 0.1, R = 0, T = 1$.

Jakého rozdílu ve sledovaném parametru AUC je dosaženo při použití fonémové mřížky oproti prosté nejlepší hypotéze z těchto mřížek (tedy změna příznaků T), při jinak optimálních parametrech, vidíme v tabulce 7.2. V tabulkách používáme jeho anglické označení Extra-Tree.

7.4 Kombinace klasifikátorů a přístupu přesné shody

V průběhu experimentů bylo zjištěno, že pro některé množiny trénovacích dat dosahujeme výsledků horších, než je baseline. Přistoupili jsme tedy k vytvoření dalšího experimentu.

V tomto experimentu nejdříve hledáme přesnou shodu podle 6.6, a nalezeným entitám přiřadíme posteriorní pravděpodobnost 1. Tímto způsobem je tvořena baseline, my však k těmto entitám navíc přidáme entity určené klasifikátorem.

Tato kombinace byla provedena s parametry, které byly dříve určeny jako optimální u daných klasifikátorů. V budoucnu by bylo vhodné provést navíc i analýzu, zda byly parametry nalezené pro samotné klasifikátory optimální i pro zde navrženou kombinaci.

Výsledky tohoto experimentu budeme označovat jako „LR+přesná shoda“ pro logistickou regresi a „ET+přesná shoda“ pro extrémně znáhodněné stromy (Extra-Tree).

V tabulce 7.3 pak porovnáním s předchozími přístupy vidíme, že pro malé

množiny trénovacích dat (popsány dále) došlo ke zlepšení, nicméně větší množiny zaznamenaly pokles hodnoty AUC. To může být způsobeno sémantickými entitami, které mají svůj fonémový přepis shodný s některými běžnými slovy, kterým pak samotný klasifikátor běžně přiřazoval nižší odhad posteriorní pravděpodobnosti a tím dosáhl lepších výsledků.

Porovnání ROC křivek samotných klasifikátorů a jejich kombinací s přístupem přesné shody je na obr. 7.3.

Klasifikátor	Trénovací data	ST	30ST	200ST	500ST	1000ST
Logistická regrese	mřížky	0.713	0.683	0.704	0.662	0.700
	nejlepší hypotézy	0.613	0.625	0.615	0.615	0.618
Extra-Tree	mřížky	0.732	0.549	0.628	0.427	0.625
	nejlepší hypotézy	0.655	0.571	0.605	0.337	0.588

Klasifikátor	Trénovací data	TT	TTX
Logistická regrese	mřížky	0.869	0.869
	nejlepší hypotézy	0.803	0.803
Extra-Tree	mřížky	0.869	0.867
	nejlepší hypotézy	0.804	0.802

Tabulka 7.2: Hodnoty AUC pro různé velikosti množiny trénovacích sémantických entit ukazující rozdíl mezi použitím mřížek a první nejlepších hypotéz při trénování. Trénování probíhalo na train sadě při použití optimálních parametrů pro daný klasifikátor

7.5 Vliv velikosti trénovacích dat na přesnost detekce

O vlivu na přesnost můžeme hovořit jak u počtu mřížek, které máme k dispozici pro trénování, tak i u vztahu velikosti množiny sémantických entit určených k trénování vzhledem k hledaným sémantickým entitám.

Počet trénovacích mřížek

K trénování klasifikátoru byly využity dvě sady mřížek nazvané `train_h` a `train_t`, přičemž `train_h` sada byla přibližně 10 krát menší, než `train_t`.

Klasifikátor	Trénovací sada	ST	30ST	200ST	500ST	1000ST
Logistická regrese	`train_t`	0.713	0.683	0.704	0.662	0.700
	`train_h`	0.663	0.663	0.659	0.623	0.640
Extra-Tree	`train_t`	0.732	0.549	0.628	0.427	0.625
	`train_h`	0.596	0.489	0.520	0.302	0.475
LR + přesná shoda	`train_t`	0.682	0.673	0.681	0.652	0.676
	`train_h`	0.657	0.647	0.666	0.627	0.648
ET + přesná shoda	`train_t`	0.704	0.616	0.651	0.569	0.653
	`train_h`	0.652	0.609	0.644	0.524	0.583

Klasifikátor	Trénovací sada	TT	TTX
Logistická regrese	`train_t`	0.869	0.869
	`train_h`	0.868	0.869
Extra-Tree	`train_t`	0.869	0.867
	`train_h`	0.867	0.866
LR + přesná shoda	`train_t`	0.869	0.869
	`train_h`	0.869	0.869
ET + přesná shoda	`train_t`	0.868	0.867
	`train_h`	0.868	0.867

