VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

METODY STROJOVÉHO UČENÍ VE ZPRACOVÁNÍ PŘIROZENÉHO JAZYKA

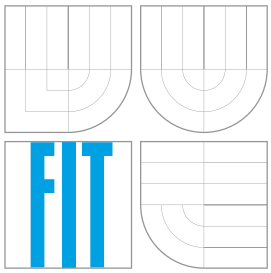
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MAREK VANTUCH

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

METODY STROJOVÉHO UČENÍ VE ZPRACOVÁNÍ PŘIROZENÉHO JAZYKA

MACHINE-LEARNING METHODS IN NATURAL LANGUAGE PROCESSING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MAREK VANTUCH

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. LUBOMÍR OTRUSINA

BRNO 2011

Abstrakt

Práce se zabývá automatickým značkováním českého jazyka za pomocí existujících implementací, využívajících model Conditional Random Fields a algoritmy L-BFGS a SDG. Jsou popsány základní pravidla značkování a problémy, se kterými se tento obor potýká v případě zpracování českého jazyka. Čtenáři jsou vysvětleny principy použitých algoritmů a modelů, které jsou implementovány v programech CRF++ a CRFSuite. Práce se poté zaměřuje na vlastní testování úspěšnosti na českém korpusu a snaží se nalézt nejvhodnější hodnoty parametrů při využití všech rysů. Při nalezení rozumného kompromisu mezi časem a přesností se poté snaží tuto hodnotu ještě zpřesnit za pomoci analýzy přínosu jednotlivých rysů a možností jejich vynechání.

Abstract

Firstly, basic rules of tagging of the Czech language are described as well as problems connected to this field. Thereafter the focus of the thesis is put on the success rate of testing on the Czech corpus and at the same time trying to find the most suitable parameter values for using the features. After reaching a reasonable compromise between duration and accuracy, the value is then attempted to be improved using analysis of separate features and their eventual omission.

Klíčová slova

strojové učení, značkování textu, Conditional Random Fields, L-BFGS

Keywords

Machine Learning, POS tagging, Conditional Random Fields, L-BFGS

Citace

Marek Vantuch: Metody strojového učení ve zpracování přirozeného jazyka, bakalářská práce, Brno, FIT VUT v Brně, 2011

Metody strojového učení ve zpracování přirozeného jazyka

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Lubomíra Otrusiny

.....
Marek Vantuch
16. května 2011

Poděkování

Děkuji panu Ing. Lubomíru Otrusinovi za jeho pomoc a vedení při tvorbě této práce, RNDr. Pavlu Smrži, PhD. za možnost na tomto tématu pracovat a Ing. Marku Schmidtovi za poskytnuté rady.

Obsah

Seznam tabulek

Kapitola 1

Úvod

Tato práce pojednává o využití programů určených k řešení úloh strojového učení (Conditional Random Fields s využitím algoritmů Limited-memory BFGS a Stochastic Gradient Descent) a porovnání výsledků s existujícími řešeními pro značkování českého jazyka, kterými v našem případě je program *Morče* [1] vyvíjený na pražské Univerzitě Karlově (používající Hidden Markov Model). Pro tento úkol byly vybrány programy *CRF++* a *CrfSuite*.

V dokumentu bude nejdříve popsáno značkování češtiny a pravidla s ním spojená. Budou zmíněny potíže, které jsou s ním spojeny v porovnání s jinými, morfologicky jednoduššími jazyky. Dále budou rozebrány matematické modely a algoritmy, které jsou použity při procesu učení. K celému procesu bylo potřeba vytvořit několik nástrojů, buď konvertujících vstupní data pro jednotlivé programy, nebo ulehčujících vlastní proces testování. Mezi tyto patří skritpy napsané v jazyce *Python* a *BASH* a informační systém, starající se o správu běžících a dokončených testů a informací o nich, implementovaný za pomocí *PHP*, *SQL* a *HTML*, postavený na open-source CMS[2] systému *Drupal*[3]. Veškerý kód byl spravován pomocí verzovacího systému *GIT*[4] a je veřejně dostupný na internetu[5].

Nejdůležitější částí dokumentů pak jsou výsledky testování, zpracovávaného na školních strojích přes vzdálený přistup. Testy byly provedeny z mnoha různých úhlů pohledu na efektivitu a náročnost.

[1] Stránky programu: http://ufal.mff.cuni.cz/morce/

[2] Z angl. content management system - software pro správu obsahu

[3] Stránky programu: http://www.drupal.org

[4] Stránky programu: http://git-scm.com/

[5] Adresa repozitáře: http://github.com/kanei/MoTag

1.1 Termíny

Pro další text definujeme následující termíny:

- **Značka** – *angl. Label* – řetězec popisující vlastnost daného prvku, v našem případě určení mluvnických tříd
- **Rys** – *angl. Feature* – prvek používaný při trénování, specifikující vlastnost daného prvku ve vstupních datech
- **Vlastnost** – *angl. Atribut* – pravidlo pro generování jednotlivých rysů
- **Lemma** – základní podoba slova (slovníkový tvar)

Kapitola 2

Značkování českého jazyka

2.1 Proč značkovat jazyk

Při mnoha úkonech v dnešní době chceme po počítači, aby co nejlépe rozumněl našim požadavkům a byl schopen nám poskytnout co nejpřesnější výsledky. K tomuto je často potřeba, aby stroj pochopil význam textu a tím pádem morfologické stavbě slov. V českém jazyce má mnoho slov tzv. homonymní tvar – jedno slovo může být v různých případech naprosto odlišným slovním druhem, nebo mít odlišné určení slovních kategorií (např. pila, kolem, ...). Při strojové analýze textu, překladech apod. je proto důležité, abychom byli schopni tyto kategorie rozlišovat a přiřazovat jednotlivým slovům jejich správnou značku.

Při procesu značkování je pro každé slovo vstupního textu přiřazena značka, určující jeho mluvnické kategorie. Tento proces je často rozdělen na několik kroků, kdy slovu nejdříve přiřadíme všechny možné značky a poté teprve vybereme tu správnou. V této práci se budeme zabývat pouze druhým krokem a všechna vstupní data proto již budou mít nastavena množinu všech značek, připadajících v úvahu, a také tu správnou (ručně zadanou) poskytující nám vlastní možnost trénování.

2.2 Kategorie popisované značkou v českém jazyce

Každé slovo českého jazyka má mnoho kategorií, do kterých může být zařazeno – od slovního druhu po slovesný vid. Ve strojovém zpracování jazyka lze použít několika různých přístupů k uložení těchto informací. V našem případě bude struktura značek odpovídat tzv. pozičnímu systému[1] používanému na pražské Univerzitě Karlově (existuje také tzv Brněnský systém[2]). Tento systém se skládá ze 16 pozic (viz. tabulka 2.1).

[1] Popis morfologických značek: http://ucnk.ff.cuni.cz/bonito/znacky.php

[2] Více informací a srovnání na: http://ufal.mff.cuni.cz/jazz/pml/old/skladiste/morfologicka_sbirka.htm

Pozice	Popis
1	Slovní druh
2	Detail slovního druhu
3	Jmenný rod
4	Číslo
5	Pád
6	Přivlastňovací rod
7	Přivlastňovací číslo
8	Osoba
9	Čas
10	Stupeň
11	Negace
12	Aktivum/pasivum
13	Nepoužito
14	Nepoužito
15	Varianta, stylový příznak a pod.
16	Slovesný vid

Tabulka 2.1: Popis jednotlivých pozic morfologické značky

2.3 Problémy při značkování

Zatímco v angličtině při pouhém označkování slov s jednoznačnou značkou dosáhneme úspěšnosti kolem 90%, v českém jazyce je situace mnohem složitější. Kvůli výše zmíněné homonymii a množství kategorii, do kterých v českém jazyce slova můžeme zařadit, bychom se v našem případě k 90% určitě ani nepřiblížili. V angličtině se proto také používá pouze tří-pozicová značka, která velmi ulehčuje práci veškerým automatickým programům. Při takto nízkém počtu je také velmi nízká paměťová náročnost a počet možných přechodů není tak vysoký. Naproti tomu v češtině je kvůli 16 pozicím možno získat celkový počet značek v řádu tisíců unikátních stavů a tím se neúměrně zvyšuje časová i paměťová náročnost celého procesu značkování. Z tohoto důvodu jsem se pokoušel z této náročnosti ubrat vypuštěním méně důležitých kategorií a tím snížit celkový počet variant k číslu nepřevyšujícímu jeden tisíc. Náročnost na procesorový čas se tím razantně snížila a dala mi tak šanci otestovat mnohem větší množství kombinací rysů a nastavení programů.

2.4 Data

K trénování a následné evaluaci výsledků máme k dispozici okolo 77 000 vět. Data jsou získána z korpusu *PDT 2.0*[1] a uložena ve formátu dat *CSTS*[2]. Pomocí skriptů v jazyce *Python* jsou tato data následně převedena do formátu kompatibilního s použitými programy.

2.5 Existující řešení pro značkování českého jazyka

Pro porovnání výsledků této práce byl použit program *Morče* vytvořený na Ústavu formální a aplikované lingvistiky na Univerzitě Karlově v Praze. K trénování je využíván skrytý

[1]Stránky PDT 2.0: http://ufal.mff.cuni.cz/pdt2.0/
[2]Definice: http://ufal.mff.cuni.cz/pdt2.0/doc/data-formats/csts/html/DTD-HOME.html

markovský model s použitím průměrovaného perceptronu. Vstupními daty je korpus *PDT 2.0* s údaji ve formátu *CSTS*. Podle dostupných údajů program dosahuje úspěšnosti od 95 do 96 procent je tudíž nejpřesnějším značkovacím programem pro zpracování českého jazyka. Z důvodu použití stejných dat tedy nebude problém porovnat konečnou úspěšnost námi použité implementace s úspěšností programu *Morče*. Další informace o konkurenčním řešení a jeho funkcionalitě, které z velké části posloužily i jako inspirace při této práci lze nalézt v [12] a [11].

Kapitola 3

Popis algoritmů a modelů

Pro odlišení od existujících řešení byl vybrán algoritmus Conditional Random Fields (CRF), rozšiřující Hidden Markov Model a tím nabízející vyšší účinnost na vstupních datech. Pro lepší pochopení CRF je potřeba nejdříve porozumět způsobu, jakým pracuje algoritmus HMM. V mnoha existujících řešeních se poté pro určení parametrů používají různé algoritmy od Viterbiho algoritmu, používaného již od 60. let minulého století, přes Forward-Backward algoritmus, jeho modifikaci Baum-Welchův algoritmus, až po „nejmodernější" zástupce, kterými jsou Low Memory BFGS a Stochastic Gradient Descent, poskytující nejlepší výsledky a snížení paměťové a časové náročnosti.

Celý proces probíhá za tzv. učení s učitelem (Supervised Learning), kdy programu nejdříve předložíme již správně označená data a necháme jej, aby se pomocí jednoho z výše zmíněných algoritmů naučil správné parametry modelu (CRF nebo HMM). Takto získaný model pak aplikujeme na menší skupinu dat, pomocí které určíme konečnou úspěšnost celého procesu.

3.1 Skrytý markovský model (HMM)

Základní informace

Skrytý markovský model[1], popsaný v [2], [4] a [9], pojmenovaný po ruském matematikovi Andreyi Markovi, označuje statistický model určený pro popis posloupnosti obsahující Markovské stavy. Těmi máme na mysli takový stav, pro něž je důležitá pouze blízká minulost (v našem případě pouze několik předchozích stavů – značek) a vzdálená minulost je zanedbána. Ve zpracování jazyka se často omezuje zpracování pouze na dvě předchozí značky.

Popis a definice

HMM je konečným automatem, vyjádřeným jako (A, B, Π), kde A popisuje množinu hodnot pravděpodobností přechodu (označených jako a), B množinu hodnot pravděpodobností generování – výstupu (označených jako b) a Π množinu hodnot pravděpodobností úvodního stavu (označených jako p). Množina skrytých stavů Q (s jednotlivými hodnotami q_i) a seznam výstupních hodnot (pozorování) O (s hodnotami o) jsou pokládány za známé a nejsou tedy zahrnuty do definice. Aktuální stav není pozorovatelný a místo toho každý stav s určitou pravděpodobností(B) generuje výstup. Následující definice byly převzaty z článku [10].

[1] angl. Hidden Markov Model

$$\Lambda = (A, B, \Pi) \tag{3.1}$$
$$\Pi = \{p_i = P(q_i[t = 1])\} \tag{3.2}$$
$$Q = \{q_i\}, i = 1, ..., N \tag{3.3}$$

N označuje počet skrytých stavů.

$$A = \{a_{ij} = P(q_j[t + 1]|q_i[t])\} \tag{3.4}$$

$P(a|b)$ je pravděpodobnost a za předpokladu b, t vyjadřuje čas a q_i je prvkem z množiny Q.

A je tedy pravděpodobnost, že následující stav je q_j za předpokladu, že aktuálním stavem je q_i.

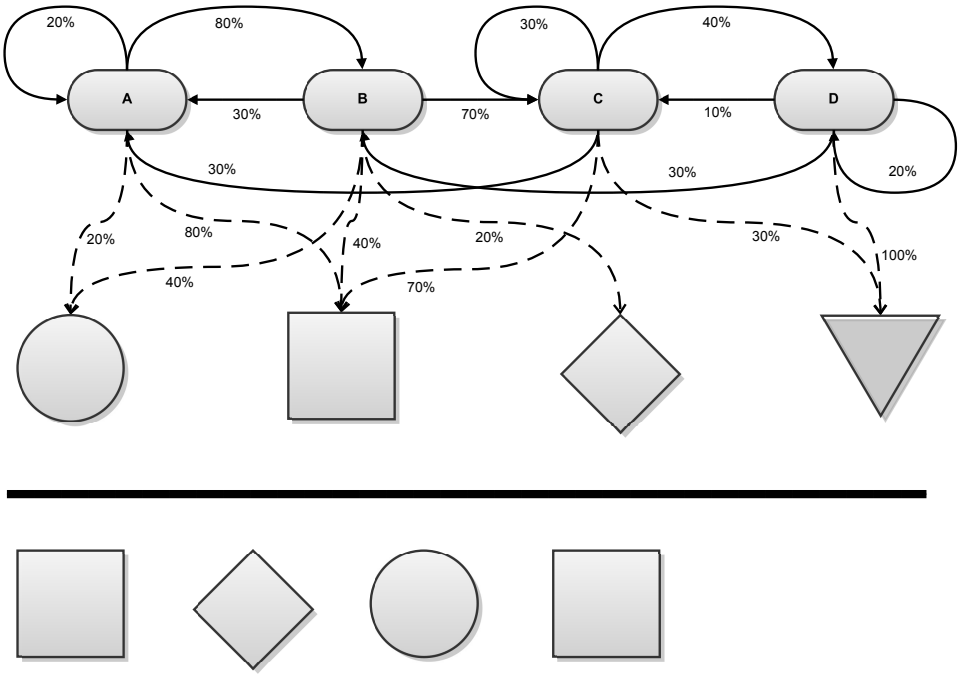
$$O = \{o_k\}, k = 1, ..., M \tag{3.5}$$

M označuje počet pozorovaných(vstupních) hodnot.

$$B = \{b_{ik} = b_i(o_k) = P(o_k|q_i)\} \tag{3.6}$$

o_k je prvkem množiny O.

O je pravděpodobnost, že výstup je o_k za předpokladu, že aktuálním stavem je q_i



Obrázek 3.1: Graf možných přechodů a výstupů modelu

A → B → A → A
A → B → A → B
A → B → B → A
A → B → B → C

Tabulka 3.1: Možné přechody mezi skrytými stavy modelu

Příklad

Zatímco u klasického Markovského modelu známe posloupnost vstupních hodnot, v našem případě známe pouze posloupnost hodnot výstupních (slovních tvarů). V každém kroku je poté uložena hodnota pravděpodobnosti přechodu do možných následujících stavů a hodnota pravděpodobnosti vygenerování určitého výstupu (značky). Celý proces je nastíněn na obrázku 3.1.

Horní řada popisuje množinu skrytých stavů(Q) a plné čáry možné přechody mezi nimi s vyznačenou pravděpodobností(A). Přerušované spojnice označují pravděpodobnosti vygenerování obrazců z jednotlivých stavu(B) a obrazce pod čarou (čtverec, kosočtverec, trojúhelník) jsou posloupností výstupních (pozorovaných) stavů(O).

Vysvětlení Abychom získali v prvním kroku kruh, musíme aktuální stav, ve kterém se nacházíme, mít možnost kruh vygenerovat – v našem případě jsou to stavy A a B. Zapíšeme je tedy do prvního sloupečku tabulky. Abychom jako druhý prvek posloupnosti získali kosočtverec, musíme být ve B, neboť ten jako jediný poskytuje tento obrazec. Při prvním kroku jsme si vypsali A a B, ve B však nemůžeme setrvat a ten tudíž vypadává z možných posloupností a zůstává nám posloupnost ($A \rightarrow B$). Kdybychom takto pokračovali až do konce, získáme čtyři možnosti, zachycené v tabulce 3.1. Nejpravděpodobnější z nich bychom byli schopni určit pomocí ohodnocení jednotlivých přechodů a výběrem toho s nejvyšší pravděpodobností, k čemuž jsou využity níže popsané algoritmy, využívající různých metod k optimalizaci tohoto procesu.

3.2 Conditional Random Fields (CRF)

Základní informace

CRF můžeme popsat jako bezsměrový grafický pravděpodobnostní model (undirected probabilistic graphical model), ve kterém každý uzel (v našem případě slovo) popisuje náhodnou proměnnou. CRF může také být vyjádřen jako model s konečným počtem stavů a nenormalizovanými pravděpodobnostmi přechodů. V porovnání k HMM nemá jasně dané hodnoty přechodů mezi stavy a disponuje možností mnohonásobných funkcí, generujících rysy. Přináší také lepší výsledky ve zpracování dat se závislostmi vyššího řádu, které většinou lépe odpovídají reálnému modelu. Na rozdíl od generativních modelů také, jakožto pravděpodobnostní (conditional) model nemusí zkoumat všechny možné sekvence pozorovaných prvků. Pro určení parametrů tohoto modelu je v současné době nejvíce doporučován algoritmus L-BFGS (Low-Memory BFGS), který by měl poskytovat nejkvalitnější výsledky (v případě této práce se to však nepotvrdilo a vhodnější se nakonec ukázal algoritmus SGD).

