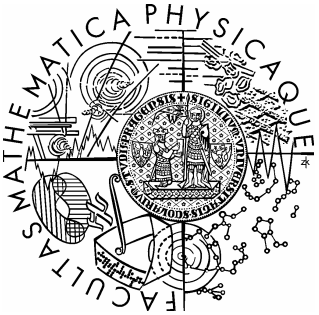
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Jan Votrubec

Volba vhodné sady rysů pro morfologické značkování češtiny

Ústav formální a aplikované lingvistiky

Vedoucí diplomové práce: Doc. RNDr. Jan Hajič, Dr.

Studijní program: informatika, počítačová a formální lingvistika

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 11. 8. 2005

Jan Votrubec

Obsah

Název práce: Volba vhodné sady rysů pro morfologické značkování češtiny
Autor: Jan Votrubec
Katedra (ústav): Ústav formální a aplikované lingvistiky
Vedoucí diplomové práce: Doc. RNDr. Jan Hajič, Dr.
e-mail vedoucího: hajic@ufal.mff.cuni.cz

Abstrakt: Tato práce navazuje na implementačně-výzkumný projekt Morče, jehož cílem bylo vytvoření co nejlepšího morfologického taggeru češtiny, založeného na skrytém Markovově modelu s průměrovaným perceptronem. Úspěšnost algoritmu závisí především na zvolené sadě rysů popisujících kontext, na jehož základě se značky vybírají. Práce stručně popisuje zvolený algoritmus a jeho implementaci. Její stěžejní část spočívá ve velké řadě provedených experimentů, které v rámci daných možností důkladně mapují možné sady rysů, jejich úspěšnosti a vztahy mezi nimi. Pro tento účel jsou definována pravidla, podle kterých se verze porovnávají. Využívá se pětinásobná crossvalidace a pro zjištění statistické významnosti výsledků je aplikován t-test. Při zahájení práce byla dána k dispozici nová data pro češtinu, takže veškeré experimenty se již prováděly nad daty z PDT 2.0. Vedlejším výsledkem práce je i statisticky významné zvýšení úspěšnosti taggeru, nicméně nejlepší tagger zřejmě překonán nebyl. Kromě ručního vývoje verzí byl projekt také upraven pro automatický vývoj, který byl v menším rozsahu proveden a popsán.

Klíčová slova: čeština, morfologie, tagger, rysy

Title: Selecting an optimal set of features for the morphological tagging of Czech
Author: Jan Votrubec
Department: Institute of Formal and Applied Linguistics
Supervisor: Doc. RNDr. Jan Hajič, Dr.
Supervisor's e-mail: hajic@ufal.mff.cuni.cz

Abstract: This work continues in implementational and experimental project Morče, which aimed to create the best possible morphological Czech tagger based on hidden Markov model with averaged perceptron. Successfulness of algorithm depends mainly on a selected set of features describing a context, which determines choice of a tag. The work describes briefly the algorithm and its implementation. Main part of the work consists of a lot of experiments which explore possible feature sets, their successfulness and their relationships. Few clear rules are defined for comparison of versions. Fivefold crossvalidation with t-test is used for verification of statistical significance. After the work was started, new Czech data became available, so all experiments used data from PDT 2.0. Side effect of this work was statistical significant improvement of successfulness. However, the best Czech tagger was obviously not overwhelmed. Some modifications were made in order to perform automatic version development. It was executed in small extent and also described.

Keywords: Czech, morphology, tagger, features

1. Úvod

Tato diplomová práce navazuje na implementačně-výzkumný projekt Morče, jehož cílem bylo vytvoření co nejlepšího morfologického taggeru češtiny. Důraz byl přitom kladen především na kvalitní implementaci. Tato práce navazuje na experimentální část projektu a podstatným způsobem ji rozšiřuje.

Cílem bylo v rámci daných možností co nejdůkladnější zmapování možných sad rysů a jejich úspěšností. K tomu bylo zapotřebí definovat jasná pravidla, podle kterých se budou verze porovnávat. Vedlejším cílem bylo také zvýšení úspěšnosti taggeru, což však nebylo podmínkou. Kromě ručního vývoje verzí byl projekt také upraven pro automatický vývoj.

Po obhájení projektu Morče byla dána k dispozici i nová data, takže ověřování úspěšností verzí a jejich uspořádání bylo již prováděno na datech z PDT 2.0.

Obsah práce sestává ze stručného popisu problému značkování, dále jsou popsány hlavní rysy algoritmu, které bezprostředně souvisí s volbou rysů, a je krátce popsán i způsob implementace a pracovní postup. Stěžejní část práce však tvoří popis vývoje verzí – kapitola 7 se zabývá automatickým vývojem a kapitola 8 vývojem ručním, který se zatím ukázal jako nejpodstatnější.

Změny, které byly v kódu provedeny oproti obhájené verzi projektu, jsou uvedeny v Příloze B.

2. Morfologické značkování

Čeština, podobně jako některé další slovanské jazyky, je známa složitostí své morfologie. Její počítačové zpracování je však komplikováno velkým výskytem homonymních slovních tvarů. Tak například slovní tvar „podle“ může být předložkou („šel podle zdi“), ale také příslovcem („choval se podle“). Homonymie se projevuje také ve velké míře v koncovkách – tvar „nehty“ může být 1., 4., 5. nebo 7. pád plurálu. Další zpracování textu (jako například automatický překlad, syntaktická analýza a podobně) zpravidla vyžaduje jednoznačné určení, který základní slovní tvar (tzv. lemma) a které gramatické kategorie (tzv. gramatická značka) přísluší danému slovnímu tvaru.

Pro češtinu se používá standardizovaný patnáctipoziční značkovací systém[1]. Každá pozice reprezentuje určitou gramatickou kategorii – například slovní druh, pád, rod, číslo... Přesný popis značkovacího systému je uveden v [5].

Jednomu slovnímu tvaru bez znalosti kontextu tedy může příslušet více gramatických značek. Teprve na základě znalosti kontextu lze vybrat značku jedinou.

Morfologické značkování se skládá ze dvou částí – jednak přiřazení všech možných značek slovním tvarům a jednak jednoznačný výběr správné značky v daném kontextu. Zadáním projektu Morče byla právě druhá část, též zvaná desambiguace.

Vstupem algoritmu je tedy předzpracovaný český text, obsahující pro každé slovo všechny možné gramatické značky a jim odpovídající lemmata. Pro natrénování a otestování úspěšnosti obsahují data navíc ručně (anotátorem) přiřazenou správnou značku a lemma. Výstupem je stejný text zachovávající veškeré vstupní informace a obsahující navíc značku přiřazenou naším algoritmem.

[1] ve skutečnosti je ovšem využito pouze třináct pozic

Příklad vstupních dat

Vezměme si větu „Na tři hlavní podezřelé byla uvalena vyšetřovací vazba.“ Následující schéma ukazuje možná lemmata a příslušné morfologické značky pro každé slovo této věty.

Na	tři	hlavní	podezřelé	byla	uvalena	vazba
Na-1	tři`3	hlavní	Podezřelý	být	uvalit	vazba-1
RR--4----------	ClXP4----------	AAFS2----1A----	AAFS2----1A----	VpQW---XR-AA---	VsQW---XX-AP---	NNFS1-----A----
RR--6----------	ClXP1----------	AAFP1----1A----	AAFP1----1A----			
	ClXP5----------	AAFP4----1A----	AAFP4----1A----			
		AAFP5----1A----	AAFP5----1A----			
	tříť	AAFS3----1A----	AAFS3----1A----			
	Vi-S---2--A----	AAFS6----1A----	AAFS6----1A----			
	Vi-S---3--A---4	AAIP1----1A----	AAIP1----1A----			
		AAIP4----1A----	AAIP4----1A----			
		AAIP5----1A----	AAIP5----1A----			
		AAMP4----1A----	AAMP4----1A----			
		AANS1----1A----	AANS1----1A----			
		AANS4----1A----	AANS4----1A----			
		AANS5----1A----	AANS5----1A----			
		AAFS1----1A----				
		AAFS5----1A----				
		AANP1----1A----				
		AANP4----1A----				
		AANP5----1A----				
		AAFS4----1A----				
		AAFS7----1A----				
		AAIS1----1A----				
		AAIS4----1A----				
		AAIS5----1A----				
		AAMP1----1A----				
		AAMP5----1A----				
		AAMS1----1A----				
		AAMS5----1A----				
		hlaveň				
		NNFP2-----A----				
		NNFS7-----A----				

Schéma 1. Příklad morfologicky předzpracovaného textu

Z příkladu je patrno několik typických homonymií. Předložka „na“ se může pojit s 1. a 4. pádem, „tři“ může být číslovka i sloveso, „hlavní“ má 27 možností jako přídavné jméno a navíc další dvě jako podstatné jméno „hlaveň“. Pro slovní tvar „podezřelé“ existuje 13 možností kvůli homonymii v některých pádech rodech a číslech.

Algoritmus přiřazuje každému slovu jednoznačnou značku a lemma. Pro uvedený příklad by tedy správný výstup měl vypadat takto:

Na	tři	Hlavní	Podezřelé	byla	uvalena	vazba
na-1	tři`3	Hlavní	Podezřelý	být	uvalit	vazba-1
RR--4----------	ClXP4----------	AAMP4----1A----	AAMP4----1A----	VpQW---XR-AA---	VsQW---XX-AP---	NNFS1-----A----

Schéma 2. Správně označkovaný text

Přestože přesné zadání vyžaduje pouze přiřazení správné morfologické značky, algoritmus vybírá i lemma. Lemma je vybíráno triviálním způsobem – vezme se první, které odpovídá vybrané značce.

Všechny úspěšnosti označkování jsou uváděny pouze pro shodu ve značkách.

3. Popis algoritmu

Pro řešení úkolu byla použita statistická učící se metoda, založená na kombinaci skrytého Markova modelu (HMM) a průměrovaného perceptronu (PP), popsaná v článku Michaela Collinse [1].

3.1. HMM

Skrytý Markovův model se obvykle používá tam, kde je třeba jednu sekvenci informací převádět na jinou, přičemž lze předpokládat, že tento převod je určen pouze historií omezené délky. Zde dochází k převodu sekvence slov na sekvenci morfologických značek.

Pro trénování a následné použití tohoto modelu se používá Viterbiho algoritmus, který je detailně popsán například v [2]. Tento algoritmus slouží k nalezení nejlépe ohodnocené výstupní sekvence pro zadanou vstupní sekvenci.[1]

Nalezení nejlépe ohodnocené cesty mezi všemi možnými vede obecně na vyzkoušení všech permutací. To je zřejmě neaplikovatelně náročná metoda. V lingvistice se osvědčil právě trigramový HMM, který uvažuje historii značek dvě pozice zpět a pozici aktuální (právě tento rozsah byl ověřen jako nejlepší kompromis mezi výpočetní náročností a úspěšností). Ostatní pozice považuje pro danou větev výpočtu za již určené. Vzhledem k tomu, že metoda pracuje statisticky, může se stát, že algoritmus někdy nevybere obecně nejlepší cestu. Přesto se trigramový model ukázal jako nejvhodnější.

Viterbiho algoritmus se zpravidla aplikuje na HMM, kde ohodnocením jednotlivých přechodů HMM jsou pravděpodobnosti. Pro použití Viterbiho algoritmu však postačí, aby pro každou dvojici prvků z množiny možných ohodnocení existovalo supremum. Algoritmus Morče používá váhové koeficienty, které sice nejsou pravděpodobnostmi, ale tuto podmínku splňují. Ohodnocením jsou buď celá čísla (ve fázi trénování) nebo čísla reálná (ve fázi testování).