Tabulka 7.3: Hodnoty AUC pro různé velikosti množiny trénovacích sémantických entit a různé trénovací sady (při použití optimálních parametrů pro daný klasifikátor)

Vliv tohoto velikostního rozdílu byl určen nejprve při vyhledávání všech sémantických entit reprezentující nádražní stanice, přičemž trénování probíhalo právě s množinou vyhledávaných entit. K trénování byly použity parametry získané v předchozí části 7.3, které považujeme za optimální. Takto získanou množinu dat budeme dále v textu a tabulkách označovat ST. Stejným způsobem jsme provedli trénování a detekci pro sémantické entity typů vlaků, které budeme označovat TT.

Při porovnání obou trénovacích sad pro entity ST vidíme v tabulce 7.3, že sledovaná hodnota AUC je vyšší u sady `train_t`, tedy k trénování je vhodné použít rozsáhlejší množinu mřížek u obou typů klasifikátoru. Avšak z obr. 7.2 je patrné, že pro určité pracovní body se ROC křivky těchto sad protínají nebo

se alespoň sobě blíží. Výběr velikosti trénovací sady pak může záviset též na požadované míře FPR.

Pro menší sadu hledaných entit TT je dostatečná už sada `train_h` a větší sada nepřináší výrazné zlepšení.

Náhodný výběr množiny sémantických entit

Z množiny 2806 sémantických entit nádražních stanic byl vytvořen seznam postupně 500 a 1000 *náhodně* vybraných stanic určených k trénování. Obdobně jako u úplného seznamu i zde proběhlo trénování jak nad `train_h`, tak `train_t` sadou.

Po natrénování proběhlo vyhledávání všech 2806 dostupných stanic v testovacích promluvách. Označíme tímto způsobem vytvořené sady jako 500ST, resp. 1000ST.

V tabulce 7.3 je patrný přibližně 10% pokles AUC při použití menší sady `train_h` u logistické regrese. S použitím rozhodovacího stromu je rozdíl ještě výraznější, můžeme tedy usuzovat, že je tento typ klasifikátoru náročnější na počet trénovacích dat a taktéž jejich vhodný výběr.

Výběr ze seřazené množiny sémantických entit

Množina nádražních stanic byla seřazena podle velikosti měst ke kterým tyto stanice přísluší. Z této seřazené množiny bylo vybráno prvních 30 a prvních 200 *největších* stanic, nad kterými byl trénován klasifikátor za použití jak `train_h`, tak `train_t` sady s dříve nalezenými optimálními parametry.

Vyhledávání bylo provedeno opět s použitím všech 2806 dostupných sémantických entit stanic. Takto vytvořené sady budeme značit 30ST, resp. 200ST, obdobně jako u náhodného výběru.

Tabulka 7.3 dokazuje, že sada `train_h` s využitím klasifikace pomocí logistické regrese vykazuje stále 10% pokles AUC oproti sadě `train_t`. V porovnání s trénováním na úplné množině sémantických entit stanic vidíme v případě sady 200ST rozdíl výrazně menší, než při použití náhodného výběru trénovací množiny 500 či 1000 sémantických entit.

Při použití rozhodovacích stromů je vidět výraznější propad sad 30ST a 200ST oproti úplné množině sémantických entit i oproti výsledkům logistické regrese.

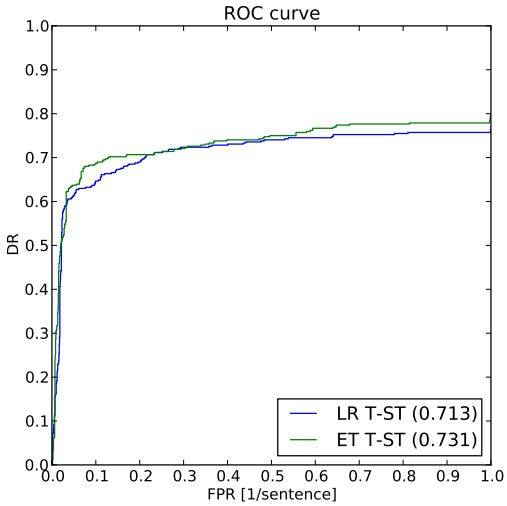
Sada `train_h` je zde opět zcela nedostatečná pro trénování.