Popis a definice

Jak je popsáno v [5], nechť X je náhodnou proměnnou dat (v našem případě vět) určených k označkování a Y náhodnou proměnnou korespondujících značek. Všechny Y_i z Y náleží abecedě ξ, která tedy obsahuje množinu všech použitých značek. Vytvoříme pak model $P(X|Y)$ z párů pozorovaných hodnot a sekvencí značek. CRF pak můžeme vyjádřit následujícím odstavcem:

Definice *Nechť $G = (V, E)$ je grafem, u kterého platí $Y = (Y_v)_{v \in V}$ tak, že Y je indexováno body vektoru G. Poté (X, Y) vyjadřuje CRF v případě, kdy náhodné proměnné Y_v jsou podmíněné X a vyhovují Markovské vlastnosti s ohledem na graf vyjádřený vzorcem 3.7, kde $w \sim v$ značí, že prvky w a v spolu ve vektoru G sousedí.*

$$p(Y_v|X, Y_w, w \neq v) = p(Y_v|X, Y_w, w \sim v) \tag{3.7}$$

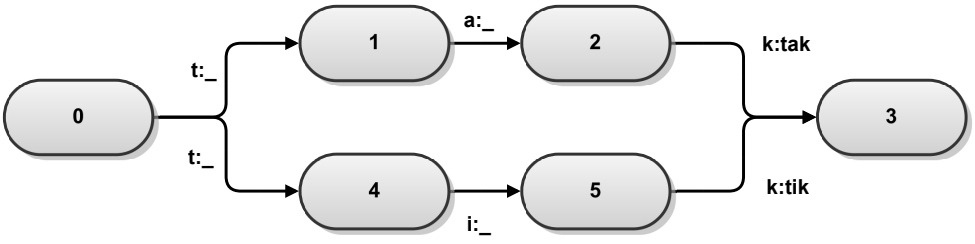
Ačkoliv by G, X i Y mohly nabývat různých forem grafů, pro potřeby této práce je budeme považovat za jednoduché řetězce - $G = (V = \{1, 2, \ldots m\}, E = \{(i, i+1)\})$, $X = (X_1, X_2, \ldots, X_n)$ a $Y = (Y_1, Y_2, \ldots, Y_n)$. Díky skutečnosti, že řetězce jsou nejjednodušší formou stromu můžeme základní teorém CRF definovat vzorcem 3.8, ve kterém x je sekvencí dat, y sekvencí značek a $y|_s$ je množinou součástí y spojených s vrcholy podgrafu s. Rysy modelu jsou označeny jako f_k a g_k.

$$p\theta(y|x) \propto exp\left(\sum_{e \in E,k} \lambda_k f_k(e, y|_e, x) + \sum_{v \in V,k} \mu_k g_k(v, y|_v, x)\right) \tag{3.8}$$

3.2.1 Label Bias problém

Problém špatného vyvažování značek (viz. [5]) spočívá v přesunu pravděpodobnostní hodnoty na další pozici v případě pouze jediného přechodu za ignorování pozorovaného prvku. Jak je ilustrováno na obrázku 3.2, v případě vstupní posloupnosti písmen *t,i,k* se v prvním kroku neumíme rozhodnout a vstupujeme do stavů *1* a *4* se stejnou pravděpodobností. Nyní, i když na vstupu je písmeno i, nemáme jinou možnost, než pokračovat do stavů *2*,

případně *5* za ignorování pozorovaného písmena a v případě HMM přenesení celé pravděpodobnostní hodnoty do dalších stavů. V případě, že by v trénovacích datech tedy převládal řetězec „tik“ nad řetězcem „tak“, byly by oba tyto řetězce vždy určeny jako „tik“, protože by byl ignorován vstup při druhém kroku, a tím pádem by byla využita cesta s vyšší vstupní hodnotou pravděpodobnosti. Toto je vyřešeno algoritmem CRF pomocí ohodnocení přechodů mezi stavy, kdy můžeme podle aktuálního pozorovaného prvku buď snížit, nebo zvýšit konečnou pravděpodobnost dané posloupnosti.



Obrázek 3.2: Ilustrace „Label Bias“ problému

3.3 Viterbiho algoritmus

Základní informace

Viterbiho algoritmus [3] byl navrhnut v roce 1967 Andrewem Viterbim jako dekódovací algoritmus pro konvoluční kódy v telekomunikačních sítích. Kvůli svému zaměření na vyhledání nejpravděpodobnější cesty sekvencí skrytých stavů (nejčastěji HMM) se však používá v mnoha odvětvích jako např. zpracování signálů, nebo značkování jazyka.

Popis a definice

Za předpokladu, že vstupní data mají stejný počet pozorovaných a skrytých stavů, obě posloupnosti jsou zarovnané a výpočet v daném místě je závislý pouze na aktuálním a předchozím prvku, můžeme s jeho pomocí získat tzv. Viterbiho cestu – nejpravděpodobnější cestu posloupností. V každém kroku algoritmus vyhodnotí všechny cesty vedoucí k aktuálnímu stavu a ponechá pouze tu nejlepší, díky čemuž není nucen uchovávat všechny existující cesty, ale pouze jednu pro každý stav. Za pomoci uchovávaní ceny stavu se pokaždé algoritmus rozhodne ze všech následujících možností a uloží kompletní cestu od začátku průchodu posloupností. Následující definice byly převzaty z článku [10].

Definice Nechť $\delta_t(i)$ je posloupností maximálních pravděpodobností, končící ve stavu i a pro daný model produkuje prvních t pozorovaných hodnot. Parametry, určující maximální hodnotu $\delta_t(i)$ jsou ukládány v matici ψ o rozměrech N na T, kde N je celkový počet stavů a T celkový počet pozorovaných hodnot. Písmena a, b, o, p a q mají stejný význam jako v případě popisu algoritmu hmm (viz. podkapitola 3.1).

$$\delta_t(i) = max\{P\big(q(1), q(2), \ldots, q(t-1); o(1), o(2), ..., o(t)|q(t) = q_i\big).\} \tag{3.9}$$

V prvním kroku algoritmu jsou maximální hodnota a hodnoty parametrů inicializovány následujícími vzorci:

$$\begin{aligned} \delta_1(i) &= p_i b_i\big(o(1)\big) \\ \psi_1(i) &= 0, i = 1, \ldots, N \end{aligned} \tag{3.10}$$

Pomocí rekurzívního zpracování jsou pak v jednotlivých krocích získávány aktuální hodnoty, závislé pouze na hodnotách předchozího kroku. Ty jsou postupně ukládány do δ a ψ:

$$\begin{aligned} \delta_t(j) &= max_i\big[\delta_{t-1}(i)a_{ij}\big]b_j\big(o(t)\big) \\ \psi_t(j) &= argmax_i\big[\delta_{t-1}(i)a_{ij}\big] \end{aligned} \tag{3.11}$$

Ukončení proběhne pomocí dosažení pozorované hodnoty na pozici T. V posledním kroku jsou provedeny následující operace:

$$\begin{aligned} p* &= max_i\big[\delta_T(i)\big] \\ q*_T &= arg\ max_i\big[\delta_T(i)\big] \end{aligned} \tag{3.12}$$

Pro zpětné dohledání celé cesty pak použijeme následující vzorec:

$$q*_t = \psi_t + 1(q*_{t+1}), t = T-1, T-2, \ldots, 1 \tag{3.13}$$

3.4 Forward-backward algorithm

Základní informace

Používá dva průchody posloupností stavů – dopředu a zpět (podle toho také vznikl jeho název). Nejdříve projde celou sekvenci a určí cenu cesty při daném průchodu. Poté projde sekvencí zpět a při každém kroku spočítá pravděpodobnost pozorování zbylých vstupních hodnot. Oba tyto průchody jsou následně „vyhlazeny“ do společného výsledku. Speciálním případem je Baum-Welchův algoritmus, jehož příklad zpracování je dostupný v [6].

Popis a definice

V prvním kroku je proveden algoritmus průchodu směrem od začátku ke konci. Nejprve je vypočtena úvodní pravděpodobnost posloupnosti o jediném prvku $o(1)$. Pravděpodobnost $\alpha_t(i+1)$ vypočteme vynásobením všech $\alpha_t(i)$ odpovídající pravděpodobností přechodu do následujícího stavu a výsledek vynásobíme výstupní pravděpodobností prvku $o(t+1)$. Po vypočtení všech kroků a jejich sečtení získáme požadovanou celkovou pravděpodobnost. Písmena a, b, o, p a q mají stejný význam jako v případě popisu algoritmu hmm (viz. podkapitola 3.1).

Definice (forward) Nechť $a_t(i)$ je pravděpodobností části sekvence pozorovaných prvků $O_t = \{o(1), o(2), \ldots, o(t)\}$ produkovaných všemi možnými posloupnostmi končícími i-tým stavem. Následující definice byly převzaty z článku [10].

$$\alpha_t(i) = P\big(o(1), o(2), \ldots, o(t)|q(t) = q_i\big) \tag{3.14}$$

V prvním kroku inicializujeme α na pozici 1.

$$\alpha_1(i) = p_i b_i\big(o(1)\big), i = 1, \ldots, N \tag{3.15}$$

Pomocí rekurzivního algoritmu pak v jednotlivých krocích vytvoříme celou posloupnost hodnot v jednotlivých stavech pro postup posloupností směrem dopředu.

$$\begin{gathered}\alpha_{t+1}(i) = \left[\sum_{j=1}^{N} \alpha_t(j) a_{ji}\right] b_i\big(o(t+1)\big) \\ i = 1, \ldots, N, t = 1, \ldots, T-1\end{gathered} \tag{3.16}$$

V posledním kroku získáme celkovou pravděpodobnost pomocí sečtení všech hodnot α na pozici T.

$$P\big(o(1)o(2)\ldots o(T)\big) = \sum_{j=1}^{N} \alpha_T(j) \tag{3.17}$$

Definice (backward) V podobném smyslu definujeme zpětnou proměnnou $\beta_t(i)$ jako podmíněnou pravděpodonost části sekvence pozorovaných prvků od $o(t+1)$ až do konce pro všechny posloupnosti začínající v i-tém stavu.

$$\beta_t(i) = P\big(o(t+1), o(t+2), \ldots, o(T) | q(t) = q_i\big). \tag{3.18}$$

Backward algoritmus počítá pomocí rekuzního průchodu směrem zpět a na rozdíl od Forward algoritmu není využíván ke generování výstupních prvků, oba jsou však využívány k aproximaci parametru HMM.

V tomto případě postupujeme od konce a jako první si tedy vyjádříme β na pozici T.

$$\beta_T(i) = 1, i = 1, \ldots, N \tag{3.19}$$

Stejně jako při průchodu směrem dopředu si pomocí rekurzivního algoritmu vyjádříme celou posloupnost stavů, kdy v tomto případě postupně využíváme následujícího prvku.

$$\begin{gathered}\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j\big(o(t+1)\big)\beta_{t+1}(j) \\ i = 1, \ldots, N, t = T-1, T-2, \ldots, 1\end{gathered} \tag{3.20}$$

Nakonec sečteme hodnoty α na pozici 1.

$$P\big(o(1)o(2)\ldots o(T)\big) = \sum_{j=1}^{N} p_j b_j\big(o(1)\big)\beta_1(j) \tag{3.21}$$

3.5 Low memory BFGS

Základní informace

L-BFGS (viz. [7]) označuje nízko paměťovou (Low memory) úpravu optimalizační metody BFGS (Broyden-Fletcher-Goldfarb-Shanno) z rodiny Quasi-Newtonových metod, určenou k aproximaci inverzní Hessovy matice[1]. Pro snížení paměťové náročnosti je ukládán pouze rozdíl v matici, jehož historie je ukládána pouze po omezený počet kroků.

[1] čtvercová matice druhých parciálních derivací funkce – popisuje místní zakřivení funkce o několika proměnných

3.6 Stochastic Gradient Descent

Základní informace

Ačkoli je algoritmus znám již dlouhou dobu, jeho přínos se nejvíce projevil až v nedávné době díky jeho effektivitě v případě trénování na velkém vzorku dat. Za předpokladu, že X je maticí o velikosti (n, p), můžeme cenu trénování vyjádřit jako $O(kn\bar{p})$, kde k označuje počet iterací a $\bar{p}$ je průměrný počet nenulových atributů každého prvku, z čehož vyplývá, že časová náročnost roste téměř lineárně s počtem vstupních vzorků. Jeho velkou výhodou je tedy vysoká účinnost a zároveň jednoduchost jeho implementace[1]. Další informace lze nalézt v článku [1].

Popis a definice

Při výpočtu SGD se snažíme minimalizovat sumu diferenciovatelných funcí, mající následující podobu:

$$Q(w) = \sum_{i=1}^{n} Q_i(w) \tag{3.22}$$

w je parametrem, který se snažíme zjistit a Q_i označuje funkci přiřazenou k jednotlivým prvkům vstupních dat a ∇ je notací pro gradient.

Při standartním „dávkovém“ Gradient Descent algoritmu jsou provedeny následující iterace:

$$w \leftarrow w - \alpha \sum_{i=1}^{n} \nabla Q_i(w) \tag{3.23}$$

Při Stochastic „on-line“ Gradient Descent je gradient $Q(w)$ aproximován gradientem v jediném stavu:

$$w \leftarrow w - \alpha \nabla Q_i(w) \tag{3.24}$$

V obou případech α označuje velikost kroku.

[1] Podrobnější informace o tomto algoritmu včetně jeho implementace v jazyce *Python* jsou dostupné na http://scikit-learn.sourceforge.net/modules/sgd.html

3.7 Regularizace

Základní informace

Regularizace je používána ve statistice a strojovém učení k zamezení jevu známému jako *overfitting* – natrénovaní „šumu“ do modelu, čímž se sice zvýší úspěšnost na trénovacích datech, ale při reálném využití úspěšnost naopak klesá. Tomu se snažíme zabránit pomocí snižování počtu rysů napojených na každý prvek modelu. Existuje několik variant regularizací a jejich aplikací, v našem případě jsou však využity pouze *L1-norm* a *L2-norm*. Více o srovnání těchto dvou v [8].

Popis a definice

V případě L1 se program snaží o snížení součtu absolutních hodnot parametrů, zatímco u L2 jde o snížení součtu čtverců hodnot parametrů. U L1 často dochází ke snižování až na hodnotu 0 a tím pádem vynechání z konečného modelu, je proto vhodnější za předpokladu, že mnoho rysů bude vhodnější vynechat.

Definice $R(\theta)$ je regularizační podmínkou a θ označuje parametr.

L1-norm

$$R(\theta) = ||\theta||_1 = \sum_{i=1}^{n} |\theta_i| \tag{3.25}$$

L2-norm

$$R(\theta) = ||\theta||_1 = \sum_{i=1}^{n} \theta_i^2 \tag{3.26}$$

Kapitola 4

Informační systém

Z důvodu velkého množství jednotlivých testů, různých nastavení a strojů, na kterých testování běží, jsem se rozhodl vytvořit za pomoci open-source CMS *Drupal* informační systém, který pomůže udržovat přehled o všech spuštěných testech a o jejich výsledcích. Systém byl vytvořen jako modul pro zmíněný CMS a využívá možností v té době dokončované verze 7.0. Pro vytvoření formulářů a jejich validaci implementuje Drupal Forms API a data ukládá do tabulek databáze dané stránky s využitím Drupal Database API.

4.1 Návrh systému

Pro systém byla navržena jednoduchá databáze, sestávající ze 4 tabulek a poskytující dostatek prostoru pro případná rozšíření o další rysy, nebo sledované výsledky. Její diagram tříd je zachycen na obrázku 4.1.

Aby systém co nejvíce ulehčoval práci, bylo nutné rozdělit proces vkládání dat na dvě části:

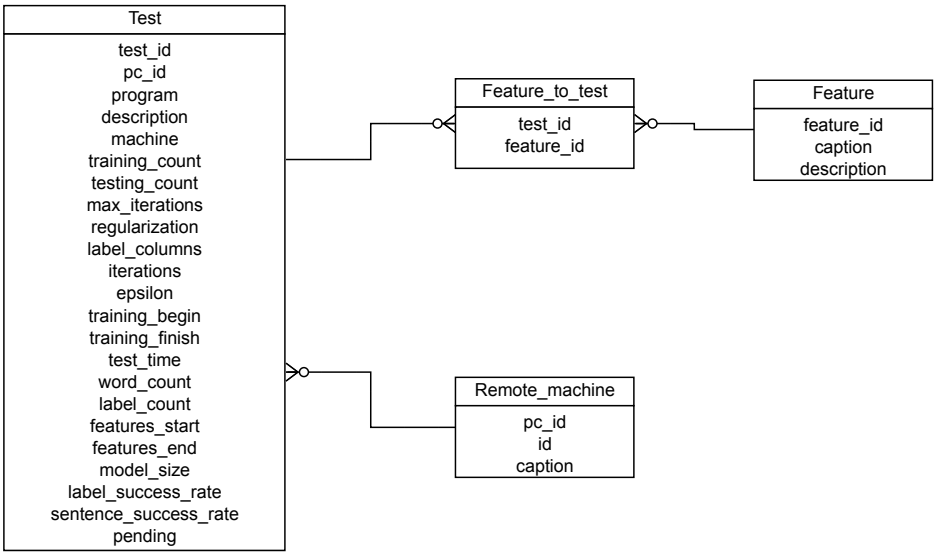
- informace, které zadáváme před započetím testování (nastavení programu, pravidel rysů, ...)
- informace, známé až po dokončení procesu (počet značek, výsledná úspěšnost, ...)