3.2. Ohodnocení přechodů HMM: Průměrovaný perceptron

To, čím je koncepce popsaná v [1] nová, je algoritmus pro ohodnocování přechodů mezi jednotlivými stavy HMM. Jedná se o tzv. „průměrovaný perceptron“, který je ve své podstatě jednoduchý (což zaručuje přijatelnou časovou náročnost), ale dává přitom

[1] Ve skutečnosti Viterbi hledá nejpravděpodobnější vstupní sekvenci pro danou výstupní sekvenci. Z hlediska použití jsou však termíny vstupní a výstupní sekvence obrácené.

pozoruhodně dobré výsledky. Průměrovaný perceptron dosud nebyl pro morfologické značkování češtiny implementován.

V experimentální části projektu pak bylo třeba zvolit vhodné vstupní parametry perceptronu, na jejichž základě se určí výsledné ohodnocení přechodů mezi stavy HMM. Tyto vstupní parametry jsou závislé jak na označkovaní předcházejícího textu, tak na informacích o textu, které na předchozím označkování nezávisí. Jsou to tak zvané rysy.

3.3. Rysy

Rysy slouží k popisu dané situace v textu, určují parametry, které se budou v textu sledovat při trénování a testování. Použitým rysům pak odpovídají příslušné váhové koeficienty, s nimiž pracuje průměrovaný perceptron.

Obecně může být rys libovolně složitý a používat libovolné informace o textu, které jsou obsaženy ve vstupních datech. Rysem je například značka na aktuální pozici, kombinace aktuálního slova s aktuální značkou či pořadí slova ve větě.

3.3.1. Příklady rysů

Pro ukázku použijeme opět stejnou ukázku správně označkovaného textu:

Na	tři	Hlavní	Podezřelé	byla	uvalena	vazba
na-1	tři`3	Hlavní	Podezřelý	být	uvalit	vazba-1
RR--4----------	ClXP4----------	AAMP4----1A----	AAMP4----1A----	VpQW---XR-AA---	VsQW---XX-AP---	NNFS1-----A----

Schéma 3. Správně označkovaný text

Je-li algoritmus při čtení vstupního textu na třetím slově z příkladu („hlavní“), pak možné rysy pro tuto pozici jsou:

- aktuální značka je `AAMP4----1A----` (značkový unigram)
- předcházející značka je `ClXP4----------` a aktuální značka je `AAMP4----1A----` (značkový bigram)
- aktuální slovo je „hlavní“ a aktuální značka je `AAMP4----1A----`
- aktuální slovo je třetí ve větě a aktuální značka je `AAMP4----1A----`
- aktuální slovo začíná malým písmenem a aktuální značka je `AAMP4----1A----`
- lemma nejbližšího následujícího možného slovesa je „být“ a aktuální pád je 4

Konkrétní rys může být pro danou pozici buď pravdivý (pokud popisuje situaci, která pro aktuální pozici nastává), nebo nepravdivý. Pro pravdivé rysy se pak při ohodnocování přechodu v HMM uplatní jejich váhové koeficienty.

Typicky rys obsahuje nějakou informaci o značce na aktuální pozici v textu, takže jej lze chápat také jako předpověď značky na základě určitého kontextu. Tato předpověď ovšem může být i částečná (například předpověď pádu). Pokud by neobsahoval žádnou informaci o aktuální značce, ztrácí svůj význam, protože všechny značky pak předpovídá se stejnou vahou. Uvedené příklady rysů by pak bylo možno interpretovat takto:

- aktuální značka je `AAMP4----1A----` (unigram)
- pokud je předcházející značka `ClXP4----------`, pak aktuální značka je `AAMP4----1A----` (bigram)
- pokud je aktuální slovo „hlavní“, pak aktuální značka je `AAMP4----1A----`
- pokud je aktuální slovo třetí ve větě, aktuální značka je `AAMP4----1A----`
- pokud aktuální slovo začíná malým písmenem, pak aktuální značka je `AAMP4----1A----`
- pokud je lemma nejbližšího následujícího možného slovesa „být“, pak aktuální pád je 4

Pro daný přechod v HMM (tedy pro výběr značky pro aktuální pozici) se uplatní všechny rysy, které jsou pravdivé. Určení a použití váhových koeficientů jednotlivých rysů je dáno průměrovaným perceptronem a bude popsáno dále.

3.3.2. Termíny

Pro další text definujeme následující termíny:

elementární rys – elementárním rysem rozumíme základní a dále nedělitelnou část rysu, popisující určitý kontext. Elementární rys obsahuje informaci o pozici, ke které se vztahuje, o druhu informace, kterou popisuje, a konkrétní hodnotu této informace (viz 4.2.).

složený rys – vznikne složením dvou a více elementárních rysů. Složený rys je platný právě tehdy, když jsou platné všechny elementární rysy, z nichž se skládá.

předpověď rysu – část nebo části rysu, které se vztahují k aktuální značce. Nejčastěji to bývá právě číselný kód aktuální značky, ale mohou to být také pouze části značky (např. samostatně pád).

kontext rysu – část rysu, která neobsahuje žádnou předpověď pro aktuální značku. Každý rys se tedy skládá z kontextu a předpovědi.

3.4. Učící se algoritmus: průměrovaný perceptron

Váhy přechodů mezi jednotlivými stavy HMM ve Viterbiho algoritmu jsou definovány jako součet váhových koeficientů příslušných rysů v daném textu při zvoleném výběru značek.

Definujeme n jako počet všech rysů, H množina všech historií, T množina všech značek, $\varphi: HxT \rightarrow \{0,1\}^n$ je vektorová funkce, jejíž jednotlivé složky ukazují, které rysy při dané historii a zvolené značce jsou pravdivé. Je-li tedy například 150. rys definován jako „pokud je předcházející značka `ClXP4----------`, pak aktuální značka je `AAMP4----1A---`“, bude $\varphi(h,t)_{150}=1$ všude tam, kde je tato definice splněna, všude jinde bude $\varphi(h,t)_{150}=0$.

Dále definujeme váhový vektor α jako prvek $\boldsymbol{Z}^n$. Každému rysu přísluší celočíselný váhový koeficient α_n.

Formálně lze tedy funkci, která ohodnotí daný přechod mezi stavy HMM zapsat jako:

$$w(h,t) = \alpha.\varphi(h,t) = \sum_{i=1..n} \alpha_i.\varphi(h,t)_i$$

3.4.1. Trénování

Na začátku jsou váhové koeficienty všech rysů nastaveny na nulu. V několika iteracích se procházejí celá vstupní data. Viterbiho algoritmus postupně vybírá nejlepší cestu (tj. nejlepší značky) pro každou větu s použitím aktuálních váhových koeficientů. Po dokončení každé věty dojde k aktualizaci váhových koeficientů. To se opakuje, dokud není dosaženo požadovaného počtu průchodů vstupními daty.

3.4.2. Aktualizace váhových koeficientů

Pro rysy odpovídající dané větě a algoritmem vybraným značkám jsou příslušné váhové koeficienty sníženy o 1. Pro rysy odpovídající dané větě a správným značkám jsou

příslušné váhové koeficienty zvýšeny o 1. Při správném označkování věty tedy zůstávají koeficienty nezměněny.

Délku věty označíme *d*. Definujeme globální rysový vektor (historie *h* a označkování *t* jsou závislé na pozici ve větě *k*):

$$\Theta = \sum_{k=1..d} \varphi(h,t)$$

Každá složka pak představuje počet výskytů rysu při zvoleném označkování a zvolené historii. Správnou historii a správné označkování označíme h', t' a příslušný globální rysový vektor Θ'. Algoritmem vybranou historii a značky označíme h'', t'' a příslušný globální rysový vektor Θ''.

Nyní můžeme formálně aktualizaci váhového vektoru zapsat takto:

$$\alpha' = \alpha + \Theta' - \Theta''$$

3.4.3. Test (použití)

Test se v zásadě neliší od jedné iterace trénování – nejlepší cesta se opět vybírá Viterbiho algoritmem po větách. Nedochází však již k aktualizaci váhových koeficientů.

Ve vylepšené variantě algoritmu se navíc při testu používají takzvané „průměrované váhové koeficienty“, které potlačují oscilace koeficientů a zlepšují úspěšnost algoritmu (viz [1]). Alfy už nejsou celočíselné, ale reálné.

Jednoznačné přiřazení morfologických značek vstupnímu textu je tedy v projektu Morče podmíněno třemi kroky:

a) volba vstupních parametrů Markovova modelu (výběr sady rysů)

b) natrénování Markovova modelu (určení váhových koeficientů)

c) spuštění natrénovaného modelu na neznámých datech

4. Implementace

Celý projekt byl naprogramován v jazyce C na linuxové platformě. Všechny části fungují výhradně v textovém režimu. Práce na projektu sestávala ze dvou částí – implementační (s cílem co nejkvalitněji naprogramovat popsaný algoritmus) a experimentální (s cílem nalézt parametry dávající nejlepší výsledky). Rozšíření experimentální části a její zevrubný popis je právě hlavním tématem této práce.

Následující popis zachycuje pracovní postup experimentátora tak, jak byl implementován. Detailní popis všech programů a skriptů pro používání všech funkcí projektu je uveden v [4].

4.1. Pracovní postup při experimentu

Aby byla zachována co největší modularita jednotlivých částí projektu, byl pracovní postup při experimentech rozdělen na několik kroků:

1. volba sady rysů (přesněji volba použitých typů rysů)
2. vytvoření seznamu platných rysů z trénovacích dat
3. odfiltrování rysů s příliš malým počtem výskytů v datech
4. vytvoření konečného automatu ze seznamu rysů pro správu odpovídajících váhových koeficientů
5. natrénování váhových koeficientů nad trénovacími daty ve zvoleném počtu iterací
6. otestování (aplikace) natrénovaného modelu na testovacích datech (pro všechny iterace)
7. vyhodnocení úspěšnosti na testovacích datech (pro všechny iterace)
8. analýza chyb

4.1.1. Volba sady rysů

Pojem typ rysu označuje schéma, na jehož základě je utvořena celá řada rysů s konkrétními hodnotami. Tak například typ rysu může být „předchozí značka a aktuální značka“. Z tohoto schématu se pak na základě trénovacích dat vytvoří konkrétní rysy, například: „pokud je předcházející značka `ClXP4----------`, pak aktuální značka je `AAMP4----1A----`“.

Typy rysů, které se použijí, jsou jednoznačně určeny částí kódu, která zabezpečuje vygenerování všech platných rysů nad aktuální historií a označkováním. To je obsaženo

v souboru *generator.c* a *feature_const.h*, což jsou jediné části celého projektu, kterými se jednotlivé experimentální verze od sebe liší. *generator.c* obsahuje přímo kód, který rysy generuje, *feature_const.h* obsahuje konstanty, které blíže určují některé vlastnosti dané verze (například maximální vzdálenost nejbližšího slovesa a podobně).

Pro další text zde zavedu dva termíny:

verze – je jednoznačně určena dvojicí souborů *generator.c* a *feature_const.h*. Verzí je jednoznačně určen typ rysů, který se bude používat, včetně jejich syntaxe a dodatečných parametrů.

následník verze – je verze, která vznikne ze svého předchůdce přidáním nebo změnou některých rysů. Bezprostřední následník zpravidla vznikne přidáním jediného rysu, případně pozměněním jediného rysu.