Použití jiných než vyhledávaných sémantických entit

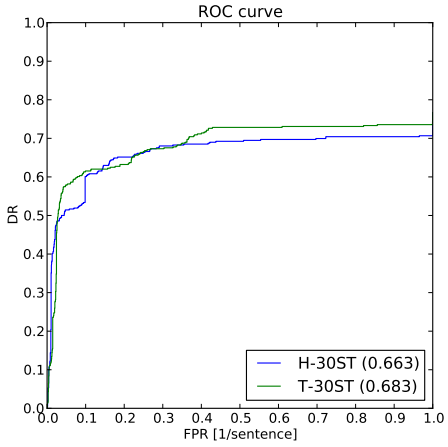
Dalším experimentem bylo natrénování s dříve získanými optimálními parametry nad plným seznamem 2806 sémantických entit nádražních stanic, přičemž samotná detekce probíhala nad 9 sémantickými entitami typů vlaků.

Cílem tohoto experimentu je zjistit, zda můžeme natrénovat obecný klasifikátor, který můžeme použít pro dosud nehledaná slova.

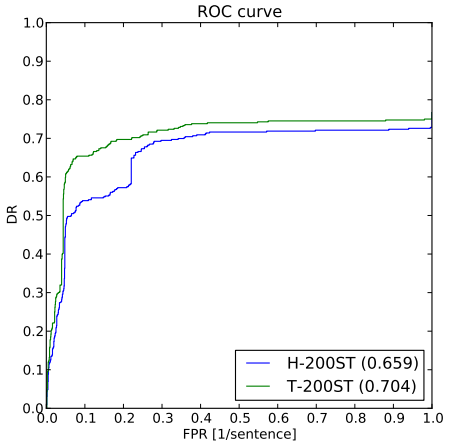
Takto získaná sada dat je označena TTX a v tabulce 7.3 je patrné, že úspěšnost detekce hodnocená AUC nezaznamenala při použití logistické regrese pokles oproti trénování přímo na množině vyhledávaných entit. U extrémně znáhodněných stromů byl zaznamenán mírný pokles hodnoty AUC.



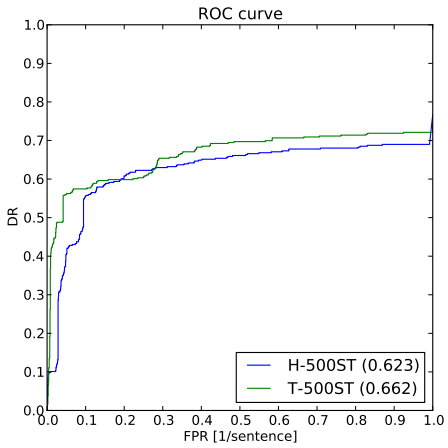
Obrázek 7.1: ROC křivka detekce nádražních stanic logistickou regresí (LR) a extrémně znáhodněnými rozhodovacími stromy (ET) v případě trénování nad sadou `train_t`



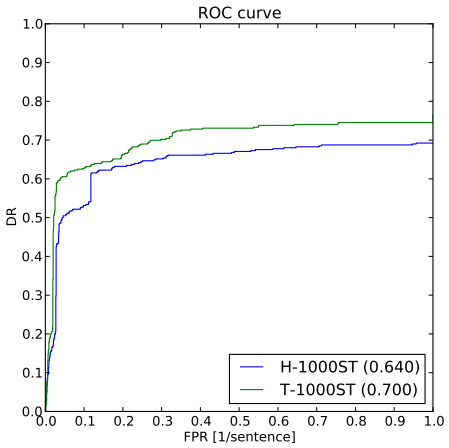
(a) 30 největších stanic



(b) 200 největších stanic



(c) 500 náhodných stanic



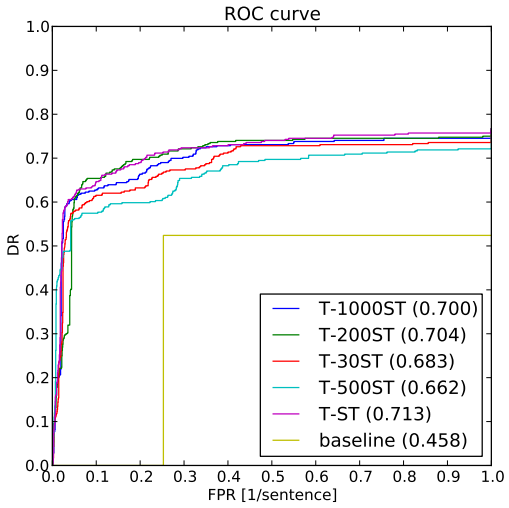
(d) 1000 náhodných stanic

Obrázek 7.2: ROC křivka ukazující vliv velikosti množiny trénovacích mřížek na sledovaný parametr kvality detekce AUC (uveden v závorce). H značí sadu `train_h` a T větší sadu `train_t`. Klasifikátor využíval logistickou regresi s parametry z části 7.3