V databázi jsou tato data uložena pouze v jediné tabulce a proto bylo nutno přidat vlajku *pending* pro možnost zjištění, zda je daný test dokončen, či ne. Pro pohodlné zjištění dat pak bylo vytvořeno více pohledů na data, každý z nich poskytující jiný přístup.

Systém je tedy tvořen pěti stránkami, kde dvě z nich slouží k vkládání dat (*spuštění nového testu*, *dokončení běžícího testu*) a tři k zobrazení uložených dat (*zobrazení běžících testů*, *zobrazení dokončených testů* a *zobrazení náhledu na všechny testy a jejich rysy*). Každá tato část je reprezentována samostatnou stránkou, přístupnou v bloku systému drupal a tím pádem umístitelnou do kterékoliv části prezentace. Pro zpřehlednění prezentace a zvětšení použitelného prostoru byl tento blok vložen do patičky stránky.

4.2 Spuštění testu

Stránka je navržena jako jednoduchý formulář se vstupními poli a zaškrtávacími tlačítky, pro maximální urychlení práce. Před přidáním nového testu je nutno vyplnit následující informace (hodnoty v závorkách označují odpovídající sloupce v databázi v tabulce Test):



Obrázek 4.1: Návrh databáze informačního systému

- **Název použitého programu** (*program*) – výběr ze dvou položek (*CRFSuite* a *CRF++*) pro určení použitého programu
- **Identifikátor stroje** (*pc_id*) – název počítače, na kterém proběhl proces testování – výběr ze všech strojů uložených v databázi, na kterých v současné době není spuštěný žádný test (vyhledává se podle vlajky *pending* v databázi testů)
- **Popis testování** (*description*) – jakékoliv poznámky k testování (například zvolený algoritmus, který nebyl v návrhu systému brán v úvahu)
- **Typ regularizace** (*regularization*) – výběr ze dvou položek (*L1* a *L2*) pro využité regularizace
- **Počet trénovacích vět** (*training_count*) – celé číslo, specifikující počet vět obsažených v souboru, využítém při procesu trénování
- **Počet testovacích vět** (*testing_count*) – celé číslo, specifikující počet vět obsažených v souboru, využitém při procesu testování (značkování)
- **Maximální počet iterací** (*max_iterations*) – celé číslo, určující maximální počet iterací provedených při trénování
- **Delta** (*epsilon*) – číslo, specifikující hodnotu Δ, která je použita jako velikost práhu určeného k ukončení testování
- **Použité rysy** – rysy, využité při trénování modelu - pro vstup je použita skupina zaškrtávacích políček, včetně možnosti zaškrtnout všechna, nebo žádné (rysy nejsou

přímo uloženy v tabulce *Test*, ale jsou uloženy jako vztah mezi jednotlivými rysy v tabulce *Feature* a daným testem, pro jehož zachycení slouží tabulka *Feature_to_test*)

- **Použité části značky** (*label_columns*)– jednotlivé části značky, které jsou použity ve vstupních souborech, stejně jako u předchozího vstupního parametru jsou definovány skupinou zaškrtávacích políček, včetně možnosti zaškrtnout všechna, nebo žádné

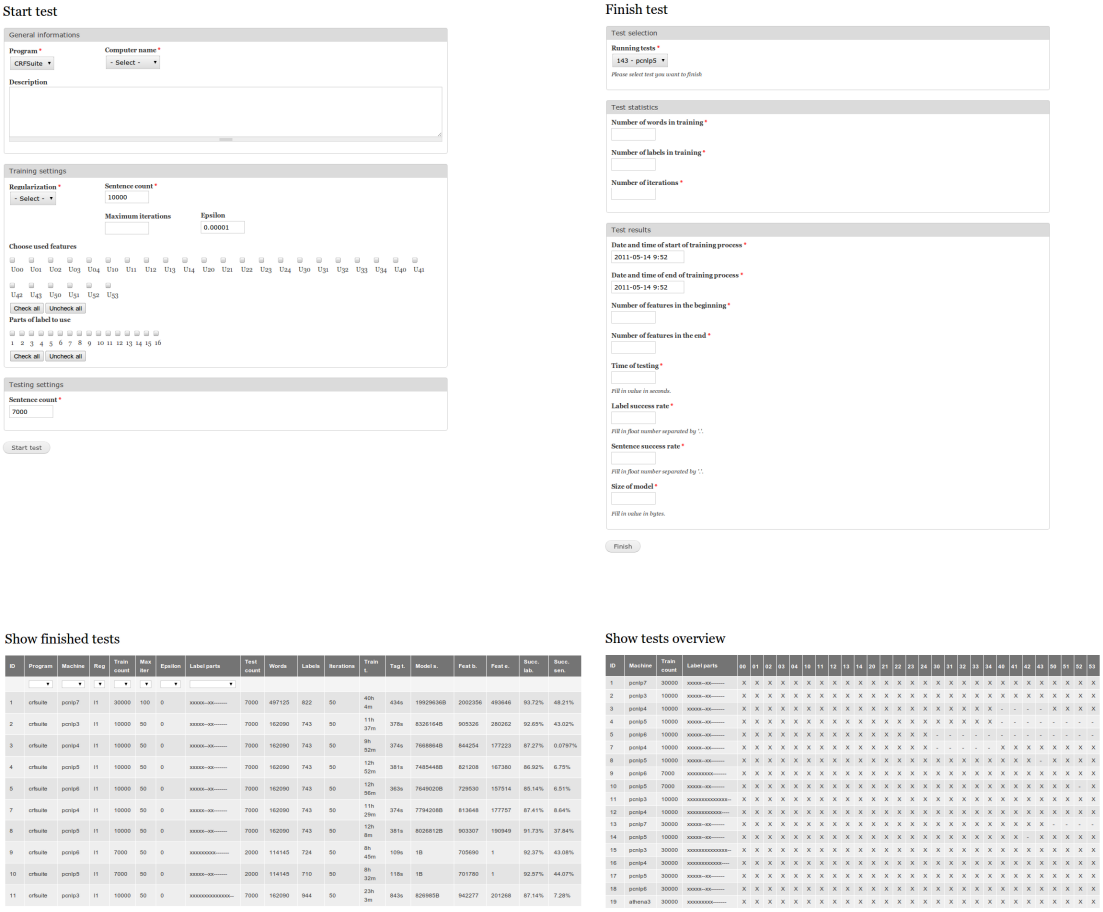
4.3 Dokončení testu

Při ukončení testu se poté dodají informace získané z logů a výstupních dat. Ty opět vyplníme do formuláře a odešleme jej na server. Mezi zjišťované informace patří:

- **Počet slov použitých při trénování** (*word_count*) – celkový počet slov, načtených programem při procesu testování
- **Počet unikátních značek v trénovacích datech** (*label_count*) – celkový počet unikátních značek
- **Počet iterací** (*iterations*) – počet trénovacích iterací, provedených před dosažením ukončující podmínky
- **Čas začátku a čas konce trénování** (*training_begin, training_finish*) – čas uvedený v logu, využitý k dopočtení celkové doby testování, která je programem chybně vypisována
- **Doba testování (značkování)** (*test_time*) – počet sekund, které trval proces značkování (jedna iterace algoritmu)
- **Počet rysů na začátku a na konci** (*features_start, features_end*) – celkový počet rysů obsažených v modelu na začátku a konci trénování
- **Velikost modelu** (*model_size*) – velikost souboru s modelem v bytech
- **Procentuální úspěšnost značkování u jednotlivých slov a u vět** (*label_success_rate, sentence_success_rate*) – konečný výsledek testování - poměr mezi celkovým počtem slov(vět) a počtem správně určených slov(vět)

4.4 Zobrazení běžících a skončených testů, zobrazení náhledu

Na těchto stránkách si můžeme prohlédnout výsledky, které můžeme později použít k dalším účelům. Výsledky jsou filtrovatelné podle většiny sloupců. V průběhu testování je velmi užitečná stránka zobrazení náhledu, která nám zobrazuje, které testy již byly provedeny, nebo právě běží, a dává nám tak jednoduchou možnost zanalyzovat další postup.



Obrázek 4.2: Snímky informačního systému

Kapitola 5

Prostředí testů a použitý software

5.1 Operační systém

Ačkoliv všechny využívané programy a skripty jsou multiplatformní, všechny testy byly spouštěny na operačním systému *Linux* nainstalovaném na univerzitních strojích, ke kterým jsem se vzdáleně připojoval přes ssh.

5.2 Hardware

Testování probíhalo průběžně až na 10 počítačích, kde tři znich jsou dostupné pro všeobecné využití (*athena1* – *athena3*) a zbylé jsou přístupné pouze pro výzkum zpracování jazyka (*pcnlp1* – *pcnlp8*, kromě *pcnlp2*, na kterém běží OS *Windows*). Z důvodu rozdílného hardware na jednotlivých strojích se při testování projevují velké skoky ve výsledných časech. V tabulce 5.1 je zobrazena stručná charakteristika jednotlivých strojů.

Název	CPU	Velikost Paměti
Athena 1	2 x Dual-Core AMD Opteron(tm) Processor 2220 @ 2,8GHz	16GB
Athena 2	2 x Dual-Core AMD Opteron(tm) Processor 2220 @ 2,8GHz	16GB
Athena 3	2 x Six-Core AMD Opteron(tm) Processor 2435 @ 2,6GHz	64GB
Pcnlp 1	Intel(R) Core(TM)2 Quad CPU Q6700 @ 2.66GHz	4GB
Pcnlp 3	Intel(R) Core(TM)2 Duo CPU E6750 @ 2.66GHz	2GB
Pcnlp 4	Intel(R) Core(TM)2 Duo CPU E6750 @ 2.66GHz	2GB
Pcnlp 5	Intel(R) Core(TM)2 Duo CPU E6750 @ 2.66GHz	2GB
Pcnlp 6	Intel(R) Core(TM)2 Duo CPU E6750 @ 2.66GHz	2GB
Pcnlp 7	Intel(R) Core(TM)2 Duo CPU E8400 @ 3.00GHz	2GB
Pcnlp 8	Intel(R) Core(TM)2 Duo CPU E8300 @ 2.83GHz	2GB

Tabulka 5.1: Charakteristika strojů použitých při zpracování testů

5.3 CRF++

Program *CRF++*[1] je open-source implementací modelu Conditional Random Fields, určenou pro značkování sekvenčních dat. Program je napsán v jazyce *C++* za použití *STL* knihovny[2]. K určení parametru používá algoritmu L-BFGS.

Nevýhodou programu je nekomentovaný zdrojový kód a program je proto nevhodný k jakýmkoliv úpravám. Při zpracování se také často vyskytl problém se zprávou *SIGSEGV*. Kód programu se zřejmě nebyl schopen vypořádat s náročnými vstupními daty a při použití všech kategorii a vyššího počtu vstupů zaslal systému výše zmíněnou zprávu a ukončil se.

5.3.1 Vlastnosti a nastavení

V tomto řešení jsou, na rozdíl od CRFSuite, přítomny tzv. bigramy, což je způsob dynamického generování rysů v průběhu testování, kdy aktuální prvek získá závislost na určené značce předchozího prvku. Pro využití této vlastnosti je však nutno vygenerovat rysy pro všechny kombinace značek a tím se do modelu přidá několik milionů rysů (čímž se razantně sníží rychlost). Pro českou značku tato vlastnost také nepřináší přílišné výhody, neboť značka je brána jako celek a nemůžeme se nijak odkazovat pouze na určité pozice (například na slovní druh). Toto je způsobeno optimalizacemi, kdy se vstupní řetězce převedou na číselnou hodnotu.

Program disponuje typickými nastaveními jako je omezení počtu iterací a určení ukončujících podmínek. Generování rysů je řešeno pomocí deklarací pravidel umístěných v konfiguračním souboru a veškeré úpravy vstupních dat je potřeba vykonávat externími prostředky. Do programu jsou pak vložena dvourozměrné pole s prvky oddělenými mezerou v první dimenzi a odřádkováním v druhé. Při testování je pak v posledním sloupci uložen správný výsledek, určený pro natrénování. Před předáním dat do programu byly také řetězce lemmat převedeny na čísla pomocí programu napsaného v jazyce *C*.

Generování vstupních dat

Nastavení v konfiguračním souboru jsou tvořena písmenem, určujícím, zda jde o unigram (vztahující se pouze k současnému prvku posloupnosti), či o bigram (výše zmíněný přechod mezi předchozím a aktuálním prvkem). Následuje dvouciferný identifikátor rysu, zajisťující unikátnost vůči ostatním definicím, dvojtečkou a řetězcem definujícím použitá vstupní data. K tomu je využíván zápis %[x,y], kde x označuje relativní pozici mezi slovy (0 označuje aktuální, záporné číslo předchozí a kladné následující prvek) a y označuje sloupec vstupních dat, který má být použit (číslovaný od 0).

Příklad Celé pravidlo může vypadat např. takto: *U01:%[-1,1]* a značí unigram s identifikačním číslem 01 a ze vstupního textu si vybere data na aktuální pozici z druhého sloupce. Pro ilustraci je v tabulce 5.2 zobrazen krátký, zjednodušený příklad vstupních dat. Pokud si určíme jako aktuální pozici slovo tenkrát, tedy $x = 2$ a použijeme výše zmíněné pravidlo, vyjde nám hodnota $x = 2 - 1 = 1$ a $y = 1$. Celý rys by pak vypadal takto: `U01:jsem`.

[1] Stránky programu: http://crfpp.sourceforge.net/

[2] Knihovna, přidávající do jazyka *C++* kontejnery, iterátory apod.

Id řádku	0: slovní tvar	1: možná lemmata	3: značka
0	Nespal	884/2119	VpYS---XR-NA---
1	jsem	152	VB-S---1P-AA---
2	tenkrát	564	Db----------
3	celou	1712/1284	AAFS4----1A----
4	noc	2122	NNFS4-----A----
5	.	124	Z:----------

Tabulka 5.2: Zjednodušený příklad vstupních dat programu CRF++

5.3.2 Popis procesu testování a výsledky

Jak bylo zmíněno v úvodu této kapitoly, program není schopen zjišťovat souvislosti mezi předchozím a aktuálním slovem na bázi jednotlivých částí značky. Proto byl celý proces testování rozdělen na několik kroků, ve kterých se postupně určí pouze některé části značky a ty budou buď využity v dalších krocích, nebo se všechny najednou využijí v jednom konečném kroku, určujícím kompletní značku. V prvním kroku proto bylo často použito slovního druhu a jeho detailu, v druhém rodu, osoby, pádu a ve třetím několika ze zbylých vlastností. Výsledky tohoto postupu však nebyly příliš uspokojující a proto od něj bylo upuštěno.

Pravidla pro generování rysů

Z důvodu několika-krokového charakteru testování se rysy velmi liší, veskrze jde však pouze o kombinace předchozích a aktuálních prvků vstupních souborů. Při generování vstupních souborů také byla provedena základní analýza a v případě, že slovní druh byl u všech možných značek konstantní, byl použit jako jeden z rysů. V opačném případě byl zapsán znak „-“. Dále byly použity několikanásobné rysy, skládající se zároveň ze dvou prvků (minulá a současná značka apod.).

Pro tyto pravidla mi povětšinou byly inspirací existující konfigurační soubory používané pro trénování anglického jazyka, které byly součástí příkladů použitých v programu *CRF++*. Pro tvoření rysů byl použit tvar slova, lemmata (složenina identifikátorů všech možných lemmat) a část značky určující slovní druh.

Spuštění a běh programu

Pro řízení procesu byl vytvořen skript v jazyce *BASH*. V nastavení tohoto skriptu můžeme určit hodnoty pro jednotlivé kroky zpracování (jako použité části značky, počet slov a pod.), skript pak v každém kroku pomocí programu v jazyce C vygeneruje aktuální data a předá je k natestování. Zároveň o všem vytváří záznamy a rozděluje jednotlivé výsledky do oddělených souborů. Na konci celého procesu pak pomocí AWK skriptu zkontroluje úspěšnost a uloží výsledky. Z důvodu stále se měnících vstupních dat je také nutno vytvořit několik oddělených souborů se vzory pravidel pro generování rysů. Ke spuštění celého procesu pak použijeme čistě řídící skript bez jakýchkoliv parametrů.

Id	Počet vět	Eta	Krok č.	Části značky	Počet příznaků	Úspěšnost
1	3000	0.05	1	1100000000000000	1 318 350	94.099%
			2	0010100100000000	2 354 463	59.105%
			3	0001000010000000	299 812	80.526%
2	3000	0.005	1	1100000000000000	2 065 745	92.487%
			2	0010100100000000	1 716 750	75.100%
			3	0001000110000000	897 138	87.055%
3	5000	0.005	1	1100000000000000	1 318 350	94.312%
			2	0010100000000000	1 097 024	71.303%
			3	0001000110000000	596 079	84.856%

Tabulka 5.3: Výsledky testování při využití programu *CRF++*

Výsledky testování

Testů bylo nakonec spuštěno několik desítek, většina z nich však nepřinášela kýžené výsledky. V tabulce 5.3 proto zahrnuji pouze vzorek dat s uspokojivými výsledky. Každý ze zachycených testů je rozdělen na 3 kroky, odlišující se využitými částmi značky. Ta je zapsána formou řetězce jedniček a nul, kdy jednička značí použití dané pozice a nula její vynechání. V posledním sloupci je pak zobrazena konečná úspěšnost procesu. V těchto testech bylo využito pouze malé množství vstupních dat a výsledky jsou převážně uspokojivé pouze při prvním kroku, využívajícím první dvě pozice značky. Přenesená informace v dalších krocích úspěšnost příliš nezvýšila a celkově výsledky nejsou využitelné. Z tohoto důvodu bylo od této testů upuštěno.