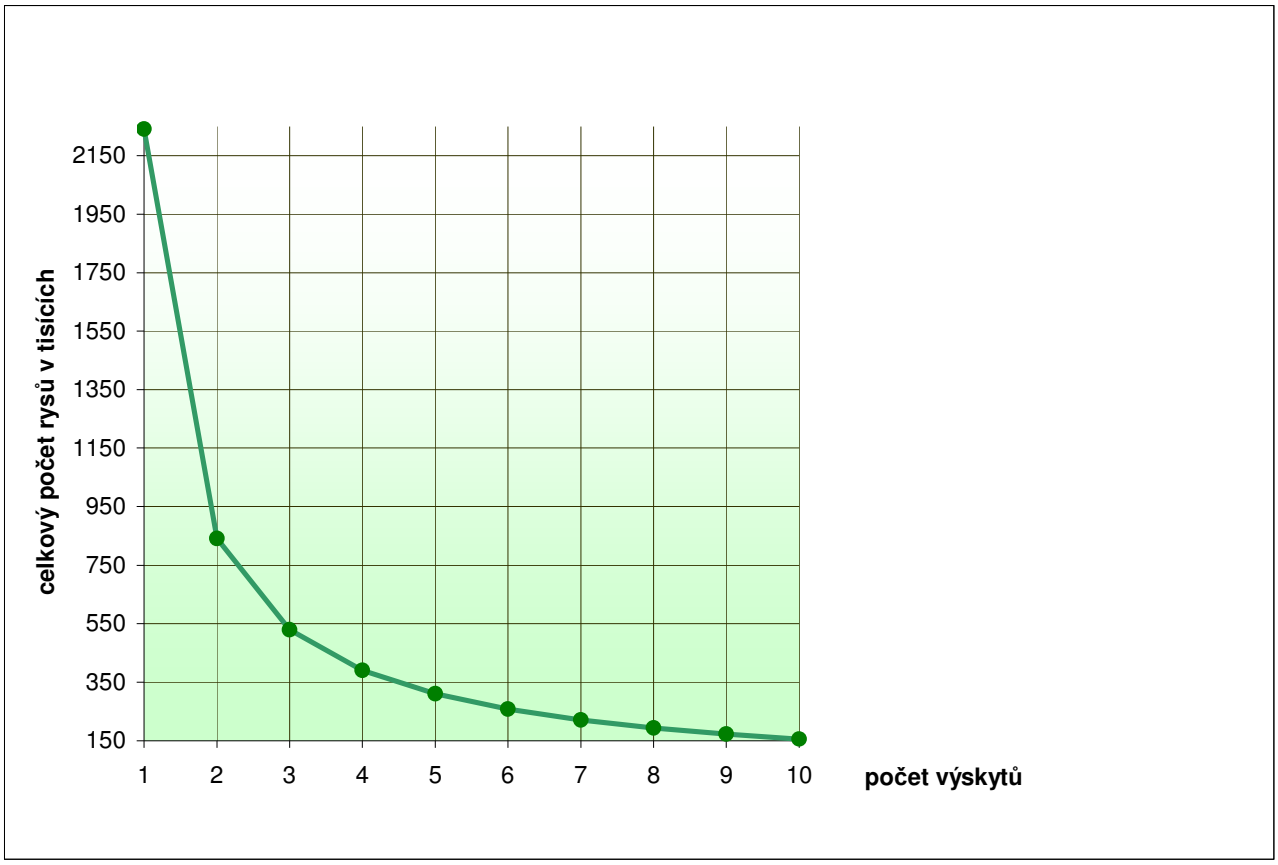
Použitá syntax pro označení rysů je uvedena samostatně na konci této kapitoly.

4.1.2. Vytvoření seznamu platných rysů z trénovacích dat

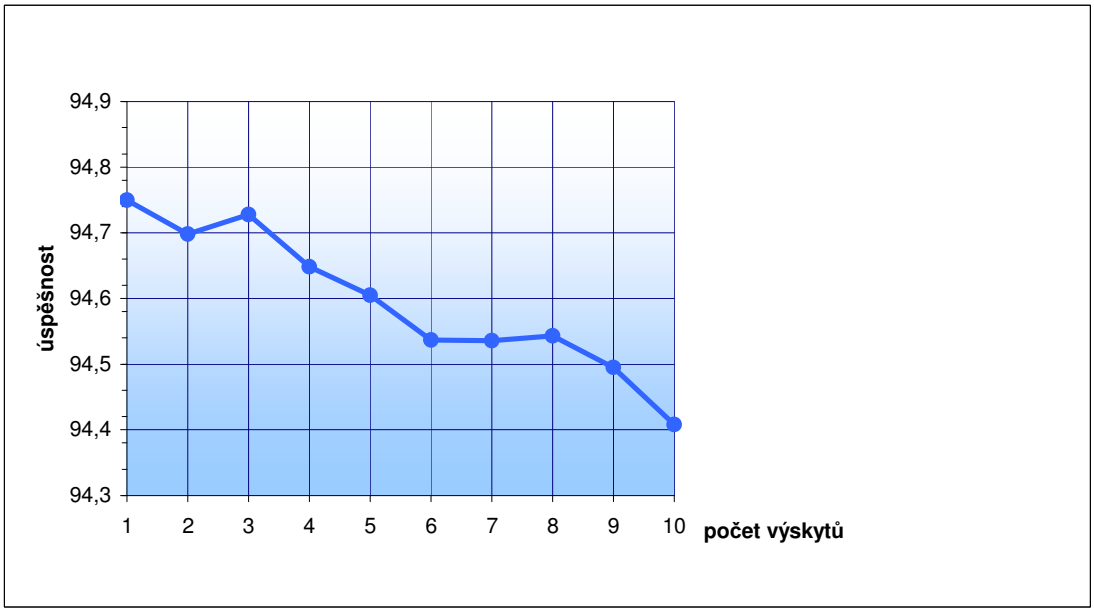
Pomocí programu make_ftrs se vytvoří seznam platných rysů z celých trénovacích dat. Vznikne textový soubor, kde každý řádek obsahuje signaturu jednoho rysu v pořadí, jak byly vygenerovány. Každý rys je v souboru tolikrát, kolikrát se vyskytl v trénovacích datech. Tento soubor je možno dodatečně upravovat.

4.1.3. Odfiltrování rysů s příliš malým počtem výskytů v datech

Zpravidla bývá vhodné nepoužít všechny rysy vytvořené z trénovacích dat, ale jen takové, které se v nich vyskytly vícekrát. Podstatně se tím snižuje jejich počet a tím se zmenšuje a zrychluje automat pro správu vah. Závislost celkového počtu rysů a závislost úspěšnosti na jejich minimálním počtu výskytů ukazují následující grafy (verze *lucifer4*).



Graf 1. Závislost celkového počtu rysů filtrovaných podle minimálního počtu výskytů



Graf 2. Závislost úspěšnosti na filtrování rysů podle minimálního počtu výskytů

Z uvedeného příkladu je patrné, že odfiltrování rysů, které mají méně než tři výskyty nesnižuje úspěšnost nijak zásadně, ale paměťová (a částečně i časová) úspora je výrazná. V ostatních verzích vypadala situace obdobně, někdy dokonce odfiltrování rysů s méně jak třemi výskyty celkovou úspěšnost zvedlo. Pro všechny experimenty bylo tedy použito toto filtrování.

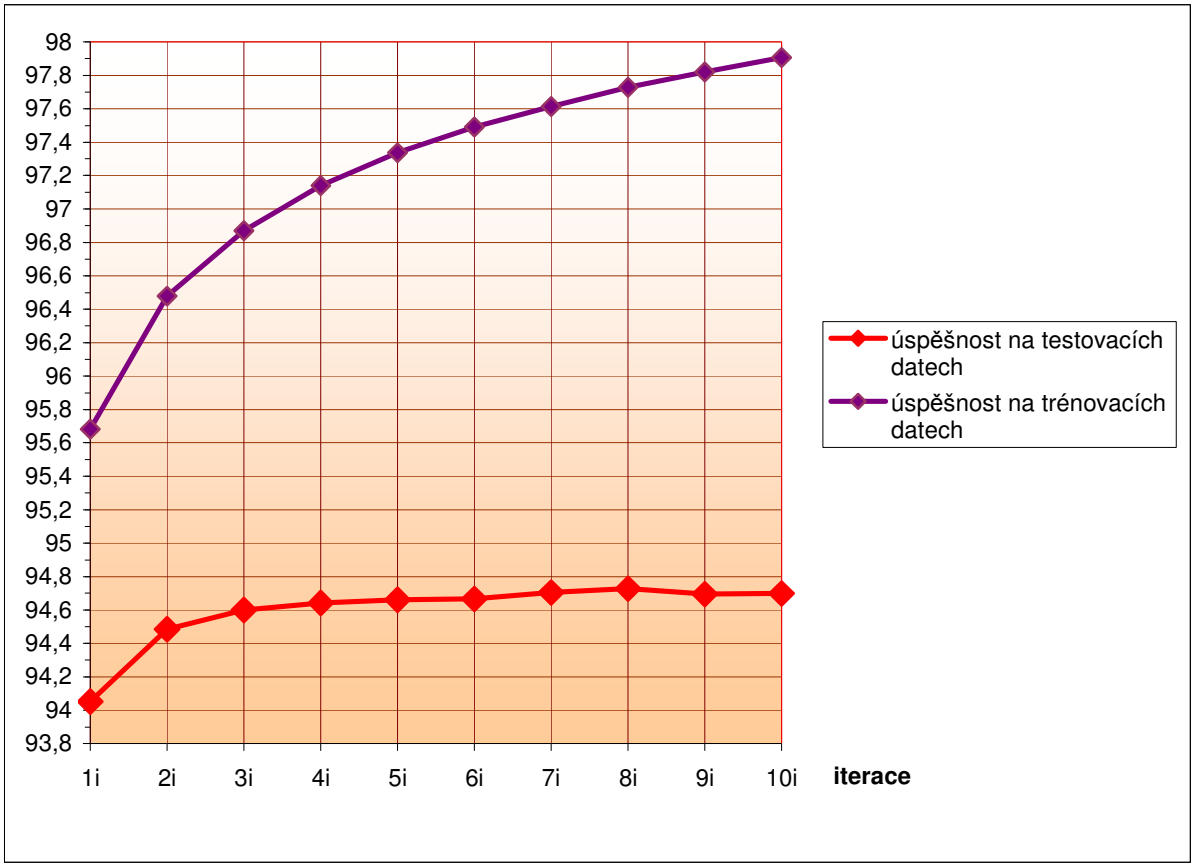
Filtrování rysů se provádí programem filter_ftrs. Jeho vstupem je textový soubor se seznamem rysů. Tento soubor musí být neunikátně setříděný, obsahuje tedy každý rys tolikrát za sebou, kolikrát se vyskytl v datech (po případných úpravách). Výstupem je opět soubor, který obsahuje pouze rysy, jejichž počet výskytů byl větší nebo roven zadané mezi. Každý rys se vypisuje pouze jednou.

4.1.4. Vytvoření konečného automatu ze seznamu rysů pro správu odpovídajících váhových koeficientů

Programem make_fsa se z textového seznamu rysů (každý rys na samostatném řádku) vytvoří konečný automat, který spravuje váhové koeficienty zadaných rysů. Výstupem je soubor s interní reprezentací automatu.

4.1.5. Natrénování váhových koeficientů nad trénovacími daty ve zvoleném počtu iterací

Trénování se spouští programem train. Postupně se procházejí celá trénovací data a podle popsaného algoritmu se aktualizují váhové koeficienty. Po každé iteraci se uloží dva soubory s váhovými koeficienty (neprůměrované a kumulované pro výpočet průměrovaných), úspěšnost je totiž testována po každé iteraci. Průběh úspěšnosti na trénovacích a testovacích datech ukazuje následující graf. Počet iterací pro trénování byl experimentálně stanoven na 10. Ukázalo se, že úspěšnost na testovacích datech dosáhne maxima v průměru kolem páté iterace (v naprosté většině verzí mezi 4 a 8 iterací) a s dalšími iteracemi klesá. Obecně platí, že čím je verze složitější (čím více typů rysů používá), tím více iterací je zapotřebí.



Graf 3. Průběh úspěšnosti na trénovacích a testovacích datech podle počtu iterací (verze lucifer4)

4.1.6. Otestování (aplikace) natrénovaného modelu na testovacích datech (pro všechny iterace)

Testování se provádí programem test. Ten projde jednou celá testovací data a použije zadané váhové koeficienty pro jednoznačný výběr značek a lemmat. Testování se provádí pro každou iteraci, přičemž pro srovnání verzí uvažujeme nejlepší výsledek.