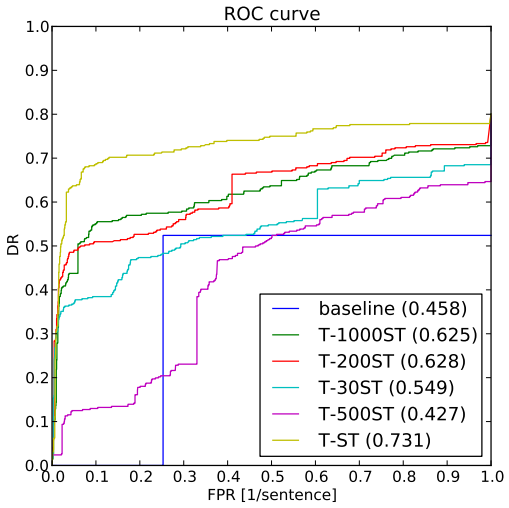
7.6 Výpočetní nároky

Paměťově nejnáročnější bylo trénování klasifikátoru a to v závislosti na míře decimace. Některé kombinace parametrů v závislosti na decimování negativních příkladů vygenerovaly vektorů příznaků tolik (přes $7 \cdot 10^6$), že překonaly i maximum 23 GB hranici paměti která byla k dispozici a algoritmus byl z důvodu nedostatku paměti ukončen.

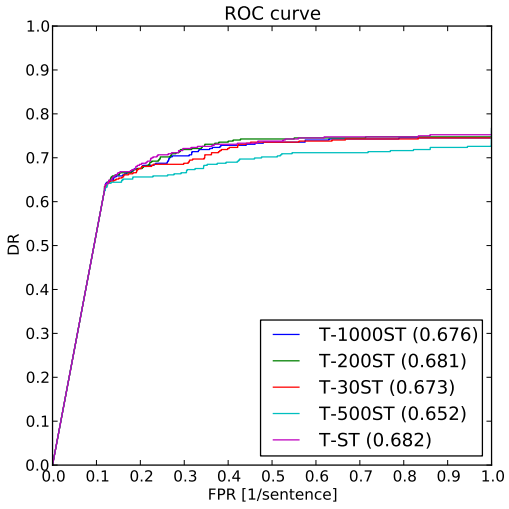
Samotná detekce po natrénování neměla zvýšené paměťové nároky, nicméně na testovací sadě 1439 promluv (jejich délka byla přibližně 1 hodina a 39 minut) si průměrně vyžádala k běhu 50 minut.



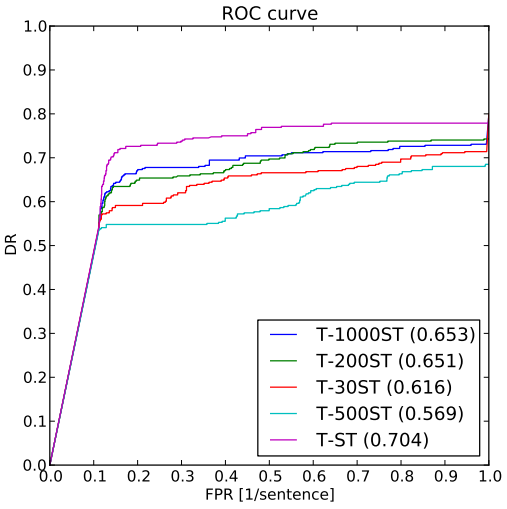
(a) Logistická regrese



(b) Extrémně znáhodněné stromy



(c) Logistická regrese kombinovaná s přístupem přesné shody



(d) Extrémně znáhodněné stromy kombinované s přístupem přesné shody

Obrázek 7.3: ROC křivka ukazující vliv velikosti množiny sémantických entit určených pro trénování na přesnost detekce. Trénování nad sadou train. Detekce probíhala s parametry podle části 7.3

8. Závěr

S využitím vážených konečných automatů a klasifikátorů byl vytvořen model umožňující detekci klíčových slov vhodnou kombinací fonémových mřížek s lexikálními realizacemi vyhledávaných sémantických entit.