5.4 CRFSuite

Program *CRFSuite*[1] je open-source implementací algoritmu Conditional Random Fields, určený pro značkování sekvenčních dat. Je naprogramován v čistém C a dle jeho autora optimalizován pro co nejvyšší rychlost za obětování obecnosti. Jeho autorem je Ph.D Naoaki Okazaki, působící na univerzitě v Tokyu. Program ke svému chodu vyžaduje knihovnu libLBFGS[1] od stejného autora.

5.4.1 Vlastnosti a nastavení

Pro aproximaci parametru program implementuje dva algoritmy - L-BFGS (obsažený ve výše zmíněné knihovně) a SGD, který je dle poznámek inspirován implementací použitou v dalším programu pro strojové učení – *CRFSGD*[2]. Pro vyhlazení výsledků nabízí regularizace L1 a L2 v případě L-BFGS a pro SGD pouze L2. Program disponuje řadou upřesňujících nastavení, kde kromě standartních (podmínka ukončení testování, maximální počet iterací) lze nastavit i parametry jednotlivých algoritmů. Z pohledu zdrojového kódu a jeho čitelnosti je v tomto případě situace o mnoho lepší, než u *CRF++*, stále však platí omezení způsobená převodem do číselných konstant a tím pádem není možně průběžně kontrolovat předchozí označený prvek - jakékoliv úpravy by proto měly důrazný dopad na rychlost zpracování. Jak bude zmíněno v následující kapitole, určitými způsoby lze tento problém zmírnit.

Generování vstupních dat

CRFSuite neimplementuje žádný automatický přístup ke generování rysů, ale k programu je přikládán skript v jazyce *python*, který je schopen upravit vstupní data anglického korpusu do správného formátu. Pro potřeby tohoto této práce jsem jej přepsal tak, aby přijímal vstup ve formátu *CSTS*.

Skript postupně načítá data ze standartního vstupu a vytváří multidimenzionální pole pro pozdější zpracování. Při tomto kroku také provádí zpracování vstupu pro vylepšení kontextu trénování. Jelikož pro každé slovo máme ve vstupních datech určeny všechny dostupné značky, můžeme analyzovat jejich společnou část a tu využít jako jeden z rysů. Skript také kombinuje všechna použitá lemmata a analyzuje velikost, pozici ve větě, koncovky a předpony slov. Vnitřně jsou data na každém řádku uložena v následujícím tvaru: *tvar slova*, *společná část značky*, *všechna lemmata*, *přípona slova*, *správná značka*.

V dalším kroku program z uložených dat generuje podle zadaných pravidel (viz. tabulka 5.4) rysy a předává je na standartní výstup.

Příklad:

```
RR--7----------
U00=Konec U01=obchodníků U02=s U03=bílým U04=masem
U10=NNIS.-----A---- U11=NNMP2-----A---- U12=.....-----.---.
U13=AA...----1A---. U14=NNNS7-----A----
U20=/konec U21=/obchodník U22=/s-1/s-2‘sekunda_\:B/s-3_^(označení_pomocí_
písmene)/s-4_,t_^(saský_genitiv)/s-9_^(být_s_to)/společnost_\:B U23=/bílý
U24=/maso_^(jídlo_apod.)
```

[1]Stránky programu: http://www.chokkan.org/software/crfsuite/
[1]Stránky knihovny: http://www.chokkan.org/software/liblbfgs/
[2]Algoritmus SGD, jeho implementace a příklady jejího použití: http://leon.bottou.org/projects/sgd

```
U30=ec U31=ků U32= U33=ým U34=em
U40=s U41=s U42=s U43=s
U50=2 U51=2 U52=1 U53=
```

Na začátku řádku se nachází výsledná značka, následovaná jednotlivými rysy. Všechny rysy jsou tak vytvořeny před samotným spuštěním testovacího programu a v průběhu testování se již nemění – mohou být vypuštěny jen před samotným začátkem (kvůli limitu minimálního počtu výskytů), nebo při vykonávání regularizace.

5.4.2 Popis procesu testování a výsledky

V případě tohoto programu nebyly využity žádné metody rozdělování průběhu do jednotlivých kroků, všechny testy jsou spouštěny jako jeden hlavní proces, který nejdříve provede trénování a pak výsledný model otestuje.

Použití části značky

Z důvodu ušetření času a teoretické jednodušší možnosti natrénování (odvozování pravidel od menšího počtu celkových značek a tím pádem vyššího počtu podobných značek, poskytujícím vyšší pravděpodobnost ovlivnění modelu) byly při testování převážně spouštěny testy nekompletní značkou - pro zjednodušení byly využity pouze nejdůležitější mluvnické kategorie. Toto zjednodušení nám dovoluje provést testy v nižším čase a se všemi pravidly, které by byly použity i v případě kompletní značky. Pro reálé nasazení by pak bylo nutno zjistit, zda jsou všechny kategorie potřebné, nebo zda stačí tento omezený počet prvků. Vliv pravidel na výkon by však neměl být ovlivněn, neboť nevyužívají žádnou z vynechaných značek.

Pravidla pro generování rysů

V této práci jsou využívány klasické rysy, které se používají prakticky ve všech typických příkladech využití strojového učení (předchozí prvek, následující prvek,...), ale i rysy speciální, které mají smysl pouze pro značkování textu, či dokonce pouze pro český jazyk (výčet všech rysů viz. tabulka 5.4). Všechny rysy dodržují označení, používané ve většině podobných programů, popsané v kapitole 5.3.2, pouze v případě tohoto programu nejsou implementovány bigramy a rysy jsou generovány před spuštěním programu.

ID	Popis
U00	Slovní tvar na pozici x-2
U01	Slovní tvar na pozici x-1
U02	Slovní tvar na pozici x
U03	Slovní tvar na pozici x+1
U04	Slovní tvar na pozici x+2
U10	Značka (společná část) na pozici x-2
U11	Značka (společná část) na pozici x-1
U12	Značka (společná část) na pozici x
U13	Značka (společná část) na pozici x+1
U14	Značka (společná část) na pozici x+2
U20	Možná lemmata na pozici x-2
U21	Možná lemmata na pozici x-1
U22	Možná lemmata na pozici x
U23	Možná lemmata na pozici x+1
U24	Možná lemmata na pozici x+2
U30	Přípona na pozici x-2
U31	Přípona na pozici x-1
U32	Přípona na pozici x
U33	Přípona na pozici x+1
U34	Přípona na pozici x+2
U40	Předpona – první písmeno slova
U41	Předpona – první dvě písmena slova
U42	Předpona – první tři písmena slova
U43	Koncovka – poslední písmeno slova
U50	Velikost písmen – všechny malé, všechny velké, první velké
U51	Číslo, označující pozici ve větě
U52	Délka slova
U53	Značka nejbližšího slovesa (vyhledáváno dopředu i dozadu)

Tabulka 5.4: Seznam rysů použitých při testování programem *CRFSuite*

Spuštění a běh programu

Pro testování byl vytvořen řídící skript v jazyce BASH, starající se o celý průběh testu (viz. obrázek 5.1) a ulehčující nastavení celé procedury a následné dohledání použitých parametrů. V hlavičce jsou definovány základní údaje a nastavení, které jsou poté použity v těle pro postupné spouštění jednotlivých příkazů. Před započetím celého procesu jsou nejdříve vymazány soubory předchozího zpracování. V průběhu operace jsou logovány všechny důležité údaje. Tento skript je naklonován pro každý využívaný počítač a tím je poskytnuta možnost zpětné kontroly nastavených údajů a vyhnutí se chyb spojených s prací na více strojích najednou. Stejný přístup byl aplikován i na skript, generující data z *CSTS* souboru.

S použitím vytvořených nástrojů je spuštění testování jednoduchou a rychlou záležitostí. Nejdříve je v konverzním skriptu (napsaném v jazyce *Python*) souboru (pojmenovaném jako `[jméno stroje].py`) nastaveno, které rysy mají být použity. V dalším kroku se v bash scriptu ([jméno stroje].sh) nastaví počet vět pro trénování a testování, algoritmus, maximální počet iterací, epsilon, regularizace a další případné parametry. Dále je nutno upravit pomocí programu ulimit limity zdrojů, dostupných pro náš program – jedná se konkrétně o maximální velikost zásobníku, maximální přidělený čas CPU a maximální velikost virtuální paměti. Ke spouštění skriptu je použit program *nohup*, zajišťující přetrvávající běh programu i po odpojení se od vzdáleného počítače.

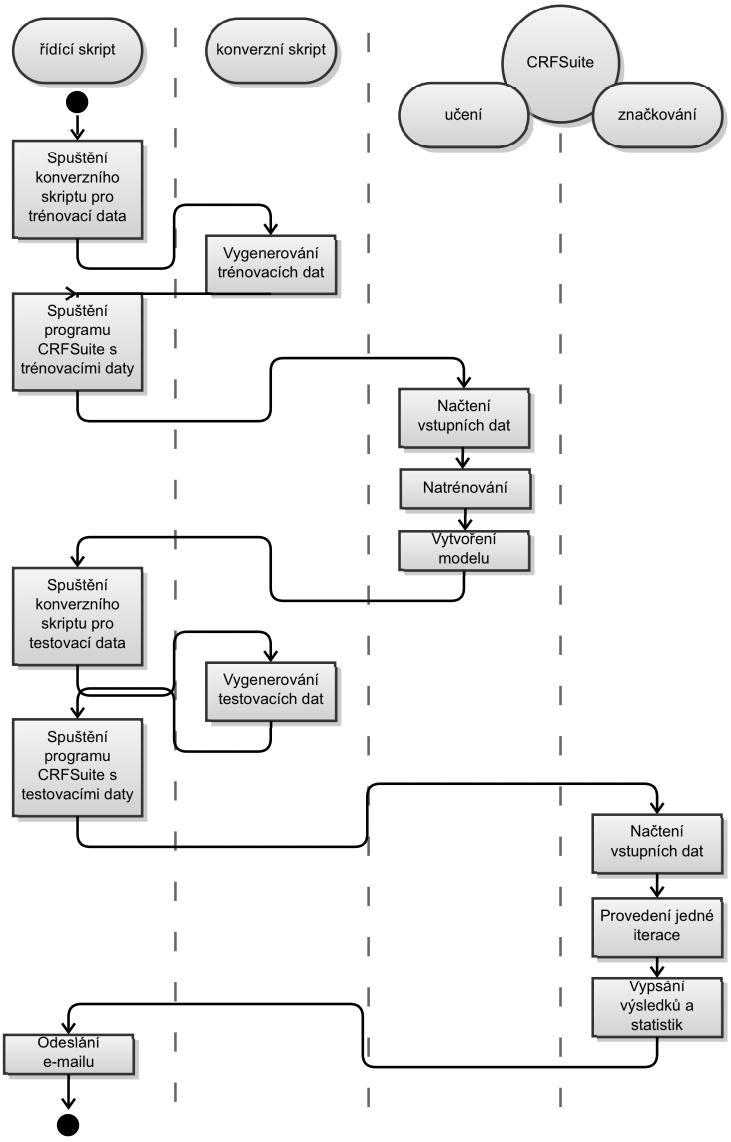
Příklad:

```
$ nohup ./[název stroje].sh \&
$ nohup ./pcnlp3.sh \&
```

Po dokončení procesu skript odešle email na nastavenou emailovou adresu a ve složce přibudou tři soubory: `[název stroje].log` s údaji o trénování, `[název stroje].result` s údaji o výsledku trénování a `[název stroje].model` s výsledným modelem.

Výsledky testování

Kvůli rozsáhlosti testů provedených s tímto programem byly všechny výsledky přesunuty do samostatné kapitoly 6.



Obrázek 5.1: Diagram průběhu testování

Kapitola 6

Testy

Před započatím řádného testování a analýzy výsledků bylo potřeba se rozhodnout, s jakými parametry bude testování prováděno. Všechny následující testy jsou provedeny v programu *CRFSuite*, který, jak je zmíněno výše, pro tuto práci přináší vhodnější prostředí. V rámci této práce bude většina testů provedena s regularizací L1 i L2 (viz. podkapitola 3.7) a výsledky budou mezi sebou porovnány. Testy jsou převážně provedáděny s využitím algoritmu L-BFGS (viz. podkapitola 3.5), v průběhu práce však byl pro několik testů použit algoritmus SGD (viz. podkapitola 3.6), který poskytl kvalitnější výsledky při nižší časové náročnosti. Na provedení komplexního srovnání a otestování všech vynechaných rysů však již nevyšel čas. Do práce je tak přidáno krátké srovnání výsledků obou zmíněných algoritmů a závěry, zjištěné pro L-BFGS, budou pokusně aplikovány i na SGD.

Další volbou byl způsob ukončení trénování – nabízely se dvě možnosti: Buď ukončení po určitém počtu iterací, nebo při dosažení práhu rozdílu celkové pravděpodobnosti mezi dvěmi po sobě jdoucími iteracemi pod úroveň konstanty Δ. Pro testování váhy jednotlivých pravidel generování rysů však byla využita druhá varianta (Δ), která by měla poskytnout vyšší variabilitu a tím možnost správně využít aktuálního nastavení (což bylo později potvrzeno mnohem vyšším počtem iterací při regularizaci L2 namísto L1 a výkyvy mezi počtem iterací v případě vynechání některých pravidel) a je také výchozím nastavením programu.

Problematickým se ukázal být počet testovacích vzorků, kde při použití více než 7000 vzorků program *CRFSuite* z nezjištěných důvodů odmítl vydat jakékoliv výsledky a zřejmě se zacyklil při vnitřní chybě. Proces stále visel v systému, nejevil však žádné známky aktivity a proto musel být ručně ukončen. Pro přesnější měření by tedy zřejmě bylo výhodnější výsledky ještě validovat oproti dalším vzorkům dat, pro základní porovnání výkonnostního přínosu jednotlivých rysů by však měl být tento počet naprosto dostačujícím.

Jak bylo zmíněno v kapitole 5, doba trénování neroste lineárně, neboť pokusy byly prováděny na různých strojích o různém výpočetním výkonu. Celková doba procesu testování je však ve všech případech velmi podobná a je závislá téměř čistě na výkonu procesoru a počtu vstupních prvků (počet rysů, ani počet iterací trénování ji neovlivní). Ve všech testech se pohybovala v rozmezí od 300 do 1000 sekund a pro ušetření místa v tabulkách nebude zahrnuta. Tato hodnota je však zobrazena v příloze v kompletním seznamu všech testů.

6.1 Popis sloupců v tabulkách

V této podkapitole budou popsány jednotlivé nadpisy použité v hlavičkách tabulek, zobrazujících výsledky a srovnání testů. V případě šetřením místem byly občas některá slova zkrácena, jejich význam by však měl zůstat pochopitelný. Ve sloupcích, popisujících jakoukoliv změnu jsou referenční hodnoty zobrazeny v závorce hned pod nadpisem sloupce.

- **Počet vět** - počet ucelených sekvencí vstupních dat, použitých při testování
- **Počet iterací** - počet iterací, provedených při testování
- **Počet značek** - celkový počet unikátních značek ve vstupních datech
- **Doba trénování** - čas potřebný k natrénování modelu
- **Velikost modelu** - velikost výsledného modelu
- **Úvodní počet rysů** - množství rysů na začátku trénování
- **Konečný počet rysů** - množství rysů na konci trénování
- **Počet rysů** - v případě, kdy je úvodní i konečný počet rysů totožný (nebo je rozdíl velmi malý), je zapsána pouze jediná hodnota
- **Úspěšnost slov** - poměr mezi počtem správně určených slov a celkovým počtem slov obsaženým v testovacích datech
- **Úspěšnost vět** - poměr mezi počtem celých správných vět (kde všechna slova byla určena správně) a celkovým počtem vstupních vět
- **Popis pravidla** - slovní popis daného pravidla, použitý pro lepší pochopení tabulek bez nutnosti nahlížet do definicí jednotlivých pravidel
- **Změna počtu rysů** - rozdíl v počtu rysů oproti referenční hodnotě (při vynechání pouze jednoho pravidla odpovídá počtu rysů, generovaných daným pravidlem)
- **Změna konečného počtu rysů** - rozdíl v počtu rysů na konci testování, odvíjející se od referenční hodnoty (kladná hodnota značí přírůstek, záporná úbytek)
- **Změna velikosti modelu** - rozdíl mezi referenční a aktuální velikostí modelu (kladná hodnota značí zvětšení souboru, záporná pak zmenšení)
- **Změna úspěšnosti slov** - rozdíl mezi referenční a aktuální úspěšností jednotlivých slov (kladná hodnota značí zlepšení, záporná zhoršení)
- **Změna úspěšnosti vět** - rozdíl mezi referenční a aktuální úspěšností celých vět (kladná hodnota značí zlepšení, záporná zhoršení)

6.2 Závislost úspěšnosti na počtu vstupních vět

Následující testy se zabývají analýzou ideálního počtu vět pro použití v dalších částech práce. Je nutno získat takovou hodnotu, která bude poskytovat jednak kvalitní, jednak časově akceptovatelné výsledky, které poskytnou možnost zpracovat všechny časově náročné testy na daném počtu strojů.

6.2.1 Konstantní počet iterací a regularizace L1

Příprava

Nejdříve bylo třeba rozhodnout, jaký počet iterací bude vhodný pro proces testování. Po několika pokusných testech se jako nejlepší ukázalo číslo 50. Okolo této hodnoty program již začal půlit kroky algoritmu pro vyšší přesnost a tím se exponencionálně zvyšovala náročnost každého dalšího kroku aproximace parametru. K dokončení 60. iterace proto čas vzrostl přibližně 3krát, zatímco úspěšnost se zvýšila pouze minimálně.

Konstantní podmínky

Algoritmus: L-BFGS
Maximální počet iterací: 50
Části značky: 1 – 5 a 8 – 9
Počet vět při testování úspěšnosti: 7000
Regularizace: L1

Vyhodnocení

Jak lze vidět v tabulce 6.1, při prvních dvou testech (5000 a 10 000 vstupních vět) a také v předposledním testu (55000 vět) se s největší pravděpodobností vyskytla „chyba“ v programu *CRFSuite*, projevující se nezvykle vysokými hodnotami úspěšnosti. Ani po přetestování[1] na odlišných strojích se však výsledky nezměnily. Tyto abnormální hodnoty se později vyskytují i v dalších testech.