4.1.7. Vyhodnocení úspěšnosti na testovacích datech (pro všechny iterace)

Vyhodnocení úspěšnosti se provádí programem eval. Ten spočítá celkovou úspěšnost v určování značek. Jeho výstupem jsou také podrobnější informace – chybovost na jednotlivých pozicích značky a chybové matice, zobrazující počty záměn pro každou pozici značky. Výstup evaluace a příklad konfúzní matice jsou na následujících schématech.

```
Train data: t2, Generator: lucifer4, Automat: f3, Iteration: 10, Test data:
s2


Global result:                          93.474503%

Tag successfulness:                     94.452003%
Lemmas successfulness:                  97.458504%
POS successfulness:                     98.392502%
SUBPOS successfulness:                  98.323502%
GENDER successfulness:                  97.496498%
NUMBER successfulness:                  97.757004%
CASE successfulness:                    95.944000%
POSGENDER successfulness:               99.973000%
POSNUMBER successfulness:               99.987503%
PERSON successfulness:                  99.912498%
TENSE successfulness:                   99.945999%
GRADE successfulness:                   99.630997%
NEGATION successfulness:                98.789497%
VOICE successfulness:                   99.945999%
VAR successfulness:                     99.369003%
```

Schéma 4. Příklad výpisu programu eval – úspěšnosti pro jednotlivé pozice značky (verze lucifer4)

CASE confusion matrix, fifth position in tag:

VŠE	1	2	3	4	5	6	7	X	-
1	26747	248	35	1536	25	23	27	177	51
2	409	26689	54	412		58	35	89	30
3	27	29	4624	24		40	37	9	1
4	1209	277	38	19715	2	184	80	20	30
5	40	3	1	1	12	2		15	9
6	27	53	30	246		16706	2	27	7
7	23	23	30	53			8180	9	16
X	63	40	1	10		1	10	4462	52
-	785	165	22	92	1	36	44	957	84753

Sum:200000

Schéma 5. Příklad konfúzní matice pro pád (verze lucifer4)

Červeně (v prvním sloupci konfúzní matice) jsou hodnoty vybrané Morčetem, modře (v prvním řádku) jsou hodnoty správné. Čísla v tabulce pak vyjadřují počet záměn na testovacích datech. Na diagonále jsou čísla, která představují počet správně určených hodnot.

4.1.8. Analýza chyb

Pro vývoj dalších verzí je zapotřebí detailní znalost chyb, kterých se stávající verze dopustila. Proto bylo vyvinuto webové rozhraní, které poskytuje přehled všech dosud natrénovaných a otestovaných verzí. Pro každou záměnu (jednu položku chybové matice) je pak možné zobrazit konkrétní příklady chyb z testovacích dat. Ukázka chybových příkladů je na následujícím schématu.

1.)	masem	Bonn	-	**Sedm**	Vietnamců	,	kteří
2.)	a	Číňany	a	**některé**	z	nich	potom
3.)	zbraně	?	-	**Něco**	nám	řekli	jejich
4.)	střílet	?	-	**Samopal**	.	Střílet	jsem
5.)	.	a	nemůžete	**to**	vypudit	z	hlavy
6.)	do	lesů	na	**druhej**	břeh	řeky	Vuka
7.)	dostat	a	dostal	**to**	do	ramena	.
8.)	a	hodil	dovnitř	**granát**	.	Potom	jsme
9.)	viděl	?	-	**Lidi**	normálně	roztrhaný	na
10.)	-	Lidi	normálně	**roztrhaný**	na	kusy	.
11.)	leteckejma	kulometama	a	**sedm**	našich	kluků	zranili
12.)	jsem	,	jak	**to**	koupil	auťák	a

Schéma 6. Příklad chybového výpisu (záměna 4. a 1. pádu ve verzi lucifer4)

Červeně jsou zvýrazněna všechna slova, kde byla značka určena chybně, prostřední modrý sloupec ukazuje výskyt zvoleného typu chyby (zde záměna 4. a 1. pádu). Pro každý příklad je ještě možno zobrazit detailní informaci, která v podstatě ukazuje všechny informace, z nichž je možné vyrábět rysy – příklad opět ukazuje následující schéma.

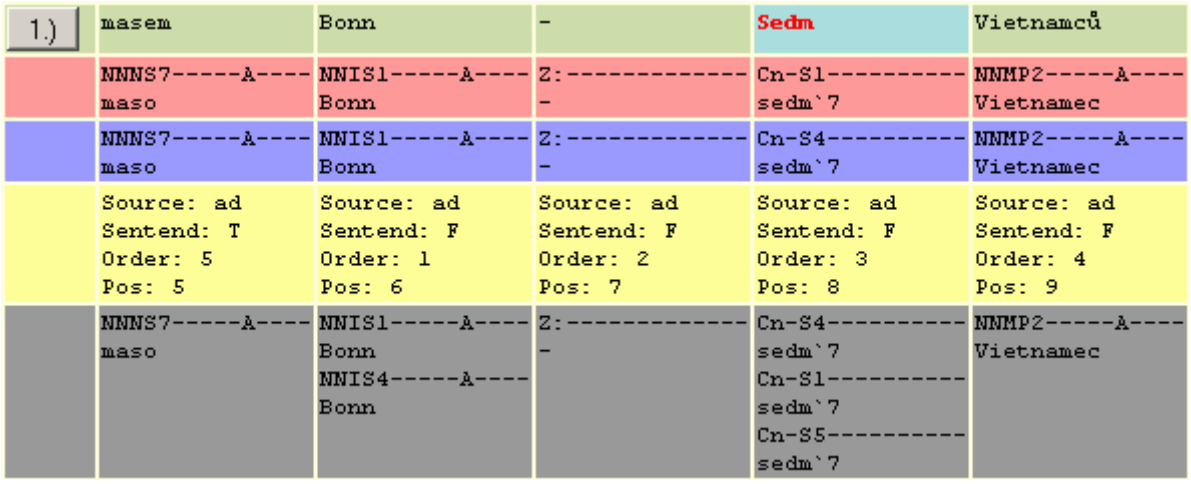


Schéma 7. Příklad chybového výpisu (záměna 4. a 1. pádu ve verzi lucifer4) s detailní informací o vybrané větě

V červeném řádku je zobrazena značka a lemma vybrané Morčetem, v modrém řádku správná značka a lemma, ve žlutém řádku dodatečné informace o slovu (zdroj možných značek, příznak konce věty, pořadí slova ve větě, pořadí slova v textu), v šedém řádku možné značky a příslušná lemmata.

4.2. Syntax signatur rysů

Aby rysy bylo možné nějak rozumně zpracovávat, byl zvolen celkem jednoduchý formalismus. Rysy jsou reprezentovány znakovými řetězci, tzv. signaturami.

Následující popis syntaxe signatur rysů není nijak závazný pro další vývoj. Obecně je projekt naprogramován tak, že signaturami mohou být libovolné znakové řetězce. Uvedený formalismus popisuje standard, který byl používán při vývoji verzí a na nějž se budu dále v textu odkazovat.

Signatury elementárních rysů sestávají z následujících částí: ***{pozice}{druh}{hodnota}***. Pozice a druh jsou jednoznakové, počet znaků hodnoty je dán druhem. Možné hodnoty jednotlivých částí ukazují následující tabulky.

označení pozice	**význam**
|	aktuální (předpovídaná)
A	1 zpět
B	2 zpět
a dále zpět analogicky	
a	1 vpřed
b	2 vpřed
a dále vpřed analogicky	
V	nejbližší levé sloveso
U	nejbližší možné sloveso vpravo
<	nejbližší část značky vpravo

Tabulka 1. Označení pozice rysu

druh a hodnota	**význam druhu**	**význam hodnoty**
T{n}	značka	číslo značky (max. čtyřmístné číslo)
W{n}	slovní forma	číslo slova (max. pětimístné číslo)
L{n}	lemma	číslo lemmatu (max. pětimístné číslo)
O{n}	pořadí slova ve větě	pořadí slova ve větě
C{x}	pád	znak ze značky na pozici pádu (5. pozice)
R{n}	reflexivita (předchozí výskyt „se“)	0 – zvratné „se“ nenalezeno 1 – zvratné „se“ nalezeno
U{n}	velikost písmen slova	0 - první malé 1 - první velké 2 - druhé velké
V{n}	velikost písmen lemmatu	0 - první malé 1 - první velké 2 - druhé velké
R{a}{b}	dvojice slovní poddruh – pád	a: znak ze značky na pozici poddruhu (2.pozice) b: znak ze značky na pozici pádu (5.pozice)
Q{a}{b}	dvojice slovní poddruh – číslo	a: znak ze značky na pozici poddruhu (2.pozice) b: znak ze značky na pozici čísla (4.pozice)
X{a}{b}	dvojice rod – pád	a: znak ze značky na pozici rodu (3.pozice) b: znak ze značky na pozici pádu (5.pozice)

Tabulka 2. Označení druhu a hodnoty rysu

Jak je uvedeno v tabulce, místo konkrétních hodnot značek a slovních tvarů se používají jejich čísla, která jim jsou přiřazena již při zpracovávání vstupního textu. Díky tomu jsou signatury rysů podstatně kratší.

Některé další parametry rysu – například omezení, pro které značky se má nebo nemá generovat, jak daleko vyhledávat nejbližší slovesa – závisí na konkrétní verzi generátoru.

Kombinací elementárních značek vznikají složitější. Jejich pořadí není sice jednoznačně předepsáno, ale jednoznačnost je dána použitým generátorem.

Protože v dalším textu půjde především o typy rysů, nikoliv o konkrétní hodnoty, kterých nabývají, používá se pro označení typů stejná syntax, pouze *{hodnota}* se vynechává. Tak například označení AT|T představuje celou množinu rysů sestávající z kombinace „předchozí značka, aktuální značka“. Konkrétní rysy jsou pak vytvořeny na základě dat. Pokud má rys nějaké další parametry (například vzdálenost, do jaké se má vyhledávat nejbližší sloveso), uvádí se hodnota tohoto parametru v závorce a v textu je jeho význam vysvětlen. Tak například VL(20) znamená „značka nejbližšího slovesa vlevo do vzdálenosti 20 pozic“.

Protože téměř všechny rysy předpovídají celou aktuální značku, v dalším textu se kvůli přehlednosti aktuální značka neuvádí. Není-li tedy uvedena žádná předpověď, je na konci rysu vynecháno |T.

Příklad rysů

Na nejúspěšnější verzi projektu nyní ukážu použité typy rysů a jejich konkrétní hodnoty. Použiji opět stejný příklad. Modře jsou označeny již vybrané značky. V tomto kroku potřebujeme zvolit značku pro slovo „podezřelé“ (pro zvolenou historii). Konkrétní rysy budou vygenerovány pro předpověď značky `AAMP4----1A----`.

Na	tři	hlavní	podezřelé	byla	uvalena	vazba
RR--4----------	ClXP4----------	AAMP4----1A----	AAMP4----1A----			
			AAFS2----1A----	VpQW---XR-AA---	VsQW---XX-AP---	NNFS1-----A----
			AAFP1----1A----			
			AAFP4----1A----			
			AAFP5----1A----			
			....			