Z výsledků experimentů jasně vyplývá, že navržený algoritmus přináší zlepšení ve sledovaném parametru *AUC* oproti baseline pro oba typy klasifikátorů. Jednoznačně se také potvrdila výhoda fonémových mřížek oproti pouhé první nejlepší hypotéze z nich.

Přestože nejlepšího výsledku při dostatečném množství trénovacích dat bylo dosaženo s klasifikátorem využívajícím extrémně znáhodněné stromy, na omezených množinách trénovacích dat podával lepší výsledky klasifikátor využívající logistickou regresi. Doplnění klasifikátoru s extrémně znáhodněnými stromy o entity detekované přístupem přesné shody vedlo ke zlepšení sledovaného parametru *AUC* u menších množin trénovacích dat, nicméně stále nedosahoval hodnot samotné logistické regrese.

Zajímavých výsledků bylo dosaženo u hledání velmi malé množiny sémantických entit určujících typ vlaků. Zde je vidět, že i po natrénování klasifikátoru na zcela jiné množině entit je možné nadále s vysokou úspěšností detekovat.

Pro porovnání dodejme, že byl k dispozici i výsledek detekce nad první nejlepší hypotézu ze *slovních* mřížek, který dosahoval AUC 0.785 nad sémantickými entitami vlakových nádraží a AUC 0.883 nad entitami typů vlaků a navržený algoritmus dosáhl AUC 0.732, resp. 0.869. Je však třeba připomenout, že k tvorbě slovních rozpoznávačů vyžadujeme jazykový model vytvořený z mnohem většího množství dat, než které využíváme u námi navržené metody.

8.1 Budoucí práce

Přestože už při vytváření algoritmu byla snaha eliminovat operace zvyšující paměťové a časové nároky na detekci (např. omezením počtu vložení a smazání při určování editační vzdálenosti), stále je prostor pro další optimalizaci imple-

mentovaného algoritmu.

Časové nároky na detekci můžeme snížit paralelizací části algoritmu, nicméně zde je třeba analyzovat případná omezení jazyka Python a najít vhodné přístupy k paralelizaci. Jedním z těchto přístupů může být model MapReduce [8]. Tento model využívá 2 fází - `map` a `reduce`, které probíhají postupně na uzlech výpočetních clusterů, přičemž v rámci každé fáze se počítá paralelně na více datech (a opět více uzlech). Metoda `map` pak může provádět načtení prohledávané mřížky a následně `reduce` provede kompozici s hledaným dotazem. V závěrečné fázi dojde ke konsolidaci výsledků.

Jako další využití implementovaného algoritmu je zapojení do stávajících řešení indexace a vyhledávání v řečových archivech [22] a analýza jeho vlivu na kvalitu vyhledávání nad fonémovými mřížkami.

Literatura

[1] Allauzen, C.; Riley, M.; Schalkwyk, J.; et al.: *OpenFst: A General and Efficient Weighted Finite-State Transducer Library.* IN *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)*, *Lecture Notes in Computer Science*, ročník 4783, Springer, 2007, s. 11–23, `http://www.openfst.org`.

[2] Allauzen, C.; Riley, M.; Schalkwyk, J.; et al.: *OpenFst.* Duben 2011. Dostupné z WWW: `<http://www.openfst.org/twiki/bin/view/FST/FstExamples#Edit_Distance>`

[3] Béchet, F.; Gorin, A. L.; Wright, J. H.; et al.: *Detecting and extracting named entities from spontaneous speech in a mixed-initiative spoken dialogue context: How May I Help You? Speech Communication*, ročník 42, č. 2, Únor 2004: s. 207–225, ISSN 01676393, doi:10.1016/j.specom.2003.07.003.

[4] Bishop, C. M.: *Pattern Recognition and Machine Learning (Information Science and Statistics).* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006, ISBN 0387310738.