Nebýt třech podivných výsledků, je jasně vidět stoupající tendence testů až k 50 000 vět. Od této úrovně už se poté úspěšnost spíše snižuje – což může být způsobeno omezením maximálního počtu iterací na 50.

Z rozdílu mezi počtem úvodních a konečných rysů lze dobře pozorovat přínos využité regularizace, odmazávající nevýrazné rysy. Konečný počet je tak v průměru 4 krát nížší, čímž se i snižuje velikost výsledného modelu.

[1] Pro ověření byly zvovu provedeny testy s následujícím počtem vět: 5 000, 10 000, 55 000, 50 000

Počet vět	Počet značek	Doba trénování	Velikost modelu	Úvodní počet rysů	Konečný počet rysů	Úsp. slov	Úsp. vět
5000	671	5h 30m	4 751 KB	558871	113149	91.04%	34.70%
10000	743	20h 47m	8 326 KB	905326	200613	92.65%	43.02%
15000	769	23h 46m	10 990 KB	1202680	266245	87.90%	7.73%
20000	792	33h 41m	13 706 KB	1464587	334939	88.37%	8.07%
25000	811	45h 25m	17 140 KB	1724528	423649	88.68%	8.20%
30000	822	37h 56m	18 953 KB	1968646	461337	88.71%	7.99%
35000	833	56h 44m	22 744 KB	2260670	564590	89.38%	10.26%
40000	843	66h 54m	23 583 KB	2488724	586499	90.12%	15.69%
45000	849	128h 26m	27 177 KB	2716001	680082	90.62%	18.99%
50000	854	199h 44m	28 602 KB	2924981	717070	89.45%	6.85%
55000	867	90h 22m	30 087 KB	3140944	758370	94.74%	53.77%
60000	870	111h 35m	34 118 KB	3329504	863220	89.63%	8.05%

Tabulka 6.1: Výsledky testů při regularizaci L1 a konstantním počtu iterací

6.2.2 Konstantní počet iterací a regularizace L2

Příprava

V tomto případě jsem využil stejných pravidel, jako u minulého testu, pouze jsem změnil regularizaci pro získání výsledků k porovnání. Jediným rozdílem je velikost kroku mezi jednotlivými testy, která byla pro urychlení zdvojnásobena na 10 000 vět.

Konstantní podmínky

Algoritmus: L-BFGS
Maximální počet iterací: 50
Části značky: 1 – 5 a 8 – 9
Počet vět při testování úspěšnosti: 7000
Regularizace: L2

Vyhodnocení

Jak je popsáno v podkapitole 3.7, při regularizaci L2 nedochází k tak výraznému úbytku konečných rysů (v tomto případě nulovému, v tabulkách s L2 je proto zobrazován pouze jeden sloupecs počtem rysů) a tím pádem by konečný výsledek měl přinést vyšší úspěšnost. Jak lze vidět v tabulce 6.2, úspěšnost roste bez větších výkyvů od 10 až po 70 tisíc vstupních vět, se zvyšujícím se počtem rysů se rozdíl mezi jednotlivými kroky však snižuje. Úspěšnost se ve všech případech pohybuje nad hranicí 90 procent a tento způsob zpracování tak poskytuje velmi uspokojivé výsledky.

Počet vět	Počet značek	Doba trénování	Velikost modelu	Počet rysů	Úsp. slov	Úsp. vět
10000	743	15h 56m	36 244 KB	905326	91.97%	40.64%
20000	792	44h 30m	57 072 KB	1464587	92.78%	45.28%
30000	822	42h 26m	76 697 KB	2002356	93.51%	48.41%
40000	843	84h 50m	94 190 KB	2488724	93.61%	49.06%
50000	854	128h 19m	109 663 KB	2924981	93.83%	50.26%
60000	870	181h 14m	123 654 KB	3329504	94.04%	51.09%
70000	881	191h 8m	136 263 KB	3697048	94.05%	51.52%

Tabulka 6.2: Výsledky testů při regularizaci L2 a konstantním počtu iterací

6.2.3 Konstantní počet iterací - srovnání mezi L1 a L2

U L2 je v porovnání s regularizací L1 úspěšnost (tabulka 6.4) úspěšnost v rámci jednotlivých slov vyšší o několik procent a u celých vět je rozdíl dokonce několik desítek procent (nepočítáme-li výsledky způsobené anomálií vyskytující se při některých výsledcích u L1). Toto zlepšení je však vykoupeno skutečností, že velikost modelu (tabulka 6.3) je v průměru 4krát vyšší a doba testování ze zvýšila přibližně o polovinu. Je tedy otázkou, zda je pro nás tato skutečnost akceptovatelná a zda nám stojí za zvýšení úspěšnosti o několik procent, nebo zda dáme přednost malému modelu a rychlejšímu zpracování – to už však záleží na konkrétním využití programu. Překvapivě se však nijak nezvýšila doba trénování.

Ze srovnání lze také vypozorovat, že nekonsistentní výsledky z testu s regularizací L1 se velmi přibližují výsledkům s regularizací L2.

Počet vět	Doba trénování		Doba testování		Konečný počet rysů	
	L1	L2	L1	L2	L1	L2
10000	20h 47m	15h 56m	646s	595s	200 613	905 326
20000	33h 41m	44h 30m	527s	935s	334 939	1 464 587
30000	37h 56m	42h 26m	421s	584s	461 337	2 002 356
40000	66h 54m	84h 50m	604s	555s	586 499	2 488 724
50000	199h 44m	128h 19m	738s	1516s	717 070	2 924 981
60000	111h 35m	181h 14m	696s	1913s	863 220	3 329 504

Tabulka 6.3: Srovnání mezi L1 a L2 při konstantním počtu iterací (výsledky)

6.2.4 Regularizace L1 a ukončení pomocí dosažení práhu Δ

Příprava

Nejdříve byla uskutečněna skupina testů se snižující se hodnotou Δ od 0.1 po 0.0001, po spuštění testů a dokončení pouze dvou z nich, kdy od hodnoty 0.001 již byla podmínka ukončení příliš náročná a proces testování by trval řádově měsíce, jsem se rozhodl od tohoto testu upustit. Zároveň hodnota 0.1 poskytovala kvalitní výsledky v rozumném čase a počet iterací okolo 50, což nám nabízí rozumné srovnání s předchozím způsobem testování.

Konstantní podmínky

Algoritmus: L-BFGS
Δ*:* 0.1
Části značky: 1 – 5 a 8 – 9
Počet vět při testování úspěšnosti: 7000
Regularizace: L1

Vyhodnocení

Jak jde vidět v tabulce 6.5, opět se projevila anomálie při regularizaci L1, kdy 2 z pěti výsledků mají neúměrnou úspěšnost. Ze zbylých tří testů lze pak vidět existující stoupající tendenci v úspěšnosti dle jednotlivých slov, výsledek je ale z důvodu chyb velmi neprůkazný.

Počet vět	L1 velikost modelu	L2 velikost modelu	L1 úsp. slov	L2 úsp. slov	L1 úsp. vět	L2 úsp. vět
10000	8 326 164B	36 244 KB	92.65%	91.97%	43.02%	40.64%
20000	1 3706 360B	57 072 KB	88.37%	92.78%	8.07%	45.28%
30000	18 953 540B	76 697 KB	88.71%	93.51%	7.99%	48.41%
40000	23 583 752B	94 190 KB	90.12%	93.61%	15.69%	49.06%
50000	28 602 144B	109 663 KB	89.45%	93.83%	6.85%	50.26%
60000	34 118 604B	123 654 KB	89.63%	94.04%	8.05%	51.09%

Tabulka 6.4: Srovnání mezi L1 a L2 při konstantním počtu iterací (data)

Počet vět	Počet iterací	Doba trénování	Velikost modelu	Úvodní počet rysů	Konečný počet rysů	Úsp. slov	Úsp. vět
10000	47	19h 20m	8 722 KB	905326	209912	92.61%	43.12%
20000	50	58h 54m	13 706 KB	1464587	334939	88.37%	8.07%
30000	51	107h 55m	19 722 KB	2002356	488244	93.50%	46.25%
40000	55	155h 57m	22 254 KB	2488724	551872	89.67%	11.26%
50000	53	110h 58m	27 773 KB	2924981	696340	89.42%	6.79%

Tabulka 6.5: Výsledky testů při regularizaci L1 a konstantním počtu iterací

6.2.5 Algoritmus SGD

Příprava

Algoritmus Stochastic Gradient Descent, popsaný v podkapitole 3.6) bohužel většinu času stál mimo hlavní zájem. Po jeho vyzkoušení však byly kvůli jeho kvalitám všechny konečné testy spouštěny i s tímto algoritmem. Nejdříve byly provedeny tři testy pro zjištění vhodné hodnoty Δ a poté 5 testů s počtem vstupních vět pohybujícím se mezi 10 a 50 tisíci vzorků, aby mohly získané výsledky být porovnány se testy provedenými za pomoci algoritmu L-BFGS. Použitý program *CRFSuite* podporuje v případě použití tohoto algoritmu pouze regularizaci L2 a není tedy nutno zrcadlit testy na jakékoliv jiné varianty.

Konstantní podmínky

Algoritmus: SGD
Části značky: 1 – 5 a 8 – 9
Počet vět při testování úspěšnosti: 7000
Regularizace: L2

Vyhodnocení

Jako při většině ostatních testů, i v tomto případě se zvyšováním přesnosti a počtu vstupních vzorků se plynule zvyšuje i úspěšnost. Jak lze vidět v tabulce 6.6, tento algoritmus pro určení parametrů je vůči ostatním testovaným nastavením mnohem rychlejší a poskytuje kvalitnější výsledky, kdy se při 50 000 větách dostáváme až na 94,31% a při větším počtu vstupních prvků, nebo v kombinaci s vyšší přesností, porovnané v tabulce 6.7, by tak nejspíše byla pokořena hranice 95%. Při porovnání s hodnotami získanými testováním s algoritmem L-BFGS je tento až 3krát rychlejší a úspěšnost je přitom vyšší o 0.5% (při 30 000 vstupních větách a deltě 0.1).

K testu zkoumajícímu ideální hodnotu Δ je pak vhodné poznamenat, že stejné ukončení pro 0.01 i 0.001 je způsobeno velmi nízkým rozdílem v 57. iteraci, který splnil podmínku ukončení pro obě sledované varianty.

Počet vět	Počet iterací	Doba trénování	Velikost modelu	Počet rysů	Úsp. slov	Úsp. vět
10000	27	8h 37m	36 244 KB	905 326	92.34%	42.85%
20000	29	21h 40m	57 072 KB	1 464 587	93.33%	48.67%
30000	35	30h 13m	76 696 KB	2 002 349	93.96%	51.60%
40000	36	41h 25m	94 190 KB	2 488 724	94.31%	53.45%

Tabulka 6.6: Závislost úspěšnosti na počtu vstupních vět při algoritmu SGD a regularizaci L2

Počet iterací	Doba trénování	Velikost modelu	Počet rysů	Úsp. slov	Úsp. vět
35	30h 13m	76 696 KB	2 002 349	93.96%	51.60%
57	48h 38m	76 696 KB	2 002 356	94.01%	52.16%
57	52h 10m	76 696 KB	2 002 350	94.01%	52.16%

Tabulka 6.7: Závislost úspěšnosti na hodnotě Δ při algoritmu SGD, regularizaci L2 a konstantním počtu vět (30 000)

6.3 Porovnávání úspěšnosti při vynechání určitých pravidel rysů

Po zhodnocení předchozích testů jsem připravil pravidla pro nejdůležitější část, kterou je testování úspěšnosti při vynechání určitých pravidel rysů (zjišťování váhy pravidla na konečný výsledek). Jak bylo zmíněno výše, všechny následující testy budou pracovat s ukončovací podmínkou určenou dosažením prahu Δ. V tomto kroku bude provedeno testování při postupném vynechávání všech pravidel a bude zanalyzován jejich přínos pro úspěšnost určení. Do testování je zahrnut i test bez vynechaného rysu, který bude sloužit jako referenční a k jeho hodnotám pak budou vztahovány jednotlivé výsledky. Nejméně výhodné příznaky budou vypuštěny a celý model bude otestován bez nich. Získaný výsledek bude porovnán s původní hodnotou a dle tohoto výsledku bude určena úspěšnost tohoto přístupu.

6.3.1 Regularizace L1

Příprava

Před započetím testování s regularizací L1 bylo nutno zvážit přínos, který této práci, s ohledem na problémy s anomáliemi, přinese. Pro úplnost však byly tyto testy provedeny a tím byla poskytnuta možnost komplexního srovnání mezi oběmi regularizacemi.

Konstantní podmínky

Algoritmus: L-BFGS
Δ*:* 0,1
Části značky: 1 – 5 a 8 – 9
Počet vět při testování úspěšnosti: 7000
Regularizace: L1

Vyhodnocení

Při pohledu do tabulky 6.8 lze spatřit, že referenční test bohužel produkuje abnormální (chybné) výsledky a tím komplikuje celý proces určení hodnoty jednotlivých rysů. Při zanedbání chybných testů lze vidět, že úspěšnost určení jednotlivých slov se pohybuje v rámci jednoho procenta a žádný z rysů tedy nemá na konečný výsledek rozhodující vliv.

Pro lepší srovnání byla vytvořena tabulka 6.9, která zachycuje rozdíl mezi úspěšností při vynechání daného rysu a referenční hodnotou. Ta je zapsána pod hlavičkou tabulky a ve většině případů je použita výsledná hodnota referenčního výsledku bez vynechaných pravidel. Vyjímku tvoří poslední sloupec, kde je použito číslo 89, které nejlépe odděluje užitečné příznaky od těch neužitečných. Jedná se však pouze o odhad a proto je možné, že neužitečných pravidel je ve skutečnosti mnohem více, či jsou dokonce užitečné všechny. Pro větší názornost byl do tabulky přidán sloupec s popisem jednotlivých pravidel generování rysů.

Z tabulky lze vyčíst několik údajů. Jednak počet rysů se zvyšuje se vzdáleností od aktuálního prvku, což je logické, když se zamyslíme nad způsobem zpracování programem. Každá unikátní vlastnost je vztahována k současnému prvku. Při určení aktuálního prvku tedy generujeme pouze 1 unikátní vlastnost, v kombinaci s prvky ležícími v sousedství však musíme vždy uložit kombinaci daného prvku s prvkem aktuálním.

Dále se na rysy můžeme podívat z pohledu paměťové náročnosti, kde několika sty tisíci unikátními vlastnostmi jasně dominují možná lemmata a slovní tvary, které se tak stávají kandidátem na vypuštění z modelu v případě potřeby snížení paměťové náročnosti trénování (za předpokladu, že by tím nebyla výrazně narušena úspěšnost trénování). To také dokazuje výsledná velikost modelu, kde k největšímu snížení došlo u možných lemmat a slovního tvaru na aktuální pozici a to přes 3MB. Naproti tomu při vynechání společné značky na aktuální pozici se model ještě o 3MB zvětšil.

Tučně vyznačené záznamy v tabulce jsou ty, u kterých se projevila anomálie regularizace L1 - nemají tedy žádnou váhu při analýze a nebudou zařazeny do vybírání nejméně užitečných prvků. Pro detailnější a správnější zpracování by bylo nutno provést celý test ještě jednou na jiném vzorku dat a poté pracovat pouze se správnou hodnotou, resp. s průměrem v případě nevyskytující se chyby v obou případech pro dané pravidlo. Na toto zpracování se však nedostávalo času a bylo nutno jej vynechat a přesunout se k dalším testům, které mohou přinést kvalitnější a cennější výsledky.