Schéma 8. Příklad značkované věty

značka nebo slovo	**číslo**
`--------------` (žádná značka)	-1
`AAMP4----1A----`	126
`ClXP4----------`	134
(žádné slovo)	-1
hlavní	56
podezřelé	212
tři	344

Tabulka 3. Výběr z tabulek značek a slovních tvarů

typ rysu	**popis**	**konkrétní hodnota (signatura rysu)**
|T	aktuální značka (dále *)	|T126
AT|T	předchozí značka a * (značkový bigram)	AT126|T126
BTAT|T	značkový trigram	BT134AT126|T126
BT|T	značka dvě pozice zpět a *	BT134|T126
|W|T	aktuální slovo a *	|W212|T126
AW|W|T	slovní bigram a *	AW56|W212|T126
AW|T	předchozí slovo a *	AW56|T126
BW|T	slovo dvě pozice zpět a *	BW344|T126
|O|T	pořadí slova ve větě a *	|O4|T126
AC|T	předchozí pád a *	AC4|T126
|U|T	velikost písmen slova a *	|U0|T126
|V|T	velikost písmen lemmatu a *	|V0|T126
VT|T	značka nejbližšího slovesa a *	VT-1|T126
VL|T	lemma nejbližšího slovesa a *	VL-1|T126
VL|C	lemma nejbližšího slovesa a aktuální pád	VL-1|C4

Tabulka 4. Ukázka rysů vygenerovaných pro jeden přechod stavů HMM verzí lucifer4

5. Uspořádání verzí

Jedním ze zásadních problémů, které je třeba vyřešit před samotnou experimentální částí, je definice nějakého uspořádání verzí. V počáteční fázi vývoje byla používáno pouze srovnání úspěšností na testovacích datech. To sice vedlo k poměrně dobrým výsledkům, nicméně pro detailnější práci bylo třeba stanovit další kritéria, aby se minimalizovala možnost zlepšení způsobeného pouhou náhodou.

5.1. Měření úspěšnosti

Úspěšnost značkování je definována zcela standardním způsobem:

$Acc(S) = 1 - \frac{Error(S)}{|S|}$, kde $|S|$ je počet slov, $Error(S)$ je počet chybně určených značek a $Acc(S)$ je výsledná úspěšnost.

5.2. Crossvalidace

Obecně lze říci, že ručně označkovaných dat pro trénování a testování je vždycky málo. Crossvalidace je jedním z často používaných způsobů jak zvýšit přesnost testů na daných datech. Její princip spočívá v tom, že se data, která jsou k dispozici, sloučí a rozdělí se několikrát jinak na dvě části – trénovací a testovací. Vzhledem k tomu, že se pak test provádí pokaždé na jiných datech, lze považovat naměřené hodnoty za nezávislé. Počet dělení je určen především výpočetní kapacitou – proto jsem zvolil crossvalidaci pouze pětinásobnou. Úspěšnost jednotlivých verzí uváděná v tabulkách je pak aritmetickým průměrem úspěšností na těchto pěti sadách. Automaty vytvořené nad jednotlivými částmi jsem označil c0...c4.

5.3. Měření významnosti rozdílů – t-test

Pro určení toho, zda jsou naměřené rozdíly statisticky významné anebo jsou jen dílem náhody, byl použit takzvaný t-test[1]. Jeho výhoda je v tom, že jej lze aplikovat na porovnání dvou sad měření, kde se průměry těchto dvou měření liší jen o velmi málo oproti rozptylu každé sady. Lze však využít faktu, že vzorky ze dvou měřených sad lze spárovat, protože měření jsou prováděna vždy na stejných blocích dat.

[1] Detailní popis t-testu uvádí [3].

Použití t-testu ukážeme na porovnání verze *blažej2* a *kostěj*. Výsledky při pětinásobné crossvalidaci shrnuje následující tabulka. Pro každou část je uvedena nejvyšší dosažená úspěšnost.

	c0	**c1**	**c2**	**c3**	**c4**	**průměr**
blažej2 (Xi)	94,638	94,371	94,622	94,414	94,555	94,520
kostěj (Yi)	94,683	94,410	94,695	94,430	94,599	94,563

Tabulka 5. Naměřené výsledky pro verze blažej2 a kostěj při pětinásobné crossvalidaci

Tabulka ukazuje hodnoty naměřené pro jednotlivé části c0 až c4. Porovnáním průměrů těchto hodnot se zdá, že *kostěj* je nepatrně lepší. Chceme však vědět, zda se jedná o statisticky významný rozdíl (případně s jakou pravděpodobností lze říci, že to není dílem náhody).

Předně spočítáme rozdíly úspěšností na každém dílu zvlášť podle vzorce $D_i = X_i - Y_i$.

	c0	**c1**	**c2**	**c3**	**c4**
$\boldsymbol{D_i}$	0,045	0,039	0,073	0,016	0,044

Graf 4. Rozdíly naměřených hodnot

Nyní spočítáme průměr rozdílů, počet vzorů je $N=5$.

$$M_D = \frac{\sum_i D_i}{N} = 0{,}0434$$

Dále spočteme součet druhých mocnin odchylek:

$$SS_D = \sum D_i^2 - \frac{(\sum D_i)^2}{N} = 0{,}0016492$$

Standardní odchylka vzorkovací distribuce M_D se spočítá následovně:

$$\sigma_{M_D} = \sqrt{\frac{SS_D}{N(N-1)}} = 0{,}0090807$$

Nyní spočítáme výslednou hodnotu t:

$$t = \frac{M_D}{\sigma_{M_D}} = 4{,}7793415$$

Tuto hodnotu nyní porovnáme s tabulkou, která udává minimální hodnotu *t* pro statistickou významnost dané úrovně.

úroveň významnosti	0,05	0,02	0,01	0,001
t	2,78	3,75	4,60	8,61

Tabulka 6. Tabulka hodnot t pro obousměrný t-test na 5 měřeních (stupeň volnosti df=4)

Z porovnání s tabulkou plyne, že dvě porovnávané verze se od sebe opravdu liší, a to s významností 0,01. To lze přeložit tak, že pravděpodobnost, že se úspěšnost těchto dvou verzí od sebe liší pouhou náhodou, není větší než 0,01.

6. Data

Data pro trénování a testování musí být ve formátu SGML a odpovídat definici CSTS[1]. Pro účely tohoto projektu musí data obsahovat český text, kde jsou každému slovu přiřazeny možné značky, možná lemmata a je uvedena i správná (anotátorem vybraná) značka a lemma.

Data jsou rozdělena na tři části:

- trénovací data – největší blok, na kterém se provádí trénování
- testovací data – menší blok, na kterém se provádí testování a vyhodnocení chyb
- evaluační data – druhý menší blok, který se použije jen jednou na závěr a na kterém se určuje výsledná úspěšnost algoritmu

Během vývoje nových trénovacích verzí se vychází z chyb, které nastaly na testovacích datech. Tím ovšem dochází k tomu, že se algoritmus postupně přizpůsobuje i datům testovacím. Proto je zapotřebí mít ještě třetí blok dat (evaluační), nikdy neviděných, na kterých se vyhodnotí výsledná úspěšnost.

6.1. Charakteristika dat

Data byla součástí zadání. Původně byla k dispozici data z PDT 1.0. Po obhájení projektu Morče a zahájení této diplomové práce byla již k dispozici data z PDT 2.0, která jsou trochu větší, obsahují méně chyb a jsou jinak rozdělená. Stručná charakteristika původních i nových dat je uvedena v následujících tabulkách.

[1] viz [6]

	trénovací data	**testovací data**	**evaluační data**
počet slov	1470711	129574	129574
maximální počet lemmat na slovo	29	26	26
maximální počet rysů na slovo	118	117	87
průměrný počet lemmat na slovo	1,44	1,43	1,43
průměrný počet rysů na slovo	3,78	3,81	3,80

Tabulka 7. Původní data z PDT 1.0

	trénovací data	**testovací data**	**evaluační data**
počet slov	1539235	201651	219771
maximální počet lemmat na slovo	17	17	17
maximální počet rysů na slovo	82	57	57
průměrný počet lemmat na slovo	1,47	1,47	1,48
průměrný počet rysů na slovo	3,74	3,74	3,71

Tabulka 8. Nová data z PDT 2.0

6.2. Crossvalidace

Metoda srovnávání verzí, kterou jsem zavedl pro nová data, je založena na crossvalidaci. Data tedy byla spojena v pořadí trénovací, testovací, celkově je jejich velikost 1 740 886 slov. Evaluační data byla ponechána nedotknutá, aby bylo možno provést závěrečný test nezávisle na trénování. Přesné rozdělení dat ukazuje následující tabulka.

název	**začátek**	**konec**	**velikost trénovacího bloku**	**velikost testovacího bloku**
c0[1]	1 539 236	1 740 886	1 539 235	201 651
c1	1	200 000	1 540 886	200 000
c2	200 001	400 000	1 540 886	200 000
c3	400 001	600 000	1 540 886	200 000
c4	600 001	800 000	1 540 886	200 000

[1] Toto rozdělení zcela odpovídá původnímu dělení na trénovací a testovací data.

6.3. Analýza dat

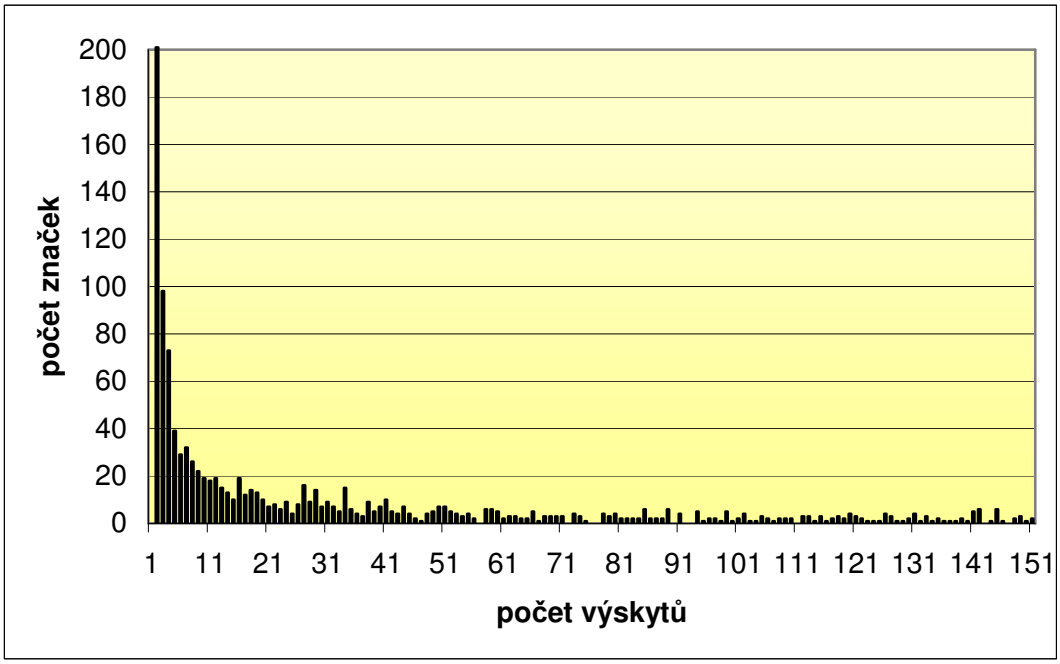
Na sloučených trénovacích a testovacích datech z PDT 2.0 jsem provedl jednoduchou frekvenční analýzu. Informace této analýzy mohou být cenné pro uvažování o chování algoritmu.

Celkový počet různých značek v datech je 1132. Z toho se 201 značek vyskytlo pouze jednou, 98 dvakrát a 73 třikrát.