[5] Can, D.: *Indexation, Retrieval & Decision Techniques for Spoken Term Detection.* 2010. Dostupné z WWW: `<http://www.busim.ee.boun.edu.tr/~dogan/files/msthesis-dogancan.pdf>`

[6] Černá, I.; Křetínský, M.; Kučera, A.: *Formální jazyky a automaty I.* Brno: Masarykova univerzita, 2006. Dostupné z WWW: `<http://is.muni.cz/elportal/?id=703389L>`

[7] Crochemore, M.: *Transducers and repetitions. Theoretical Computer Science*, ročník 45, 1986: s. 63–86. Dostupné z WWW: `<http://www.sciencedirect.com/science/article/pii/0304397586900411>`

[8] Dean, J.; Ghemawat, S.: *MapReduce: simplified data processing on large clusters. Commun. ACM*, ročník 51, č. 1, Leden 2008: s. 107–113, ISSN 0001-0782, doi:10.1145/1327452.1327492. Dostupné z WWW: `<http://doi.acm.org/10.1145/1327452.1327492>`

[9] Demlová, M.; Koubek, V.: *Algebraická teorie automatů*. Praha: SNTL, 1990, ISBN 80-03-00348-2.

[10] Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; et al.: *LIBLINEAR: A Library for Large Linear Classification. Journal of Machine Learning Research*, ročník 9, 2008: s. 1871–1874.

[11] Geurts, P.; Ernst, D.; Wehenkel, L.: *Extremely randomized trees. Machine Learning*, ročník 63, č. 1, 2006: s. 3–42, ISSN 0885-6125, doi:10.1007/s10994-006-6226-1. Dostupné z WWW: <http://dx.doi.org/10.1007/s10994-006-6226-1>

[12] Hunt, A.; McGlashan, S.: *Speech Recognition Grammar Specification Version 1.0*. W3C recommendation, W3C, Březen 2004, http://www.w3.org/TR/2004/REC-speech-grammar-20040316/. Dostupné z WWW: <http://w2.syronex.com/jmr/w3c-biblio>

[13] Ircing, P.: *Hierarchical Discriminative Model for Spoken Language Understanding. Text, Speech and Dialogue*, 2013, bude publikováno.

[14] Ircing, P.; Svec, J.: *Semantic entity detection from the ASR lattices within the WFST framework. Text, Speech and Dialogue*, 2013: s. 4–8, bude publikováno.

[15] Jurčíček, F.; Zahradil, J.; Jelínek, L.: *A human-human train timetable dialogue corpus. Proceedings of EUROSPEECH, Lisboa*, 2005: s. 1525–1528.

[16] Mohri, M.; Pereira, F.; Riley, M.: *Weighted finite-state transducers in speech recognition. Computer Speech & Language*, ročník 16, č. 1, Leden 2002: s. 69–88, ISSN 08852308, doi:10.1006/csla.2001.0184. Dostupné z WWW: <http://linkinghub.elsevier.com/retrieve/pii/S0885230801901846>

[17] Novotnỳ, M.: *S algebrou od jazyka ke gramatice a zpět*. Cesta k vědění, Academia, 1988.

[18] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; et al.: *Scikit-learn: Machine Learning in Python . Journal of Machine Learning Research*, ročník 12, 2011: s. 2825–2830.

[19] Pereira, F. C. N.; Wright, R. N.: *Finite-state approximation of phrase structure grammars*. IN *Proceedings of the 29th annual meeting on Association for Computational Linguistics ACL '91*, Berkeley: Association for Computational Linguistics, 1991, s. 246–255, doi:10.3115/981344.981376.

[20] Psutka, J.; Müller, L.; Matoušek, J.; et al.: *Mluvíme s počítačem česky*. Praha: Academia, 2006, ISBN 80-200-1309-1, 752 s.

[21] Skantze, G.; Schlangen, D.: *Incremental dialogue processing in a micro-domain. Proceedings of the 12th Conference of the ...*, , č. April, 2009: s. 745–753. Dostupné z WWW: <http://dl.acm.org/citation.cfm?id=1609150>

[22] Vavruška, J.: *Vyhledávání slov v rozsáhlém archívu mluvené řeči*. Diplomová práce, Západočeská univerzita v Plzni, Plzeň, 2012.

[23] Švec, J.; Šmídl, L.: *On the Use of Phoneme Lattices in Spoken Language Understanding. Text, Speech and Dialogue*, 2013, bude publikováno.

[24] Wikipedia: *Levenshtein distance — Wikipedia, The Free Encyclopedia*. 2013, [Online; 13.5.2013]. Dostupné z WWW: <http://en.wikipedia.org/w/index.php?title=Levenshtein_distance&oldid=554850983>

[25] Young, S. J.; Kershaw, D.; Odell, J.; et al.: *The HTK Book Version 3.4*. Cambridge University Press, 2006.