Vyn. rys	Počet iterací	Doba trénování	Velikost modelu	Úvodní počet rysů	Konečný počet rysů	Úsp. slov	Úsp. vět
-	51	56h 53m	19 722 KB	2002356	488244	93.50%	46.25%
U00	52	66h 26m	20 448 KB	1795523	517260	88.94%	8.54%
U01	53	54h 50m	17 981 KB	1820650	448183	94.15%	51.47%
U02	51	87h 46m	16 350 KB	1908841	415861	92.30%	35.52%
U03	54	70h 7m	16 607 KB	1827303	414097	88.98%	8.16%
U04	52	83h 49m	16 794 KB	1797529	417891	89.26%	10.58%
U10	52	75h 0m	17 564 KB	1973324	427379	94.02%	50.21%
U11	52	82h 55m	19 037 KB	1982074	549159	88.95%	8.37%
U12	55	77h 2m	23 358 KB	2000160	549160	91.75%	38.07%
U13	52	113h 12m	17 389 KB	1982580	426650	88.91%	7.67%
U14	52	58h 7m	17 847 KB	1973946	434700	88.93%	8.03%
U20	50	83h 45m	18 678 KB	1836466	464943	88.89%	8.59%
U21	52	77h 52m	17 532 KB	1859057	434576	93.36%	44.68%
U22	53	87h 11m	16 573 KB	1919326	419928	88.95%	7.81%
U23	55	52h 54m	16 751 KB	1851001	413535	89.10%	8.59%
U24	52	83h 25m	17 933 KB	1828280	446111	88.99%	8.31%
U30	50	50h 15m	19 175 KB	1958718	466492	89.08%	8.83%
U31	51	121h 18m	19 211 KB	1969486	469025	88.90%	8.26%
U32	53	82h 12m	19 290 KB	1995508	474503	88.82%	8.00%
U33	51	78h 14m	17 678 KB	1969100	431720	89.37%	11.19%
U34	51	103h 53m	19 382 KB	1959411	475058	89.01%	8.29%
U40	52	86h 0m	19 246 KB	1998814	449381	88.97%	7.78%
U41	53	104h 4m	19 012 KB	1969647	463998	89.06%	8.41%
U42	53	77h 5m	18 186 KB	1938959	444168	88.99%	8.23%
U43	54	120h 38m	18 580 KB	1999796	455661	88.88%	8.37%
U50	51	80h 57m	17 576 KB	2000853	431772	88.96%	8.04%
U51	53	52h 39m	17 419 KB	1983793	424259	92.61%	37.42%
U52	52	95h 39m	18 321 KB	1998814	449381	88.97%	7.78%
U53	53	71h 45m	18 361 KB	1992254	450316	88.72%	8.16%

Tabulka 6.8: Výsledky testů vynechání příznaků při regularizaci L1

ID	Popis pravidla	Zm. poč. rysů	Zm. kon. poč. rysů (488 244)	Zm. vel. modelu (19 722KB)	Zm. úsp. slov (93.50%)	Zm. úsp. slov (89.00%)
U00	Slovní tvar [-2]	206 833	29 016	725 KB	-4.56%	-0.06%
U01	Slovní tvar [-1]	**181 706**	**-40 061**	**-1 740 KB**	**0.65%**	**5.15%**
U02	Slovní tvar [0]	**93 515**	**-72 383**	**-3 372 KB**	**-1.20%**	**3.30%**
U03	Slovní tvar [+1]	175 053	-74 147	-3 115 KB	-4.52%	-0.02%
U04	Slovní tvar [+2]	204 827	-70 353	-2 927 KB	-4.24%	0.26%
U10	Spol. značka [-2]	**29 032**	**-60 865**	**-2 158 KB**	**0.52%**	**5.02%**
U11	Spol. značka [-1]	20 282	60 915	- 685 KB	-4.55%	-0.05%
U12	Spol. značka [0]	**2 196**	**60 916**	**3 636KB**	**-1.75%**	**2.75%**
U13	Spol. značka [+1]	19 776	-61 594	-2 333 KB	-4.59%	-0.09%
U14	Spol. značka [+2]	28 410	-53 544	-1 875 KB	-4.57%	-0.07%
U20	Lemma [-2]	165 890	-23 301	-1 043 KB	-4.61%	-0.11%
U21	Lemma [-1]	**143 299**	**-53 668**	**-2 190 KB**	**-0.14%**	**4.36%**
U22	Lemma [0]	83 030	-68 316	-3 148 KB	-4.55%	-0.05%
U23	Lemma [+1]	151 355	-74 709	-2 970 KB	-4.40%	0.10%
U24	Lemma [+2]	174 076	-42 133	-1 788 KB	-4.51%	-0.01%
U30	Přípona slova [-2]	43 638	-21 752	- 546 KB	-4.42%	0.08%
U31	Přípona slova [-1]	32 870	-19 219	- 511 KB	-4.60%	-0.10%
U32	Přípona slova [0]	6 848	-13 741	- 432 KB	-4.68%	-0.18%
U33	Přípona slova [+1]	33 256	-56 524	-2 044 KB	-4.13%	0.37%
U34	Přípona slova [+2]	42 945	-13 186	- 339 KB	-4.49%	0.01%
U40	První písmeno	3 542	-38 863	- 476 KB	-4.53%	-0.03%
U41	První dvě písmena	32 709	-24 246	- 710 KB	-4.44%	0.06%
U42	První tři písmena	63 397	-44 076	-1 536 KB	-4.51%	-0.01%
U43	Poslední písmeno	2 560	-32 583	-1 142 KB	-4.62%	-0.12%
U50	Velikost písmen	1 503	-56 472	-2 146 KB	-4.54%	-0.04%
U51	Pozice ve větě	**18 563**	**-63 985**	**-2 302 KB**	**-0.89%**	**3.61%**
U52	Délka slova	3 542	-38 863	-1 400 KB	-4.53%	-0.03%
U53	Zn. nejbl. slovesa	10 102	-37 928	-19 722 KB	-4.78%	-0.28%

Tabulka 6.9: Srovnání výsledků jednotlivých pravidel s referenčními daty při regularizaci L1

Zhodnocení jednotlivých pravidel

Pro výběr nejméně vhodných pravidel použijeme tabulku 6.9, přesněji její poslední sloupec, ve kterém jsou pravidla porovnávány oproti hodnotě 89% a pomineme všechny tučně zvýrazněné řádky. Kladná hodnota znamená zlepšení v případě vynechání daného pravidla a proto budeme vyhledávat ty řádky, které mají co nejvyšší kladnou hodnotu.

Po výběru pouze kladných hodnot nám zůstalo 6 pravidel (viz. tabulka 6.10). Dle testů jsou nejméně přínosná pravidla popisující příponu slova na pozici [x+1] a slovní tvar na pozici [x+2]. Při vynechání všech těchto pravidel ubereme ze vstupního modelu přes půl milionu rysů.

Naopak nejvíce přínosné se zdá být pravidlo popisující značku nejbližšího slovesa. Ostatní prvky, které se ukázaly být důležité v druhém testu, využívajícím regularizaci L2, v tomto případě jsou ovlivněny chybou a proto je těžké rozhodnout, zda mají pro testování stejný přínos.

ID	Popis pravidla	Zm. poč. rysů	Zm. kon. poč. rysů	Zm. vel. modelu	Zm. úsp. slov
U04	Slovní tvar [+2]	204 827	-70353	-2 927 KB	0.26%
U23	Lemma [+1]	151 355	-74 709	-2 970 KB	0.10%
U30	Přípona slova [-2]	43 638	-21 752	- 546 KB	0.08%
U33	Přípona slova [+1]	33 256	-56 524	-2 044 KB	0.37%
U34	Přípona slova [+2]	42 945	-13 186	- 339 KB	0.01%
U41	První dvě písmena	32 709	-24 246	- 710 KB	0.06%
	Celkem	508730	-260 770	-9 539 736B	
	Teoretický výsledek	1 493 626	227 474	10 182 KB	

Tabulka 6.10: Nejméně vhodná pravidla generování rysů při regularizaci L1 (s nezápornou hodnotou)

6.3.2 Regularizace L2

Příprava

Při tomto testu byla využita stejná pravidla, jako při tom předchozím - budou tedy postupně spouštěny testy a po jednom budou vypouštěna všechna pravidla pro generování rysů. Výsledky budou porovnány s referenčními hodnotami, tvořenými výsledkem testu se všemi dostupnými pravidly.

Konstantní podmínky

Algoritmus: L-BFGS
Δ*:* 0.1
Části značky: 1 – 5 a 8 – 9
Počet vět při testování úspěšnosti: 7000
Regularizace: L2

Vyhodnocení

Při zobrazení výsledků v tabulce 6.11 jsem se rozhodl kompletně vynechat zobrazení počtu příznaků, neboť je závislé pouze na daném rysu a kvůli využití regularizace L2 neměnné. Všechny hodnoty jsou tedy již popsány v tabulkách 6.8 a 6.9.

Při pohledu na výsledky, obzvláště ty zachycené v tabulce 6.12, jde vidět, že jednotlivá pravidla mají na výsledek testu mnohem větší vliv, než v předešlém případě. Při vynechání některých pravidel úspěšnost klesá až k hranici 90% a naopak u jiných stoupá až téměř k 94%. Co je však překvapivé, že při vynechání většiny pravidel se konečná úspěšnost zvýšila. Mohlo by se tedy zdát, že celý model by bylo nejlepší otestovat s množinou pouze 8 pravidel, jejichž vypuštění mělo na výsledek záporný vliv, bychom měli získat nejlepší výsledky.

Co se týče paměťové náročnosti, je v tomto případě model povětšinou téměř stejný, jako bez vynechání jakéhokoliv pravidla - změna se pohybuje do 10%, což se přímo odvíjí od počtu rysů vygenerovaných vynechaným pravidlem. Nejvíce místa opět zabírají stejná pravidla jako v předchozí kapitole tj. Slovní tvary a možná lemmata.

Vyn. rys	Počet iterací	Doba trénování	Velikost modelu	Úsp. slov	Úsp. vět
-	85	68h 18m	76 697 KB	92.17%	37.81%
U00	92	97h 42m	68 334 KB	92.06%	37.18%
U01	89	94h 11m	68 711 KB	90.06%	23.14%
U02	89	111h 48m	70 778 KB	92.81%	43.13%
U03	93	110h 56m	69 117 KB	93.21%	45.61%
U04	89	74h 25m	69 117 KB	93.34%	46.75%
U10	87	74h 2m	75 978 KB	92.25%	38.21%
U11	87	78h 34m	76 188 KB	91.88%	36.02%
U12	106	109h 0m	76 622 KB	88.41%	21.38%
U13	85	117h 7m	76 200 KB	93.54%	48.15%
U14	84	132h 24m	75 993 KB	92.83%	42.47%
U20	90	88h 28m	70 713 KB	92.65%	41.10%
U21	86	103h 12m	71 135 KB	92.64%	41.96%
U22	87	145h 2m	72 557 KB	92.52%	40.71%
U23	86	0h 0m	70 987 KB	93.19%	45.22%
U24	85	0h 0m	70 507 KB	92.67%	40.78%
U30	82	137h 37m	75 593 KB	92.74%	41.84%
U31	84	84h 49m	75 850 KB	92.80%	42.09%
U32	87	71h 9m	76 473 KB	91.65%	32.23%
U33	85	110h 53m	75 841 KB	93.03%	44.06%
U34	82	85h 21m	75 610 KB	91.60%	32.17%
U40	85	81h 29m	76 496 KB	92.07%	36.79%
U41	86	94h 15m	75 831 KB	92.40%	39.32%
U42	87	69h 2m	74 748 KB	92.88%	43.22%
U43	85	154h 52m	75 527 KB	92.38%	39.52%
U50	84	183h 48m	76 661 KB	90.43%	25.11%
U51	89	97h 36m	76 243 KB	92.26%	38.77%
U52	81	82h 28m	76 611 KB	93.06%	44.39%
U53	85	71h 54m	76 452 KB	90.04%	24.93%

Tabulka 6.11: Výsledky testů vynechání příznaků při regularizaci L2

ID	Popis pravidla	Zm. vel. modelu (76 697 KB)	Zm. úsp. slov (92.17%)
U00	Slovní tvar [-2]	-8 363 KB	-0.11%
U01	Slovní tvar [-1]	-7 986 KB	-2.11%
U02	Slovní tvar [0]	-5 918 KB	0.64%
U03	Slovní tvar [+1]	-7 579 KB	1.04%
U04	Slovní tvar [+2]	-7 579 KB	1.17%
U10	Společná značka [-2]	- 718 KB	0.08%
U11	Společná značka [-1]	- 508 KB	-0.29%
U12	Společná značka [0]	- 74 KB	-3.76%
U13	Společná značka [+1]	- 496 KB	1.37%
U14	Společná značka [+2]	- 703 KB	0.66%
U20	Možná lemmata [-2]	-5 983 KB	0.48%
U21	Možná lemmata [-1]	-5 561 KB	0.47%
U22	Možná lemmata [0]	-4 140 KB	0.35%
U23	Možná lemmata [+1]	-5 709 KB	1.02%
U24	Možná lemmata [+2]	-6 190 KB	0.50%
U30	Přípona slova [-2]	-1 103 KB	0.57%
U31	Přípona slova [-1]	- 847 KB	0.63%
U32	Přípona slova [0]	- 223 KB	-0.52%
U33	Přípona slova [+1]	- 855 KB	0.86%
U34	Přípona slova [+2]	-1 086 KB	-0.57%
U40	První písmeno	- 200 KB	-0.10%
U41	První dvě písmena	- 866 KB	0.23%
U42	První tři písmena	-1 948 KB	0.71%
U43	Poslední písmeno	-1 170 KB	0.21%
U50	Velikost písmen	-76 697 KB	-92.17%
U51	Pozice ve větě	- 453 KB	0.09%
U52	Délka slova	- 86 KB	0.89%
U53	Značka nejbl. slovesa	- 244 KB	-2.13%

Tabulka 6.12: Srovnání výsledků jednotlivých pravidel s referenčními daty při regularizaci L2

Zhodnocení jednotlivých pravidel

Nejsilnější vliv na kvalitu modelu má podle tabulky 6.13 pravidlo určující *společné sloupce značky* na aktuální pozici, následované *značkou nejbližšího slovesa* a *slovním tvarem předcházejícího prvku*, kde po vynechání kteréhokoliv ze zmíněných pravidel klesá úspěšnost o více než 2 procenta. Následuje *velikost písmen* a poté výrazný propad až k hranici 0.5% a zbylým pravidlům.

Naproti tomu, jak je vyobrazeno v tabulce 6.14, nejhůře dopadlo pravidlo se *společnoou značkou následujícího prvku*, oba *nadcházející slovní tvary* (na pozici +1 a +2) a *možná lemmata* na pozici +1, kdy po vynechání těchto prvků stoupla konečná úspěšnost o více než jedno procento. Tato pravidla jsou následována dalšími osmi pravidly se změnou větší než půl procenta a zbylými pravidly klesajícími až k nule. Po vynechání všech prvků se záporným přínosem bychom se mohli dostat až na konečnou velikost modelu okolo 19MB a na čtvrtinový počet rysů, což by mohlo přínést markantní zvýšení rychlosti a úsporu místa.

ID	Popis pravidla	Počet rysů	Zm. vel. modelu	Zm. úsp. slov
U12	Společná značka [0]	2 196	- 74 KB	-3.76%
U53	Značka nejbl. slovesa	10 102	- 244 KB	-2.13%
U01	Slovní tvar [-1]	181 706	-7 986 KB	-2.11%
U50	Velikost písmen	1 503	- 36 KB	-1.74%
U34	Přípona slova [+2]	42 945	-1 086 KB	-0.57%
U32	Přípona slova [0]	6 848	- 223 KB	-0.52%
U11	Společná značka [-1]	20 282	- 508 KB	-0.29%
U00	Slovní tvar [-2]	206 833	-8 363 KB	-0.11%
U40	První písmeno	8 169	- 200 KB	-0.10%

Tabulka 6.13: Pravidla s kladným efektem, seřazená podle změny úšpěšnosti

ID	Popis pravidla	Počet rysů	Zm. vel. modelu	Zm. úsp. slov
U13	Společná značka [+1]	19 776	- 496 KB	1.37%
U04	Slovní tvar [+2]	175 053	-7 579 KB	1.17%
U03	Slovní tvar [+1]	204 827	-7 579 KB	1.04%
U23	Možná lemmata [+1]	151 355	-5 709 KB	1.02%
U52	Délka slova	3 542	- 86 KB	0.89%
U33	Přípona slova [+1]	33 256	- 855 KB	0.86%
U42	První tři písmena	63 397	-1 948 KB	0.71%
U14	Společná značka [+2]	28 410	- 703 KB	0.66%
U02	Slovní tvar [0]	93 515	-5 918 KB	0.64%
U31	Přípona slova [-1]	32 870	- 847 KB	0.63%
U30	Přípona slova [-2]	43 638	-1 103 KB	0.57%
U24	Možná lemmata [+2]	174 076	-6 190 KB	0.50%
U20	Možná lemmata [-2]	165 890	-5 983 KB	0.48%
U21	Možná lemmata [-1]	143 299	-5 561 KB	0.47%
U22	Možná lemmata [0]	83 030	-4 140 KB	0.35%
U41	První dvě písmena	32 709	- 866 KB	0.23%
U43	Poslední písmeno	46 198	-1 170 KB	0.21%
U51	Pozice ve větě	18 563	- 453 KB	0.09%
U10	Společná značka [-2]	29 032	- 718 KB	0.08%
	Celkem	1 512 662	-57 914 KB	
	Teoretický výsledek	479 081	18 688 KB	

Tabulka 6.14: Pravidla se záporným efektem, seřazená podle změny úšpěšnosti

6.4 Využití získaných poznatků pro zlepšení úspěšnosti testování

Tato skupina testů se zaobírá využitím zjištěných údajů k vlastnímu zlepšení výsledků algoritmů. Za pomoci vynechávání pravidel pro generování rysů, která podávala nejhorší výsledky, bude v několika specifikovaných krocích prováděno testování (vždy několik dalších pravidel bude přidáno do vynechané množiny) a získané výsledky budou porovnávány s nejkvalitnějším výsledkem z původního testu.

6.4.1 Regularizace L1

Pro tyto testy byly nakonec použity pouze dva kroky: bez tří a šesti nejméně vhodných pravidel. Takto malý počet testů byl způsoben anomáliemi při regularizaci L1 a nepřesvědčívými výsledky po jejich provedení. Při vynechání prvních třech pravidel se úspěšnost zvýšila o 28 setin procenta, při dalších třech se ale již snížila pod úroveň úspěšnosti s vynecháním pouze jediného prvku. Zřejmě by bylo možné ještě provést několik dalších testů za přidání dalších prvků k prvnímu kroku, z časových důvodů však byla vypočetní síla přesunuta na testy s vyšší hodnotou. Jediná přínosná část při tomto nastavení by tedy mohla být velikost modelu, která se nakonec pohybuje okolo 16MB a je tedy mnohem nižší, než u L2. Více podrobností v tabulkách 6.15 a 6.16.

ID	Vynechaná pravidla
Krok 1	U04 U23 U33
Krok 2	U04 U23 U30 U33 U34 U41

Tabulka 6.15: Výčet kroků a jednotlivých vynechaných pravidel

ID	Doba trénování	Velikost modelu	Úvodní počet rysů	Konečný počet rysů	Úsp. slov	Úsp. vět
U33	78h 14m	17 678 KB	1 969 100	431 720	89.37%	11.19%
Krok 1	46h 17m	16 552 KB	1 612 918	413 408	89.65%	13.92%
Krok 2	46h 15m	14 796 KB	1 493 626	354 589	88.99%	8.00%

Tabulka 6.16: Výsledky při vynechání nejméně vhodných pravidel při regularizaci L1

6.4.2 Regularizace L2

V této části budou popsány výsledky s použitím regularizace L2 v algoritmech L-BFGS a SGD a výsledky budou vzájemně porovnány. Jako referenční hodnota nebude sloužit nejlepší výsledek při vynechání jednotlivých pravidel, jak tomu bylo v předchozí kapitole, ale hodnota, získaná při využití všech pravidel. Důvodem je absence dat s jednotlivými vypuštěnými pravidly pro algoritmus SGD a tím pádem nemožnost využití těchto hodnot ke vzájemnému srovnání. Jednotlivé kroky s vypuštěnými pravidly jsou popsány v tabulce 6.17 - Nejprve jsou vypuštěna tři nejméně vhodná pravidla a v každém dalším kroku jsou k nim přidána další dvě.