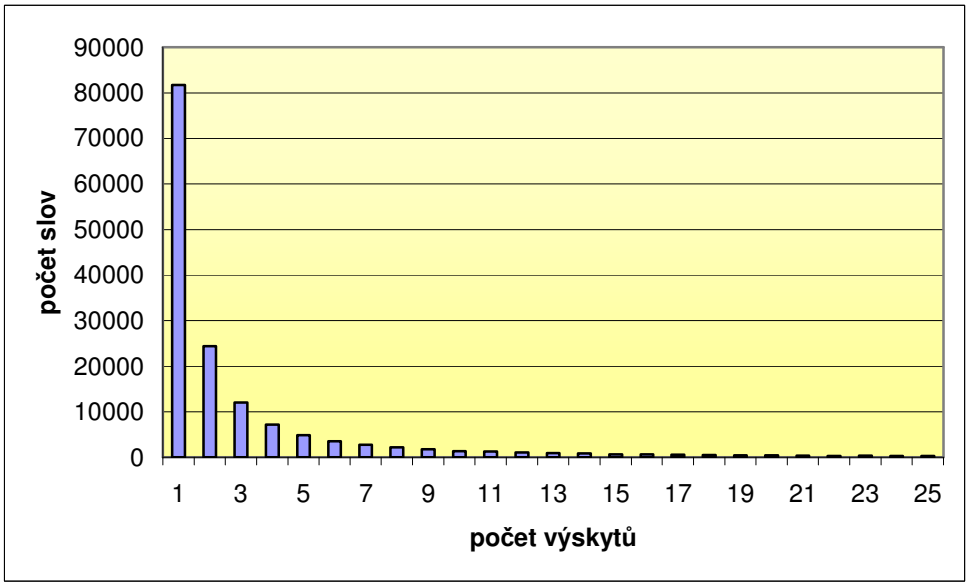
Celkový počet různých slov je 150674. Z toho se 81745 slov vyskytlo pouze jednou, 24361 dvakrát a 12075 třikrát.

Grafy 5 a 6 ukazují počty různých značek a slov v závislosti na jejich celkovém počtu výskytů v datech.

Tyto informace jsou důležité pro posouzení chování vlivu filtrování rysů. Víme-li například, že 201 značek se v textu vyskytuje pouze jednou, pak je jisté, že pro tyto značky nebude po odfiltrování rysů s méně jak třemi výskyty existovat žádný rys.



Graf 5. Počty značek podle jejich počtu výskytů v datech



Graf 6. Počty slov podle jejich počtu výskytů v datech

7. Automatický vývoj rysů

Práce experimentátora sestává zčásti z mechanického skládání rysů a jejich kombinování do verzí. Tento základní vývoj by mohl probíhat i automaticky. Navíc automatický vývoj verzí nemůže preferovat lingvistickou intuici stejně jako experimentátor, a tak může posloužit k ověření, že nebyla přehlédnuta žádná důležitá možnost. Automatické testování také může dobře posloužit pro ověření některých obecných vlastností algoritmu.

Pro účel automatického testování jsem upravil stávající kód a vytvořil pomocné programy, které budou dále popsány.

Vzhledem k tomu, že počet všech možností je neúnosně vysoký[1], bylo třeba jednak počet prozkoumávaných verzí zásadně omezit a jednak naprogramovat celou úlohu tak, aby mohla být řešena paralelně ve více procesech.

Obecně sestává schéma ze dvou typů procesů:

- *otrokář* (master): zajišťuje samotný vývoj – na základě dosud naměřených hodnot vytváří podle zadaných kritérií další verze
- *otrok* (slave): přebírá úkoly vytvořené otrokářem a předává mu naměřené výsledky

Automatický vývoj sestává ze dvou nezávislých částí:

- skládání elementárních rysů do složených
- kombinování rysů do verzí

Protože skládání a kombinování rysů zajišťuje pouze otrokář, byl otrok byl pro obě části shodný. Práce otroka spočívá v podstatě jen v tom, že spustí zadaný skript se správnými parametry.

U otrokářů bylo třeba zvolit způsob vytváření nových verzí a kritéria, na jejichž základě budou nové verze přijímány nebo odmítány.

[1] viz 7.1

7.1. Celkový počet možných verzí

Všechny rysy lze rozdělit na elementární části, ze kterých se skládají. Každý rys musí obsahovat nějakou předpověď aktuální značky[1]. Elementární rysy lze tedy libovolně kombinovat při dodržení této podmínky. V naprosté většině verzí rysy předpovídaly pouze celou značku, což představuje jeden povinný elementární rys |T.

Označme počet elementárních rysů *n*. Z celkového počtu elementárních rysů pak lze spočítat počet všech možných verzí.[2]

1.) vytvoření složených rysů

Odečteme elementární rys |T, který musí být obsažen ve všech složených rysech. Ostatní rysy popisují kontext, takže je lze libovolně kombinovat. Celkově je tedy počet složených rysů $s = 2^{(n-1)}$.

2.) kombinování složených rysů do verzí

Každá verze se skládá z *0..s* složených rysů. Počet verzí je tedy $v = 2^s = 2^{2^{(n-1)}}$. To již při velmi malém počtu elementárních rysů dává obrovský počet verzí, který není možné otestovat. Pro vývoj verzí tedy bylo potřeba stanovit omezení, která počet testovaných verzí radikálně sníží. Při definici těchto omezení jsem se snažil formalizovat minimální intuitivní požadavky, které bych kladl na úspěšné verze.

7.2. Kódování rysů a verzí

Pro skládání a kombinace rysů jsem použil následující kódování:

Čísla *1..n* označují elementární rysy. Jsou-li *x* a *y* dva rysy, pak *x:y* označuje rys vzniklý složením *x* a *y*. Rys *x:y* tedy obsahuje všechny elementární rysy z *x*, všechny elementární rysy z *y*, každý rys však nejvýše jednou.

Základní verze obsahují pouze jeden rys. Jsou-li *x* a *y* dvě verze, pak *x, y* označuje složení těchto verzí. Verze *x, y* obsahuje všechny rysy z *x*, všechny rysy z *y*, každý rys však nejvýše jednou.

[1] Předpovědí aktuální značky je buď značka samotná, nebo její část (pád, slovní poddruh a podobně).
[2] Skutečný počet může být nakonec ještě větší, protože některé elementární rysy lze dále parametrizovat – například omezením, jak daleko se má vyhledávat nejbližší sloveso, přes jaké slovní druhy již nevyhledávat a podobně.

Symbolem absolutní hodnoty budu označovat úspěšnost dané verze. Pokud použiji symbol absolutní hodnoty na rys, označuji tím úspěšnost verze sestávající pouze z tohoto rysu.

Symbol ε označuje minimální mez, o kterou se verze musí zlepšit, aby byly přijaty. Tato hodnota se zadává parametrem při spuštění otrokáře.

Příklad

Pokud budou použity elementární rysy 1 – aktuální značka, 2 – předchozí značka, 3 – značka tři pozice zpět, pak kód verze odpovídající trigramovému modelu s vyhlazováním (v kapitole 8 označována jako verze *jennefer*) bude zapsán takto: „1, 1:2, 1:2:3“.

7.3. Skládání rysů

Při skládání rysů je ručně zadána sada rysů. Každá základní verze sestává z jediného rysu. Navíc jsou generovány verze pomocné skládající se ze dvou rysů, které slouží pouze pro ověřování přijatelnosti základních verzí.

Přijatelnost rysu *x* jsem definoval následovně:

R1) žádný rys *y*, jehož elementární rysy jsou podmnožinou rysu *x,* nesmí být odmítnutý

R2) pro žádný rys *y*, jehož elementární rysy jsou podmnožinou rysu *x,* nesmí platit: $|y| \geq |x : y, y| + \varepsilon$

Záměrně jsou pravidla formulována negativně, protože pokud se nenalezne žádná verze, která by způsobila nepřijetí verze *x*, je *x* přijata. Její přijetí však může být pouze dočasné do doby, než se objeví verze, která jej zruší.

Pravidlo R1 způsobí, že po odmítnutí nějakého rysu se nebude v dané větvi vývoje pokračovat. Pravidlo R2 požaduje, aby složený rys přinášel ke všem svým předchůdcům zlepšení alespoň o zadanou mez.

Vzhledem k tomu, že celý vývoj může běžet paralelně ve více procesech, a tudíž není zaručeno v jakém pořadí budou verze otestovány, je postup ověřování komplikovanější. Při vytváření nového rysu se postupuje následovně:

1. ověří se, že rys dosud nebyl vytvořen
2. ověří se, že neexistuje podmnožina rysu, která již byla odmítnuta

3. rys x je vytvořen a uložen do seznamu verzí připravených k otestování, zároveň je pro každou podmnožinu tohoto rysu y vytvořena pomocná verze x:y,x

Při otestování základní verze (s jedním rysem) se postupuje následovně:

1. pokud byl rys zatím odmítnut, odmítnut zůstane (to se může stát proto, že některé rysy mohou vzniknout více způsoby a mohlo zatím dojít k odmítnutí některého z jeho předchůdců)
2. pro žádný rys *y*, jehož elementární rysy jsou podmnožinou *x* a který již byl otestován, nesmí platit: $|y| \geq |x : y, y| + \varepsilon$ (aplikace pravidla R2 směrem k předchůdcům *x*)
3. pro žádný rys *z*, kde elementární rysy rysu *x* jsou podmnožinou *z*, nesmí platit: $|x| \geq |z : x, x| + \varepsilon$ (aplikace pravidla R2 směrem k následníkům *x*)
4. jestliže je rys přijat, je složen se všemi dosud přijatými rysy

Při otestování pomocné verze *x,y* (se dvěma rysy) se postupuje následovně:

1. pokud byl již otestován rys *y* a platí podmínka: $|y| \geq |x : y, y| + \varepsilon$, je rys *x* odmítnut
2. pokud byl již otestován rys *x* a platí podmínka: $|x| \geq |x : y, x| + \varepsilon$, je rys *y* odmítnut

Tímto postupem je zaručeno, že po skončení všech procesů budou verze označené jako přijaté splňovat podmínky R1 a R2.

7.4. Kombinování verzí

Ručně je zadána sada základních verzí (každá z nich sestává z jediného rysu). Jakmile jsou tyto verze otestovány, otrokář začne generovat nové verze, které vzniknou jejich kombinací, a verze, které slouží pro ověření přijatelnosti.

Přijatelnost verze *x* jsem definoval následovně:

V1) žádná verze *y*, jejíž rysy jsou podmnožinou verze *x* (*x* vznikne z *y* přidáním rysů), nesmí být odmítnutá

V2) pro žádnou verzi *y*, jejíž rysy jsou podmnožinou verze *x* (*x* vznikne z *y* přidáním rysů), nesmí platit: $|y| \geq |x| + \varepsilon$

Opět jsou pravidla formulována negativně ze stejných důvodů jako při skládání rysů (přijetí verze může být pouze dočasné).

Pravidlo V1 způsobí tedy to, že se nepokračuje ve vývojové větvi, která navazuje na alespoň jednu odmítnutou verzi. Pravidlo V2 formuluje požadavek, aby se každým přidáním rysů zvedla úspěšnost alespoň o zadanou mez.

Při vytváření nové verze se postupuje následovně:

1. ověří se, že verze dosud nebyla vytvořena
2. ověří se, že neexistuje podmnožina, která již byla odmítnuta
3. verze je vytvořena a uložena do seznamu verzí připravených k otestování

Při otestování verze se postupuje následovně:

1. pokud byla verze zatím odmítnuta, odmítnuta zůstane (to se může stát proto, že některé verze mohou vzniknout více způsoby a mohlo zatím dojít k odmítnutí některé z jejích předchůdců)
2. pro žádnou verzi *y*, jejíž rysy jsou podmnožinou verze *x* (*x* vznikne z *y* přidáním rysů) a která již byla otestována, nesmí platit: $|y| \geq |x, y| + \varepsilon$
3. pro žádnou verzi *z*, kde rysy verze *x* jsou podmnožinou rysů verze *z*, nesmí platit: $|x| \geq |x, z| + \varepsilon$
4. Pokud je verze přijata, je zkombinována s každou dosud přijatou verzí.

Tímto postupem je zaručeno, že po skončení všech procesů budou verze označené jako přijaté splňovat podmínky V1 a V2.

7.5. Implementace

Pro automatické testování byl vytvořen zvláštní generátor (označený jako verze *emil*), ve kterém jsou definovány elementární rysy, ale typy skutečně použitých rysů definuje inicializační soubor – tímto generátorem lze tedy vytvořit jakoukoliv verzi sestávající z těchto elementárních rysů. Dále byly upraveny i některé další části kódu, aby načítaly inicializační soubor, ovšem při zachování zpětné kompatibility s ostatními generátory.

Komunikace mezi procesy probíhá pomocí souborů. Díky tomu je možné celý vývoj obnovit od určitého bodu (buď při nějaké ruční změně v konfiguraci, anebo po zkolabování některého procesu). Počet otroků není omezen, lze je spouštět a zastavovat libovolně podle aktuálně dostupných zdrojů.

7.5.1. Synchronizace

Synchronizace práce se sdílenými soubory se provádí standardním způsobem pomocí jejich uzamykání.

Uzamykání souboru probíhá podle následujícího algoritmu:

1. vytvoř nový link na soubor (pokud to nelze, počkej 30s a pokus opakuj, maximálně však 30krát)
2. zjisti počet linků na soubor
3. je-li počet větší než 2 (původní a tímto procesem vytvořený), zruš svůj link, počkej 30s a pokračuj bodem 1
4. je-li počet linků roven 2, proveď se souborem požadované operace
5. zruš svůj link
6. proveď akci
7. pokračuj bodem 1

7.5.2. Používané soubory

Všechny soubory jsou textové, řádkově orientované. Řádky s komentářem ve všech souborech začínají znakem #.

tasks.m – hlavní soubor, kde se ukládají úkoly pro otroky. Každý úkol představuje jednu verzi k otestování. Zároveň se u každé verze uchovává informace o jejím stavu. Soubor se čte po řádcích, kde každý neprázdný řádek představuje jednu verzi nebo komentář.
Formát řádku je následující:
{stav} {skript a parametry} {název verze} {inicializační soubor}{ID procesu}
Jednotlivé položky jsou odděleny tabelátorem.
Možné stavy verze jsou:

F – vytvořena a volná ke zpracování

P – zpracovávána určitým otrokem

D – otestování hotovo, ke zpracování otrokářem

R – verze odmítnuta

A – verze přijata

U – pomocná verze, nepřijímá se ani neodmítá

slaves.m – seznam běžících otroků. Každý neprázdný řádek je buď záznam jednoho otroka nebo komentář.
Záznam otroka má následující formát:

{stav} {název stroje}_{ID procesu}

Možné stavy otroka jsou:

R – běžící

S – žádost o zastavení běhu

Žádost o zastavení otroka se provádí ručně. Otrok pak ukončí svůj běh po dokončení právě rozdělaného úkolu.

inicializační soubory – jejich názvy sestávají z tohoto schématu:
{výchozí název}-{číslo}.init
Výchozí název je určen parametrem při spuštění otrokáře.
Inicializační soubor se načítá po řádcích, kde na každém neprázdném řádku je buď typ rysu, nebo komentář. Komentáře začínají znakem #. Typy rysů jsou zapsány ve tvaru x:y:z:..., kde x, y, z... jsou čísla elementárních rysů.

výsledky – soubory s názvy se ukládají do adresáře *res* a jejich název je:
{název verze}.eval
Pro každou otestovanou iteraci obsahují dva řádky: první řádek s informacemi o generátoru, verzi, automatu, datech a iteraci, druhý řádek s výslednou úspešností.

7.5.3. Činnost otroka

1. Při spuštění se otrok zapíše do seznamu otroků v souboru slaves.m. Zapíše svůj stav (běžící – R), název stroje a číslo procesu.
2. Otrok zkontroluje, zda na jeho řádku v slaves.m nebyl změněn stav na S (žádost o zastavení). Pokud ano, skončí svůj běh.
3. Otrok zamkne tasks.m a vyhledá první volnou verzi ke zpracování (stav F). Označí ji jako zpracovávanou (stav P), připíše na její řádek své identifikační údaje (název stroje a číslo procesu), tasks.m odemkne a verzi otestuje. Po dokončení výpočtu opět zamkne tasks.m, a příslušnou verzi označí jako zpracovanou (stav D).
4. Otrok může běžet ve dvou módech – smrtelný a nesmrtelný. Jestliže otrok nenajde žádnou verzi ke zpracování, pak ve smrtelném módu okamžitě ukončí svůj běh. V nesmrtelném módu čeká, dokud není tasks.m změněn, a pak se opět pokusí o nalezení nezpracované úlohy (bod 2).
5. Pokud otrok ukončí svoji činnost, ať už proto, že ve smrtelném módu nenašel volnou verzi, nebo protože došlo k nějaké kritické chybě, pokusí se škrtnout ze seznamu běžících otroků v souboru slaves.m.

7.5.4. Činnost otrokáře

1. Otrokář zamkne soubor tasks.m a načte všechny verze, které jsou v něm uloženy.

Načtení verze sestává z následujících kroků:

- z tasks.m načtení názvu verze, inicializačního souboru a stavu verze
- načtení inicializačního souboru verze
- pokud má verze stav R, D nebo A, načtení úspěšnosti verze (pokud je stav R, může se stát, že žádný výsledek neexistuje, v tomto stavu to není chyba)

2. Pro všechny verze, které jsou hotovy, ale nezpracovány otrokářem (stav D), otestuje jejich přijatelnost a případně vygeneruje nové verze, které zapíše na konec souboru. Soubor tasks.m je pak odemknut.
3. Otrokář čeká na změnu souboru tasks.m, pak opět pokračuje bodem 1.

7.5.5. Parametry spouštění

Zdrojový kód programů se nachází v projektovém adresáři v podadresáři *src*, zkompilované binární soubory v podadresáři *bin*, skripty v podadresáři *scripts*. Pro spuštění experimentu je vhodné vytvořit samostatný adresář, nakopírovat do něj všechny potřebné binární soubory a potřebné skripty.

Spuštění otrokáře

Otrokář existuje ve dvou variantách:

- skládání verzí: m_master
- kombinování rysů: m_master2

Obě varianty mají stejné parametry.

m_master[2] [parametry]

-p cesta	cesta do adresáře s experimentem
-g generátor	jméno generátoru
-b cesta	cesta ke spustitelným souborům
-n jméno	jméno inicializačních souborů
-e epsilon	prahová hodnota pro přijímání verzí

Spuštění otroka

m_slave [parametry]

-p cesta cesta do adresáře s experimentem

-i nesmrtelnost (pokud nejsou žádné volné úlohy, čeká se dál) (volitelné)

K samotnému otestování se používá skript *do_atrain*, jehož parametry jsou následující:
do_atrain {cesta k binárním souborům} {název generátoru} {název automatu} {inicializační soubor pro rysy}

Tento skript může být potřeba před samotným spuštěným upravit. Na jeho začátku je potřeba případně změnit název trénovacích dat, testovacích dat, počet iterací a počet výskytů pro filtrování rysů.

7.6. Výsledky automatického vývoje

Všechny pokusy byly prováděny na rysech popisujících značky a slovní formy až tři pozice zpět. Elementární rysy popisuje následující tabulka.

číslo	signatura rysu	význam rysu
1	|T	aktuální značka
2	AT	předchozí značka
3	BT	značka dvě pozice zpět
4	|W	aktuální slovo
5	AW	předchozí slovo
6	BW	slovo dvě pozice zpět

Tabulka 9. Přehled elementárních rysů pro automatický vývoj verzí

7.6.1. Skládání rysů

Výchozí verze pro skládání rysů jsou uvedeny v následující tabulce.

kód rysu	signatura rysu
1	|T
1:2	AT|T
1:3	BT|T
1:4	|W|T
1:5	AW|T
1:6	BW|T

Tabulka 10. Výchozí rysy pro automatické skládání

Aktuální značka byla použita samostatně, každý další elementární rys pak v kombinaci s aktuální značkou. Tím je zaručeno, že i při kombinování bude každý rys obsahovat informaci o aktuální značce, a tudíž bude smysluplný.

Experiment byl proveden na trénovacích a testovacích datech z PDT 2.0. Hodnota epsilon byla nastavena na 0,5, což znamená alespoň půl procenta zlepšení pro přijetí verze. Rysy s méně než třemi výskyty byly odfiltrovány. Trénování se provádělo v 5 iteracích.

Celý experiment běžel na 15 procesorech (14 otroků a 1 otrokář) zhruba 24 hodin. Všechny přijaté rysy shrnuje následující tabulka:

kód rysu	**signatura rysu**	**úspěšnost**
1:2	AT|T	91,099
1:2:3	BTAT|T	88,864
1:2:5	AWAT|T	82,540
1:2:4	AT|W|T	80,453
1:3	BT|T	79,707
1	|T	77,353
1:2:3:5	BTAWAT|T	74,988
1:3:6	BWBT|T	73,297
1:3:4	BT|W|T	72,238
1:3:5	BTAW|T	69,159
1:5	AW|T	67,621
1:4	|W|T	67,451
1:6	BW|T	65,310
1:2:4:5	AWAT|W|T	62,662
1:3:4:6	BWBT|W|T	60,481
1:3:4:5	BTAW|W|T	59,234
1:3:5:6	BWBTAW|T	59,179
1:5:6	BWAW|T	57,756
1:4:5	AW|W|T	57,204
1:4:6	BW|W|T	57,079

Tabulka 11. Přehled přijatých rysů při automatickém skládání

Z celkového počtu 32 možných rysů jich bylo přijato 21 (z toho 6 bylo součástí zadání). Všechny přijaté rysy se skládají maximálně ze čtyř elementárních rysů. Ze všech možných čtveřic (10) jich byla přijata pouze polovina. To podporuje pozorování, že pro algoritmus jsou vhodnější rysy spíše kratší (méně komplikované).

7.6.2. Kombinování verzí

Výchozími verzemi pro kombinování byly všechny smysluplné rysy (obsahující aktuální značku) složené z elementárních rysů uvedených v Tabulce 9, nejvýše však trojice elementárních rysů. Všechny výchozí verze shrnuje následující tabulka:

kód rysu	signatura rysu
1	|T
1:2	AT|T
1:2:3	BTAT|T
1:2:4	AT|W|T
1:2:5	ATAW|T
1:2:6	BWAT|T
1:3	BT|T
1:3:4	BT|W|T
1:3:5	BTAW|T
1:3:6	BTBW|T
1:4	|W|T
1:4:5	AW|W|T
1:4:6	BW|W|T
1:5	AW|T
1:5:6	BWAW|T

Tabulka 12. Přehled výchozích verzí pro automatické kombinování

Experiment byl opět proveden na nových datech, hodnota epsilon byla nastavena na 0,5. Rysy s méně než třemi výskyty byly odfiltrovány. Protože nyní již mohly vznikat verze poměrně komplikované, zvýšil jsem počet iterací z 5 na 8.

Celý experiment běžel na 15 procesorech (14 otroků a 1 otrokář) zhruba tři týdny. Během této doby bylo otestováno 1039 verzí, z nichž 650 bylo přijato. Celkový počet automaticky vygenerovaných verzí se vyšplhal na 4086, většina z nich však byla odmítnuta dříve, než mohla být otestována (byl odmítnut některý z jejích předchůdců).