ID	Vynechaná pravidla
Krok 1	U03 U04 U13
Krok 2	U03 U04 U13 U23 U52
Krok 3	U03 U04 U13 U23 U33 U42 U52
Krok 4	U02 U03 U04 U13 U14 U23 U33 U42 U52

Tabulka 6.17: Výčet kroků a jednotlivých vynechaných pravidel

L-BFGS

Výsledky za použití algoritmu L-BFGS jsou zachyceny v tabulce 6.18. Můžeme vypozorovat, že s větším počtem vynechaných pravidel se zvyšuje počet uskutečněných iterací, potřebných k dosažení rozdílu mezi iteracemi o hodnotě Δ. Celkový potřebný čas se však viditelně nezvýšil. Počet rysů se při v kroku 4 snížil až na dvě třetiny velikosti originální množiny a tím pádem se snížila i velikost modelu z původních 77MB na 45MB.

Zatímco bez vynechání jakéhokoliv rysu je úspěšnost určení jednotlivých slov rovna $92,17\%$, v prvním kroku se zvýšila na $93,3\%$. Při vynechání dalších dvou prvků se ovšem snížila o $0,3\%$, z čehož vyplývá, že alespoň jedno z pravidel U23 a U52 je vhodné zachovat. Při vynechání dalších dvou se úspěšnost opět zvýšila a je dokonce o $0,01\%$ vyšší, než v kroku 1. Pravidla U42 a U52 (nebo alespoň jedno z nich) tedy nejsou pro trénování vhodná a je výhodnější je do použité množiny nezahrnout. Při kroku 4 se opět dostáváme k hodnotě 93% a tím pádem na úroveň kroku 2, tato pravidla je tím pádem opět výhodnější zachovat.

Pro další zlepšení výsledků by bylo vhodné vytvořit více skupin vypuštěných pravidel a hlouběji zanalyzovat důsledky jejich vynechání na celkovou úspěšnost.

ID	Počet iterací	Doba trénování	Velikost modelu	Počet rysů	Úsp. slov	Úsp. vět
-	85	68h 18m	76 697 KB	2 002 356	92.17%	37.81%
Krok 1	87	68h 56m	60 470 KB	1 602 700	93.30%	47.08%
Krok 2	91	100h 43m	54 674 KB	1 447 803	93.03%	45.45%
Krok 3	97	75h 34m	51 870 KB	1 351 150	93.31%	47.74%
Krok 4	101	92h 4m	45 247 KB	1 229 225	93.00%	45.99%

Tabulka 6.18: Výsledky při vynechání nejméně vhodných pravidel při regularizaci L2 a algoritmu L-BFGS

SGD

Z tabulky 6.19 je vidět, že u algoritmu SGD se při vynechávání rysů počet iterací nemění, stejně jako se nezvyšuje ani doba trénování. Počet rysů a velikost modelu je pak téměř identická jako v případě algoritmu L-BFGS. Co se týče úspěšnosti určení jednotlivých slov, je situacie podobná jako v předchozím testu. Jediným rozdílem je, že algoritmus SGD je zřejmě schopen lépe vynechat nevhodné rysy a nejvyšší úspěšnosti dosahuje při využití všech pravidel. Při kroku 1 se však úspěšnost sníží pouze nepatrně a to o 0.01%. U následujících kroků se opakuje stejný průběh jako u L-BFGS, kdy u kroku 2 se sníží úspěšnost, aby se při kroku 3 opět o kousek zvýšila a v kroku 4 klesla na úroveň kroku 2.

ID	Počet iterací	Doba trénování	Velikost modelu	Počet rysů	Úsp. slov	Úsp. vět
-	35	30h 13m	76696816B	2002349	93.96%	51.60%
Krok 1	35	54h 44m	60470280B	1602698	93.95%	51.86%
Krok 2	35	45h 44m	54608868B	1446867	93.69%	50.37%
Krok 3	36	46h 28m	51811720B	1351150	93.72%	51.22%
Krok 4	38	44h 57m	45191864B	1229225	93.61%	50.52%

Tabulka 6.19: Výsledky při vynechání nejméně vhodných pravidel při regularizaci L2 a algoritmu SGD

Srovnani L-BFGS a SGD

Konečné srovnání je zachyceno v tabulce 6.20. Jak jde vidět, nejlepší výsledky podává algoritmus SGD a to při využití všech pravidel rysů. L-BFGS by se mu mohl teoreticky přiblížit pouze po důkladné analýze a vynechání všech nevhodných pravidel. Zatímco u L-BFGS je krok 4 přibližně stejně úspěšný jako krok 2, u SGD jsou kroky 2 až 4 na podobné úrovni a tudíž přibližně o 0.3% méně úspěšné, než v případě využití všech pravidel, nebo při kroku 1.

ID	Doba trénování		Úspěšnost slov		Úspěšnost vět	
	L-BFGS	SGD	L-BFGS	SGD	L-BFGS	SGD
-	68h 18m	30h 13m	92.17%	93.96%	37.81%	51.60%
Krok 1	68h 56m	54h 44m	93.30%	93.95%	47.08%	51.86%
Krok 2	100h 43m	45h 44m	93.03%	93.69%	45.45%	50.37%
Krok 3	75h 34m	46h 28m	93.31%	93.72%	47.74%	51.22%
Krok 4	92h 4m	44h 57m	93.00%	93.61%	45.99%	50.52%

Tabulka 6.20: Srovnání mezi algoritmy L-BFGS a SGD při vynechání nejméně vhodných pravidel při regularizaci L2

6.5 Srovnání s programem Morče

Nejlepší získanou úspěšností je 94.51% při využití algoritmu SGD, 50 000 vstupních větách a $\Delta = 0.1$. Z toho vyplývá, že je stále možno přidat do testovací množiny dalších 27 000 vět a zvýšit přesnost o několik řádů. Celý proces testování by pak neměl přesáhnout několik týdnů a výsledná hodnota úspěšnosti by se pak měla ještě zvýšit. Tyto hodnoty však byly získány se zkrácenou značkou a nemůžeme je tedy přímo porovnávat s úspěšností programu Morče. K tomu byl uskutečněn test s využitím algoritmu SGD, 30 000 větami a $\delta = 0.1$. Jeho konečná úspěšnost je 92.72%, což je oproti 93.96% při využití částečné značky snížení o více než jedno procento. Pokud bychom testy spustili se všemi dostupnými daty a při vyšší přesnosti, nejspíše bychom se stále drželi daleko od hodnoty 96%, které by měl dosahovat program Morče.

Kapitola 7

Závěr

Při zpracování této práce byla vytvořena množina pravidel pro generování rysů a s použitím algoritmu Conditional Random Fields byla otestována na vstupních datech. Byly otestovány regularizace L1 a L2, kde první zmíněná poskytuje menší výsledný model, ale také nižší úspěšnost. Za využití algoritmu L-BFGS pak byla zanalyzována všechna pravidla a ta nejhorší byla postupně vynechávána za účelem zajištění co nejlepší úspěšnosti, která se nakonec dostala až na hodnotu 93.31% při 30 000 vstupních větách. Výsledky, zjištěné při vynechání jednotlivých množin pravidel, také naznačují, že při jiné kombinaci použitých a vynechaných pravidel by se mohla konečná úspěšnost dále zvyšovat.

Skupiny pravidel pro generování rysů, nevhodných pro L-BFGS, byly postupně vynechávány i při využití algoritmu SGD, jehož úspěšnost se tím však nenavyšovala, ale naopak mírně snížila. Celkově však SGD byl o několik procent úspěšnější a při využití všech pravidel poskytoval vyšší výsledky, než L-BFGS po využití všech získaných poznatků a vynechání nevhodných pravidel. Nejlepší získanou hodnotou je pak 94.51% při 50 000 vstupních větách. Dalšího zvýšení úspěšnosti bychom mohli dosáhnout přetestováním tohoto algoritmu stejným způsobem, jako L-BFGS (vynecháním jednotlivých pravidel a porovnáním s referenční hodnotou), algoritmu by to však zřejmě úspěšnost zvýšilo pouze minimálně.

Jelikož byly téměř všechny testy provedeny pouze za použití zkrácené značky, nabízí se otestování těchto pravidel i s využitím plné značky a tím pádem dalších možností pro generování rysů pomocí informací, které byly v této práci pomíjeny. Také by bylo vhodné využít všechna dostupná data, případně další dostupné korpusy, a zvýšit přesnost trénování pro detailnější informace.

Dalším možností pro zvýšení úspěšnosti je vytvoření vlastní implementace, která by byla schopna dynamicky generovat rysy za běhu programu a nevyužívat pouze ty, které byly vygenerovány před započetím zpracování –byla by tak mnohem vhodnější pro zpracování českého jazyka. Tento přístup by však byl mnohem náročnější na výpočetní výkon, neboť by bylo nutno místo číselných konstant využívat řetězce, obsahujícího všechny pozice značky.

Literatura

[1] Bottou, L.: Stochastic Gradient Learning in Neural Networks.
http://leon.bottou.org/publications/pdf/nimes-1991.pdf, 1991.

[2] Fink, G. A.: *Markov models for pattern recognition: from theory to applications.* Springer, 2008, iSBN 978-3-540-71766-9.

[3] Forney, D. G.: The Viterbi Algorithm. 1973.

[4] Kovář, J.: Skryté Markovské modely a neuronové sítě.
https://dip.felk.cvut.cz/browse/pdfcache/kovarj4_2008dipl.pdf, 2008.

[5] Lafferty, J.; McCallum, A.; Pereira, F.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data.
http://www.cis.upenn.edu/ pereira/papers/crf.pdf, 2001.

[6] Moss, L.: Example of the Baum-Welch Algorithm.
http://www.indiana.edu/ iulg/moss/hmmcalculations.pdf, 2008.

[7] Nash, S. G.; Nocedal, J.: A Numerical Study of the Limited Memory BFGS Method and the Truncated-Newton Method for Large Scale Optimization.
http://math.asu.edu/ jtaylor/teaching/Fall2010/APM530/papers/Nash91.pdf, 1991.

[8] Ng, A. Y.: Feature selection, L1 vs. L2 regularization, and rotational invariance.
http://ai.stanford.edu/ ang/papers/icml04-l1l2.pdf, 2004.

[9] Rabiner, L. R.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.
http://www.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/tutorial%20on%20hmm%20and%20applications.pdf, 1989.

[10] Shokhirev, N.: ABC Tutorials - Hidden Markov Models.
http://www.shokhirev.com/nikolai/abc/alg/hmm/hmm.html, 2010.

[11] Spoustová, D.; Hajič, J.; Raab, J.; aj.: Semi-supervised Training for the Averaged Perceptron POS Tagger.
http://www.aclweb.org/anthology/E/E09/E09-1087.pdf, 2009.

[12] Votrubec, J.: Volba vhodné sady rysů pro morfologické značkování češtiny.
http://ufal.mff.cuni.cz/czech-tagging/VotrubecMSC2005.pdf, 2005.

Příloha A

Obsah CD

`plugin/`	plugin s informačním systémem pro CMS systém Drupal
`scripts/`	skripty použité při procesu testování programů CRF++ a CRFSuite
`tex/`	zdrojové soubory práce (LaTeX)
`MoTag.pdf`	digitální verze práce
`README`	základní informace

Příloha B

Přehled provedených testů

Popis sloupců

- **ID** - číselný identifikátor testu
- **PC** - identifikátor stroje, na kterém byl test proveden
- **Reg.** - regularizace (L1 nebo L2)
- **Počet vět** - počet ucelených sekvencí vstupních dat, použitých při testování
- **Max it.** - maximální počet iterací (0 značí neomezeně)
- **Delta** - číslo, specifikující hodnotu Δ, která je použita jako velikost práhu určeného k ukončení testování
- **Části značky** - jednotlivé části značky, které jsou použity ve vstupních souborech, stejně jako u předchozího vstupního parametru jsou definovány skupinou zaškrtávacích políček, včetně možnosti zaškrtnout všechna, nebo žádné
- **Počet slov** - počet slov v souboru trénovacích dat
- **Počet značek** - celkový počet unikátních značek v souboru trénovacích data
- **Poč. it.** - počet iterací, provedených při testování
- **Doba trén.** - čas potřebný k natrénování modelu
- **Doba test.** - čas potřebný k označkování testovacího textu
- **Velikost modelu** - velikost výsledného modelu
- **Úv. poč. rysů** - množství rysů na začátku trénování
- **Kon. poč. rysů** - množství rysů na konci trénování
- **Úsp. slov** - poměr mezi počtem správně určených slov a celkovým počtem slov obsaženým v testovacích datech
- **Úsp. vět** - poměr mezi počtem celých správných vět (kde všechna slova byla určena správně) a celkovým počtem vstupních vět

ID	Pc	reg.	Počet vět	Max. it.	Delta	Části značky	Počet slov	Počet značek	Poč. it.	Doba trén.	Doba test.	Velikost modelu	Kon. poč. rysů	Úv. poč. rysů	Úsp. slov	Úsp. vět
1	pcnlp7	L1	30000	50	0	xxxxx--xx-------	497125	822	50	40h 4m	434s	19 930 KB	2002356	493646	93.72%	48.21%
2	pcnlp3	L1	10000	50	0	xxxxx--xx-------	162090	743	50	11h 37m	378s	8 326 KB	905326	280262	92.65%	43.02%
3	pcnlp4	L1	10000	50	0	xxxxx--xx-------	162090	743	50	9h 52m	374s	7 669 KB	844254	177223	87.27%	0.08%
4	pcnlp5	L1	10000	50	0	xxxxx--xx-------	162090	743	50	12h 52m	381s	7 485 KB	821208	167380	86.92%	6.75%
5	pcnlp6	L1	10000	50	0	xxxxx--xx-------	162090	743	50	12h 56m	363s	7 649 KB	729530	157514	85.14%	6.51%
7	pcnlp4	L1	10000	50	0	xxxxx--xx-------	162090	743	50	11h 29m	374s	7 794 KB	813648	177757	87.41%	8.64%
8	pcnlp5	L1	10000	50	0	xxxxx--xx-------	162090	743	50	12h 8m	381s	8 027 KB	903307	190949	91.73%	37.84%
11	pcnlp3	L1	10000	50	0	xxxxxxxxxxxxx--	162090	944	50	23h 3m	843s	827 KB	942277	201268	87.14%	7.28%
12	pcnlp4	L1	10000	50	0	xxxxxxxxxxx----	162090	944	50	25h 6m	849s	8 270 KB	942277	201268	87.14%	7.28%
13	pcnlp7	L1	30000	50	0	xxxxx--xx-------	497125	822	50	37h 56m	421s	18 954 KB	1968646	461337	88.71%	7.99%
14	pcnlp5	L1	10000	50	0	xxxxx--xx-------	162090	743	50	12h 32m	383s	7 435 KB	872386	177478	87.22%	5.89%
15	pcnlp3	L1	30000	10	0	xxxxxxxxxxxxx--	30000	1078	10	18h 54m	1141s	72 418 KB	2086633	1927445	78.66%	14.53%
16	pcnlp4	L1	30000	20	0	xxxxxxxxxxx----	30000	1078	20	33h 51m	1118s	59 145 KB	2086633	1571906	91.41%	38.35%
17	pcnlp5	L1	30000	30	0	xxxxx--xx-------	3000	822	30	25h 53m	601s	42 344 KB	2002356	1100285	93.97%	51.45%
18	pcnlp6	L1	30000	40	0	xxxxx--xx-------	30000	822	40	34h 24m	605s	29 765 KB	2002356	746813	93.96%	50.11%
19	athena3	L1	30000	60	0	xxxxxxxx--------	30000	854	60	156h 10m	1067s	16 835 KB	2006673	412986	88.91%	6.31%
21	pcnlp3	L1	30000	10	0	xxxxx--xx-------	497125	822	10	11h 42m	673s	70 273 KB	2002356	1841799	79.14%	15.04%
22	pcnlp4	L1	30000	20	0	xxxxx--xx-------	497125	822	20	20h 53m	675s	57 371 KB	2002356	1500554	91.87%	40.15%
23	pcnlp5	L1	5000	50	0	xxxxx--xx-------	83395	671	50	5h 30m	304s	4 752 KB	558871	113149	91.04%	34.70%
24	pcnlp6	L1	15000	50	0	xxxxx--xx-------	245214	769	50	23h 46m	544s	10 991 KB	1202680	266245	87.90%	7.73%
26	pcnlp3	L1	20000	50	0	xxxxx--xx-------	325818	792	50	33h 41m	527s	13 706 KB	1464587	334939	88.37%	8.07%
27	pcnlp4	L1	25000	50	0	xxxxx--xx-------	409037	811	50	45h 25m	504s	17 140 KB	1724528	423649	88.68%	8.20%
28	pcnlp5	L1	35000	50	0	xxxxx--xx-------	584740	833	50	56h 44m	590s	22 744 KB	2260670	564590	89.38%	10.26%
29	pcnlp6	L1	40000	50	0	xxxxx--xx-------	670572	843	50	66h 54m	604s	23 584 KB	2488724	586499	90.12%	15.69%
30	pcnlp3	L1	45000	50	0	xxxxx--xx-------	758020	849	50	128h 26m	718s	27 178 KB	2716001	680082	90.62%	18.99%
31	pcnlp4	L1	50000	50	0	xxxxx--xx-------	842463	854	50	199h 44m	738s	28 602 KB	2924981	717070	89.45%	6.85%
32	pcnlp5	L1	55000	50	0	xxxxx--xx-------	931776	867	50	90h 22m	591s	30 087 KB	3140944	758370	94.74%	53.77%
34	pcnlp4	L1	50000	50	0	xxxxx--xx-------	842463	854	50	93h 51m	677s	28 602 KB	2924981	717070	89.45%	6.85%
35	locaL1	L2	10000	50	0	xxxxx--xx-------	162090	743	50	15h 56m	595s	36 244 KB	905326	905326	91.97%	40.64%
36	locaL2	L2	20000	50	0	xxxxx--xx-------	325818	792	50	44h 30m	935s	57 072 KB	1464587	1464587	92.78%	45.28%
37	athena1	L1	5000	50	0	xxxxx--xx-------	83395	671	50	8h 46m	585s	4 752 KB	558871	113149	91.04%	34.70%
38	athena2	L1	10000	50	0	xxxxx--xx-------	162090	743	50	20h 47m	646s	8 326 KB	905326	200613	92.65%	43.02%
39	athena3	L1	55000	50	0	xxxxx--xx-------	931776	867	50	156h 3m	1217s	30 087 KB	3140944	758370	94.74%	53.77%
41	pcnlp3	L2	40000	50	0	xxxxx--xx-------	670572	843	50	114h 50m	555s	94 190 KB	2488724	2488724	93.61%	49.06%
42	pcnlp5	L1	60000	50	0	xxxxx--xx-------	1015213	870	50	111h 35m	696s	34 119 KB	3329504	863220	89.63%	8.05%
43	pcnlp5	L2	40000	50	0	xxxxx--xx-------	670572	843	50	54h 54m	556s	94 190 KB	2488724	2488724	93.61%	49.06%
44	athena1	L2	50000	50	0	xxxxx--xx-------	842463	854	50	128h 19m	1516s	109 664 KB	2924981	2924981	93.83%	50.26%
45	athena2	L2	60000	50	0	xxxxx--xx-------	1015213	870	50	181h 14m	1913s	123 654 KB	3329504	3329504	94.04%	51.09%
46	pcnlp3	L2	30000	50	0	xxxxx--xx-------	497125	822	50	42h 26m	584s	76 697 KB	2002356	2002356	93.51%	48.41%
47	athena1	L2	70000	50	0	xxxxx--xx-------	1185724	881	50	191h 8m	1485s	136 263 KB	3697048	3697048	94.05%	51.52%
48	pcnlp4	L1	50000	0	0.1	xxxxx--xx-------	842463	854	53	110h 58m	661s	2 777 KB	2924981	696340	89.42%	6.79%
52	athena1	L1	10000	0	0.1	xxxxx--xx-------	162090	743	47	19h 20m	679s	8 723 KB	905326	209912	92.61%	43.12%
53	athena2	L1	20000	0	0.1	xxxxx--xx-------	325818	792	50	58h 54m	1559s	13 706 KB	1464587	334939	88.37%	8.07%
54	athena3	L1	30000	0	0.1	xxxxx--xx-------	497125	822	51	107h 55m	1612s	19 723 KB	2002356	488244	93.50%	46.25%
55	pcnlp3	L1	40000	0	0.1	xxxxx--xx-------	325818	792	74	55h 11m	548s	57 072 KB	1464587	1464587	92.37%	41.68%
56	pcnlp4	L1	40000	0	0.1	xxxxx--xx-------	670572	843	55	85h 43m	559s	22 254 KB	2488724	551872	89.67%	11.26%
57	athena1	L1	40000	0	0.1	xxxxx--xx-------	670572	843	55	155h 57m	1126s	22 254 KB	2488724	551872	89.67%	11.26%
58	pcnlp4	L2	40000	0	0.1	xxxxx--xx-------	670572	843	102	127h 49m	566s	94 190 KB	2488724	2488724	89.86%	18.95%
59	pcnlp5	L2	30000	0	0.1	xxxxx--xx-------	497125	822	51	96h 8m	975s	17 678 KB	1969100	431720	89.37%	11.19%
60	athena2	L2	10000	0	0.1	xxxxx--xx-------	162090	743	55	24h 16m	687s	36 244 KB	905326	905326	92.04%	40.64%
61	athena3	L2	50000	0	0.1	xxxxx--xx-------	842463	854	105	312h 0m	1599s	24 500 KB	2924981	2924981	89.40%	6.49%
63	athena2	L1	30000	0	0.1	xxxxx--xx-------	497125	822	52	95h 39m	544s	18 322 KB	1998814	449381	88.97%	7.78%
64	pcnlp4	L2	30000	0	0.1	xxxxx--xx-------	497125	822	53	52h 39m	621s	17 420 KB	1983793	424259	92.61%	37.42%
65	athena1	L2	30000	0	0.1	xxxxx--xx-------	497125	822	51	80h 57m	1545s	17 576 KB	2000853	431772	88.96%	8.04%
66	athena2	L2	30000	0	0.1	xxxxx--xx-------	497125	822	52	86h 0m	1388s	41 KB	1998814	449381	88.97%	7.78%
67	athena1	L2	30000	0	0.1	xxxxx--xx-------	497125	822	53	104h 4m	1432s	19 012 KB	1969647	463998	89.06%	8.41%
68	athena2	L2	30000	0	0.1	xxxxx--xx-------	497125	822	53	77h 5m	1233s	18 186 KB	1938959	444168	88.99%	8.23%
69	athena3	L2	30000	0	0.1	xxxxx--xx-------	497125	822	54	120h 38m	985s	18 580 KB	1999796	455661	88.88%	8.37%

Pokračování na další straně...

ID	Pc	reg.	Počet vět	Max. it.	Delta	Části značky	Počet slov	Počet značek	Poč. it.	Doba trén.	Doba test.	Velikost modelu	Kon. poč. rysů	Úv. poč. rysů	Úsp. slov	Úsp. vět
70	pcnlp4	L1	30000	0	0.1	xxxxx--xx-------	497125	822	50	50h 15m	620s	19 176 KB	1958718	466492	89.08%	8.83%
71	athena1	L1	30000	0	0.1	xxxxx--xx-------	497125	822	51	121h 18m	1394s	19 211 KB	1969486	469025	88.90%	8.26%
72	athena2	L1	30000	0	0.1	xxxxx--xx-------	497125	822	53	82h 12m	1224s	19 290 KB	1995508	474503	88.82%	8.00%
73	athena3	L1	30000	0	0.1	xxxxx--xx-------	497125	822	51	78h 14m	874s	17 678 KB	1969100	431720	89.37%	11.19%
74	athena1	L1	30000	0	0.1	xxxxx--xx-------	497125	822	51	103h 53m	1464s	19 383 KB	1959411	475058	89.01%	8.29%
75	athena2	L1	30000	0	0.1	xxxxx--xx-------	497125	822	50	83h 45m	1240s	18 679 KB	1836466	464943	88.89%	8.59%
76	pcnlp4	L1	30000	0	0.1	xxxxx--xx-------	497125	822	50	114h 24m	593s	1 868 KB	1836466	464943	88.89%	8.59%
77	pcnlp4	L1	30000	0	0.1	xxxxx--xx-------	497125	822	50	49h 0m	599s	19 176 KB	1958718	466492	89.08%	8.83%
78	pcnlp4	L1	30000	0	0.1	xxxxx--xx-------	417125	822	55	52h 54m	612s	16 752 KB	1851001	413535	89.10%	8.59%
79	athena2	L1	30000	0	0.1	xxxxx--xx-------	497125	822	50	171h 0m	1231s	17 934 KB	1828280	446111	88.99%	8.31%
80	athena3	L1	30000	0	0.1	xxxxx--xx-------	497125	822	52	75h 0m	972s	17 564 KB	1973324	427379	94.02%	50.21%
81	athena1	L1	30000	0	0.1	xxxxx--xx-------	497125	822	52	82h 55m	1433s	19 037 KB	1982074	549159	88.95%	8.37%
82	pcnlp4	L1	30000	0	0.01	xxxxx--xx-------	497125	822	51	56h 53m	613s	19 723 KB	2002356	488244	93.50%	46.25%
83	athena1	L1	30000	0	0.1	xxxxx--xx-------	497125	822	52	77h 52m	1389s	1 753 KB	1859057	434576	93.36%	44.68%
84	athena2	L1	30000	0	0.1	xxxxx--xx-------	497125	822	53	87h 11m	1350s	16 574 KB	1919326	419928	88.95%	7.81%
85	pcnlp1	L1	30000	0	0.1	xxxxx--xx-------	497125	822	52	113h 12m	605s	17 389 KB	1982580	426650	88.91%	7.67%
86	pcnlp3	L1	30000	0	0.1	xxxxx--xx-------	497125	822	52	58h 7m	602s	17 847 KB	1973946	434700	88.93%	8.03%
87	athena3	L1	30000	0	0.1	xxxxx--xx-------	497125	822	55	77h 2m	970s	23 359 KB	2000160	549160	91.75%	38.07%
88	pcnlp4	L1	30000	0	0.1	xxxxx--xx-------	497125	822	52	66h 26m	613s	20 448 KB	1795523	517260	88.94%	8.54%
89	pcnlp5	L1	30000	0	0.1	xxxxx--xx-------	497125	822	53	54h 50m	609s	17 982 KB	1820650	448183	94.15%	51.47%
90	athena1	L1	30000	0	0.1	xxxxx--xx-------	497125	822	51	87h 46m	1394s	16 350 KB	1908841	415861	92.30%	35.52%
91	athena2	L1	30000	0	0.1	xxxxx--xx-------	497125	822	54	70h 7m	1235s	16 607 KB	1827303	414097	88.98%	8.16%
92	athena3	L1	30000	0	0.1	xxxxx--xx-------	497125	822	52	83h 49m	992s	16 795 KB	1797529	417891	89.26%	10.58%
93	pcnlp3	L2	30000	0	0.1	xxxxx--xx-------	497125	822	92	97h 42m	616s	68 334 KB	1795523	1795523	92.06%	37.18%
94	pcnlp4	L2	30000	0	0.1	xxxxx--xx-------	497125	822	89	94h 11m	619s	68 711 KB	1820650	1820650	90.06%	23.14%
95	pcnlp5	L2	30000	0	0.1	xxxxx--xx-------	497125	822	89	111h 48m	608s	70 779 KB	1908841	1908841	92.81%	43.13%
96	pcnlp1	L2	30000	0	0.1	xxxxx--xx-------	497125	822	93	110h 56m	614s	69 117 KB	1827303	1827303	93.21%	45.61%
97	pcnlp1	L2	30000	0	0.5	xxxxx--xx-------	497125	822	89	74h 25m	668s	69 117 KB	1827303	1827303	93.34%	46.75%
98	pcnlp3	L2	30000	0	0.5	xxxxx--xx-------	497125	822	87	74h 2m	615s	75 979 KB	1973324	1973324	92.25%	38.21%
99	pcnlp4	L2	30000	0	0.5	xxxxx--xx-------	497125	822	87	78h 34m	618s	76 623 KB	1982074	1982074	91.88%	36.02%
100	pcnlp5	L2	30000	0	0.5	xxxxx--xx-------	497125	822	106	109h 0m	614s	76 623 KB	2000160	2000160	88.41%	21.38%
101	athena1	L1	30000	0	0.1	xxxxx--xx-------	497125	822	53	71h 45m	1397s	18 362 KB	1992254	450316	88.72%	8.16%
102	athena2	L2	30000	0	0.5	xxxxx--xx-------	497125	822	85	117h 7m	1267s	76 201 KB	1982580	1982580	93.54%	48.15%
103	athena3	L2	30000	0	0.5	xxxxx--xx-------	497125	822	84	132h 24m	988s	75 994 KB	1973946	1973946	92.83%	42.47%
104	pcnlp6	L2	30000	0	0.1	xxxxx--xx-------	497125	822	90	88h 28m	609s	70 714 KB	1836466	1836466	92.65%	41.10%
106	athena1	L2	30000	0	0.1	xxxxx--xx-------	497125	822	87	145h 2m	1393s	72 557 KB	1919326	1919326	92.52%	40.71%
109	athena2	L2	30000	0	0.1	xxxxx--xx-------	497125	822	82	137h 37m	1400s	75 593 KB	1958718	1958718	92.74%	41.84%
110	pcnlp3	L2	30000	0	0.1	xxxxx--xx-------	497125	822	84	84h 49m	605s	75 850 KB	1969486	1969486	92.80%	42.09%
111	pcnlp7	L2	30000	0	0.1	xxxxx--xx-------	497125	822	87	71h 9m	382s	76 474 KB	1995508	1995508	91.65%	32.23%
112	pcnlp8	L2	30000	0	0.1	xxxxx--xx-------	497125	822	85	110h 53m	541s	75 842 KB	1969100	1969100	93.03%	44.06%
113	pcnlp6	L2	30000	0	0.1	xxxxx--xx-------	497125	822	82	85h 21m	606s	75 611 KB	1959411	1959411	91.60%	32.17%
114	pcnlp4	L2	30000	0	0.1	xxxxx--xx-------	497125	822	85	81h 29m	618s	76 497 KB	1994187	1994187	92.07%	36.79%
115	pcnlp7	L2	30000	0	0.1	xxxxx--xx-------	497125	822	87	69h 2m	380s	74 749 KB	1938959	1938959	92.88%	43.22%
116	pcnlp5	L2	30000	0	0.1	xxxxx--xx-------	497125	822	86	94h 15m	614s	75 831 KB	1969647	1969647	92.40%	39.32%
117	athena2	L2	30000	0	0.1	xxxxx--xx-------	497125	822	85	154h 52m	1390s	75 527 KB	1956158	1956158	92.38%	39.52%
118	athena1	L2	30000	0	0.1	xxxxx--xx-------	497125	822	84	183h 48m	1651s	76 661 KB	2000853	2000853	90.43%	25.11%
119	pcnlp3	L2	30000	0	0.1	xxxxx--xx-------	497125	822	89	97h 36m	614s	76 244 KB	1983793	1983793	92.26%	38.77%
120	pcnlp6	L2	30000	0	0.1	xxxxx--xx-------	497125	822	81	82h 28m	604s	76 611 KB	1998814	1998814	93.06%	44.39%
121	pcnlp8	L2	30000	0	0.1	xxxxx--xx-------	497125	822	85	71h 54m	411s	76 453 KB	1992254	1992254	90.04%	24.93%
122	pcnlp7	L2	30000	0	0.1	xxxxx--xx-------	497125	822	85	68h 18m	385s	76 697 KB	2002356	2002356	92.17%	37.81%
123	pcnlp4	L2	30000	0	0.1	xxxxx--xx-------	497125	822	35	30h 13m	584s	76 697 KB	2002349	2002349	93.96%	51.60%
125	pcnlp1	L2	30000	0	0.001	xxxxx--xx-------	497125	822	57	52h 10m	630s	76 697 KB	2002350	2002350	94.01%	52.16%
126	pcnlp5	L2	30000	0	0.01	xxxxx--xx-------	497125	822	57	48h 38m	591s	76 697 KB	2002356	2002356	94.01%	52.16%
127	pcnlp8	L2	30000	0	0.1	xxxxxxxxxxxxxxx	497125	1290	90	275h 52m	1689s	80 338 KB	2147029	2147029	92.72%	41.65%
129	pcnlp7	L1	30000	0	0.1	xxxxx--xx-------	497125	822	52	46h 17m	371s	16 552 KB	1612918	413408	89.65%	13.92%
130	pcnlp3	L1	30000	0	0.1	xxxxx--xx-------	497125	822	51	46h 15m	597s	14 796 KB	1493626	354589	88.99%	8.00%
131	pcnlp4	L2	40000	0	0.1	xxxxx--xx-------	670572	843	36	41h 25m	557s	94 190 KB	2488724	2488724	94.31%	53.45%
132	athena2	L2	10000	0	0.1	xxxxx--xx-------	162090	743	27	8h 37m	655s	36 244 KB	905326	905326	92.34%	42.85%
133	athena3	L2	20000	0	0.1	xxxxx--xx-------	325818	792	29	21h 40m	875s	57 072 KB	1464587	1464587	93.33%	48.67%

Pokračování na další straně...

ID	Pc	reg.	Počet vět	Max. it.	Delta	Části značky	Počet slov	Počet značek	Poč. it.	Doba trén.	Doba test.	Velikost modelu	Kon. poč. rysů	Úv. poč. rysů	Úsp. slov	Úsp. vět
134	athena2	L2	50000	0	0.1	xxxxx--xx-------	842463	854	33	111h 25m	1521s	54 609 KB	2924981	2924981	94.51%	54.21%
135	pcnlp6	L2	30000	0	0.1	xxxxx--xx-------	497125	822	87	68h 56m	379s	60 470 KB	1602700	1602700	93.30%	47.08%
136	pcnlp1	L2	30000	0	0.1	xxxxx--xx-------	497125	822	91	100h 43m	635s	54 674 KB	1447803	1447803	93.03%	45.45%
137	athena1	L2	30000	0	0.1	xxxxx--xx-------	497125	822	35	54h 44m	1513s	60 470 KB	1602698	1602698	93.95%	51.86%
138	athena3	L2	30000	0	0.1	xxxxx--xx-------	497125	822	35	45h 44m	979s	54 609 KB	1446867	1446867	93.69%	50.37%
139	pcnlp7	L2	30000	0	0.1	xxxxx--xx-------	497125	822	97	75h 34m	374s	51 870 KB	1351150	1351150	93.31%	47.74%
140	pcnlp5	L2	30000	0	0.1	xxxxx--xx-------	497125	822	101	92h 4m	614s	45 248 KB	1229225	1229225	93.00%	45.99%
141	pcnlp4	L2	30000	0	0.1	xxxxx--xx-------	497125	822	36	46h 28m	603s	51 812 KB	1351150	1350334	93.72%	51.22%
142	pcnlp3	L2	30000	0	0.1	xxxxx--xx-------	497125	822	38	44h 57m	642s	45 192 KB	1229225	1229225	93.61%	50.52%