Následující tabulka shrnuje nejúspěšnější verze (úspěšnost přes 93%):

označení verze	**signatury rysů**	**úspěšnost**
1:2, 1:3, 1:4	AT|T, BT|T, |W|T	93,784
1:2, 1:2:3, 1:4	AT|T, BTAT|T, |W|T	93,600
1:2, 1:3:5, 1:4	AT|T, BTAW, |W|T	93,549
1:2, 1:2:5, 1:3	AT|T, AWAT|T, BT|T	93,516
1:2:3, 1:2:5, 1:3, 1:4	BTAT|T, AWAT|T, BT|T, |W|T	93,458
1:2, 1:2:4, 1:3	AT|T, AT|W|T, BT|T	93,439
1:2, 1:2:3, 1:2:5	AT|T, BTAT|T, AWAT|T	93,161
1:2, 1:3, 1:4:5	AT|T, BT|T, AW|W|T	93,128
1:2, 1:2:3, 1:2:4	AT|T, BTAT|T, AT|W|T	93,111
1:2, 1:2:5, 1:3:5	AT|T, AWAT|T, BTAW|T	93,043

Tabulka 13. Přehled nejúspěšnějších verzí při automatickém kombinování

Jak je vidět, jako nejdůležitější se ukázal rys AT|T (1:2, značkový bigram) a předchozí značka i v dalších kombinacích. Poněkud překvapivě se však v žádné z těchto verzí neobjevil značkový unigram (|T). Poměrně důležitou roli hraje také aktuální slovní forma, ať už samostatně, nebo v dalších kombinacích. Naproti tomu slovní forma dvě pozice zpět se ukázala jako nevýznamná, v žádné úspěšné verzi nefiguruje ani samostatně, ani v kombinaci s dalšími rysy.

Mírně převažující je podíl jednodušších rysů (dvojice) nad složitějšími (trojice), zejména v horní části tabulky. Ze složitějších kombinací rysů nevystupuje žádná kombinace výrazněji do popředí jako častější. Zdá se tedy, že není tak důležité, jak přesně jsou rysy nakombinovány, ale především zda se ve verzi nějak vyskytují. To ve spojení s požadavkem jednoduchosti rysů ukazuje směr přidávání rysů nekombinovaných. Tento závěr je v souladu s metodou při ručním vytváření verzí.

Při ručním vytváření verzí byla nad uvedenou množinou elementárních rysů dosažena úspěšnost 94,118% (verze *ijáček*). To, že se automatický vývoj nedostal k podobné hodnotě, je zřejmě způsobeno příliš vysokou prahovou hodnotou ε. Pokud by však byla snížena, byl by celý proces při stávajícím nastavení neúměrně časově náročný.

Vzhledem k tomu, že automatický vývoj verzí nepřinesl překvapivé výsledky a v podstatě jen podpořil vývojové větve, kterými se ubíral ruční vývoj, byl v této fázi zastaven. Dalším důvodem byla zmíněná časová náročnost.

7.7. Problémy automatického vývoje verzí

Základním problémem zejména při kombinování verzí je velká časová náročnost. Přestože zformulovaná pravidla pro odmítání počet celkově testovaných verzí výrazně snižuje, trvá testování velmi dlouho (při kombinování řádově týdny na 14 procesorech). Časovou náročnost lze snížit několika způsoby:

- použitím menší části dat: tím se ovšem vystavujeme nebezpečí, že výsledky budou více náhodně kolísat a naměřené hodnoty nebudou odpovídat výsledkům na celých datech
- snížením počtu iterací: tím se však zvýhodní jednoduché verze, u nichž zpravidla nastává optimální úspěšnost v nižších iteracích
- zvýšením prahové hodnoty ε: tím dojde i k výraznějšímu odmítání verzí a celých vývojových verzí. Použitá hodnota v popsaných experimentech (0,5%) je kompromisem mezi časovou únosností a rozumnými výsledky. Při nenulové prahové hodnotě však není zaručeno nalezení nejlepší možné verze.

Pro další automatický vývoj by bylo třeba buď více strojového času, nebo další a přísnější pravidla pro odmítání verzí. Pro prahovou hodnotu by bylo zřejmě vhodnější, aby nebyla konstantní, ale aby se s rostoucí úspěšností snižovala. Zlepšení 0,5% u těch nejjednodušších verzí (úspěšnost pod 90%) není nijak výrazné, zatímco u úspěšných verzí by postačilo zlepšení výrazně menší (řádově desetina procenta).

Přestože automatický vývoj tak, jak byl proveden, nepřinesl zlepšení úspěšnosti, splnil svůj účel v ověření obecných vlastností algoritmu.

8. Ruční vývoj verzí

V této kapitole bude popsána práce experimentátora při ručním vytváření nových verzí. Každá část obsahuje jednu vývojovou větev s naměřenými hodnotami. Jsou zde popsány jak větve perspektivní, tak i větve slepé, které ke zlepšení nevedly. Na závěr kapitoly je pak uvedena hlavní vývojová větev od počáteční verze až k verzi nejlepší.

Značný rozsah provedených testů vyplynul z obecných vlastností algoritmu, především z toho, že použitelný počet typů rysů je omezen – při přílišném počtu použitých typů rysů se úspěšnost snižuje. Nelze tedy postupovat prostým zvyšováním počtu rysů, ale je zapotřebí provádět rozumný výběr z možných kombinací rysů.

8.1. Předpoklady a pravidla

Při rozvaze nad uspořádáním verzí a skládání rysů jsem byl nucen přijmout následující předpoklad:

Alespoň jedna vývojová cesta k nejlepší možné verzi nevede přes lokální minima.

Jinak řečeno: existuje způsob, jak kombinovat rysy a postupným zlepšováním úspěšnosti se dobrat až k nejlepší možné verzi[1]. Podle tohoto předpokladu pak mohou být vývojové cesty, které vedly ke zhoršení, bez obav opuštěny. Při vytváření nových verzí přidáváním dalších rysů se vychází od doposud nejlepších verzí (případně jejich předchůdců), ale nikoliv od verzí, které vedly ke zhoršení výsledku.

Přestože pravděpodobně nelze toto tvrzení teoreticky dokázat, bylo nutné jej přijmout – odpovídá intuitivním očekáváním a vnáší základní řád do práce experimentátora.

Dále jsem zavedl několik pravidel pro vytváření rysů a verzí, která vycházejí z vypozorovaných vlastností algoritmu a také omezují počet testovaných verzí. Tato pravidla nejsou nijak závazná, mají spíše obecně charakterizovat způsob mé práce a jejich charakter je doporučující.

- **skládat rysy co nejméně, maximálně do délky čtyř elementárních rysů[2]**

Delší rysy jsou specializovanější a přesněji popisují daný kontext, ale jejich celkové množství je příliš veliké. Navíc se většina z nich vyskytuje pouze jednou v daném speciálním kontextu v trénovacích datech, takže pro značkování testovacích dat se nehodí. Rysy by měly zobecňovat to nejpodstatnější z kontextu.

[1] Je míněna nejlepší možná verze v rámci zvolených podmínek – tedy v rámci použitých elementárních rysů.
[2] většinou však jen dvou

- **skládat rysy lingvisticky smysluplně**

Toto pravidlo nedoporučuje zkoušet skládání všech rysů se všemi – je to neúnosně náročné. Proto se dává při vývoji přednost rysům, které nějak odpovídají lingvistické intuici, tedy takovým, jejichž význam lze rozumně zdůvodnit. Pokud tedy ke dochází ke skládání rysů popisujících kontext, volí se většinou rysy typově podobné – skládají se například slovní formy či značky na předchozích pozicích.

- **přidávat nové rysy jen k významným verzím – alespoň jedné jednoduché, alespoň jedné komplikovanější**

Přidávat nové rysy ke všem dosud úspěšným verzím by bylo příliš zdlouhavé. Proto se nové rysy zpravidla otestují na nějaké základní jednoduché úspěšné verzi, tím se ověří jejich smysluplnost. Pokud neuspějí, dál se už nepoužijí. Pokud se osvědčí, zkusí se přidat k dosud nejlepší verzi.

8.2. Aplikace t-testu

Původní metoda vývoje verzí se zakládala pouze na zlepšení úspěšnosti na jedněch testovacích datech. Bylo potřeba ji nahradit novou metodou, založenou na pětinásobné crossvalidaci a t-testu. **Nová verze je považována za úspěšnou pouze tehdy, pokud přináší statisticky významné zlepšení oproti svému bezprostřednímu předchůdci.**

Tím se pětkrát zvýšila časová náročnost pro otestování jedné verze. Základní verze byly přetestovány novou metodou, aby se ověřilo, zda není výsledná verze *lucifer4* nejlepší pouze na původních datech a zda není možné z již otestovaných verzí dospět k jiným výsledným verzím (což se potvrdilo). Navíc jsem provedl novou řadu experimentů s dosud nepoužitými rysy.

8.3. Trénovací verze

Popis jedné vývojové větve sestává z komentáře, dvou tabulek a grafu.

Tabulka s přehledem verzí obsahuje pro každou verzi její název, kompletní výčet použitých typů rysů, jejich počet na jeden přechod mezi stavy HMM (tedy počet rysů vygenerovaných pro jednu možnou značku) a výslednou úspěšnost. Výsledná úspěšnost představuje průměr z pětinásobné crossvalidace na datech z PDT 2.0.

Srovnávací tabulka zobrazuje pro vybrané dvojice verzí (každá verze se srovnává se svými bezprostředními následníky) hodnotu koeficientu *t* spočítanou pro t-test a na jejím základě určenou úroveň významnosti. Je-li hodnota *t* záporná, znamená to, že verze 2 byla

horší než verze 1. V tabulce jsou pro srovnání uváděny i verze, které daný typ rysů sice nepoužívají, ale tyto rysy k nim byly přidány.

Graf uspořádání verzí zachycuje informace z obou tabulek. Zeleně jsou zobrazeny verze, které nejsou následníkem žádné verze v daném grafu – k těmto výchozím verzím byl přidáván daný typ rysů. Verze jsou v grafu uspořádány tak, že verze odvozené jsou vždy zobrazeny pod úrovní svých předchůdců. Každá verze je čarami spojena pouze se svými bezprostředními předchůdci. Význam šipek a jejich barev ukazuje následující přehled:

↓	zlepšení úroveň významnosti 0,001	↑	zhoršení úroveň významnosti 0,001
↓	zlepšení úroveň významnosti 0,01	↑	zhoršení úroveň významnosti 0,01
↓	zlepšení úroveň významnosti 0,02	↑	zhoršení úroveň významnosti 0,02
↓	zlepšení úroveň významnosti 0,05	↑	zhoršení úroveň významnosti 0,05
↓	zlepšení statisticky nevýznamné	↑	zhoršení statisticky nevýznamné
|	stejná úspěšnost statisticky nevýznamné	|	spojení ranné verze s nějakou pozdější, zlepšení je výrazné, ale pro danou část není podstatné

Tabulka 14. Význam barev v grafech uspořádání verzí

8.3.1. Značky a slovní formy na posledních třech pozicích

Základní verze, ze které se vycházelo, představuje tzv. trigramový model s vyhlazováním. Skládá se ze značkového trigramu, bigramu a unigramu. Její úspěšnost (92,63%) je srovnatelná s obdobnými metodami používajícími tento model. V dalších krocích jsem přidával informaci o slovních formách na posledních třech pozicích. U slovních forem se ukázalo, že komplikovanější rysy selhávají. Neosvědčil se slovní trigram (*hermiona*), ani kombinace slova dvě pozice zpět se slovem aktuálním (*minerva*, *ijáček*). Naproti tomu se ukázalo jako užitečné používat i rysy „nesouvislé“, tedy popisující nějakou vlastnost dvě pozice zpět a aktuální pozici, ale zcela přeskakující pozici předchozí. To dokládá jak použití rysu BW (slovní forma) ve verzi *šemík* a *lavender*, tak rysu BT (značka), ve verzi *kryštůfek*. Typickým příkladem, kde je tento rys vhodný, jsou jmenné fráze, protože pád aktuálního slova může záviset na předložce, která je však od jména oddělena nesklonným adverbiem („...viděli ho **na** příliš **vzdáleném místě**...“).

Dále jsem se pokusil zkombinovat informaci o slovní formě se značkou – jak na předchozí pozici (*klokánek*), tak dvě pozice zpět (*klokanice*). Tyto kombinované rysy však vedly dokonce ke zhoršení výsledku.

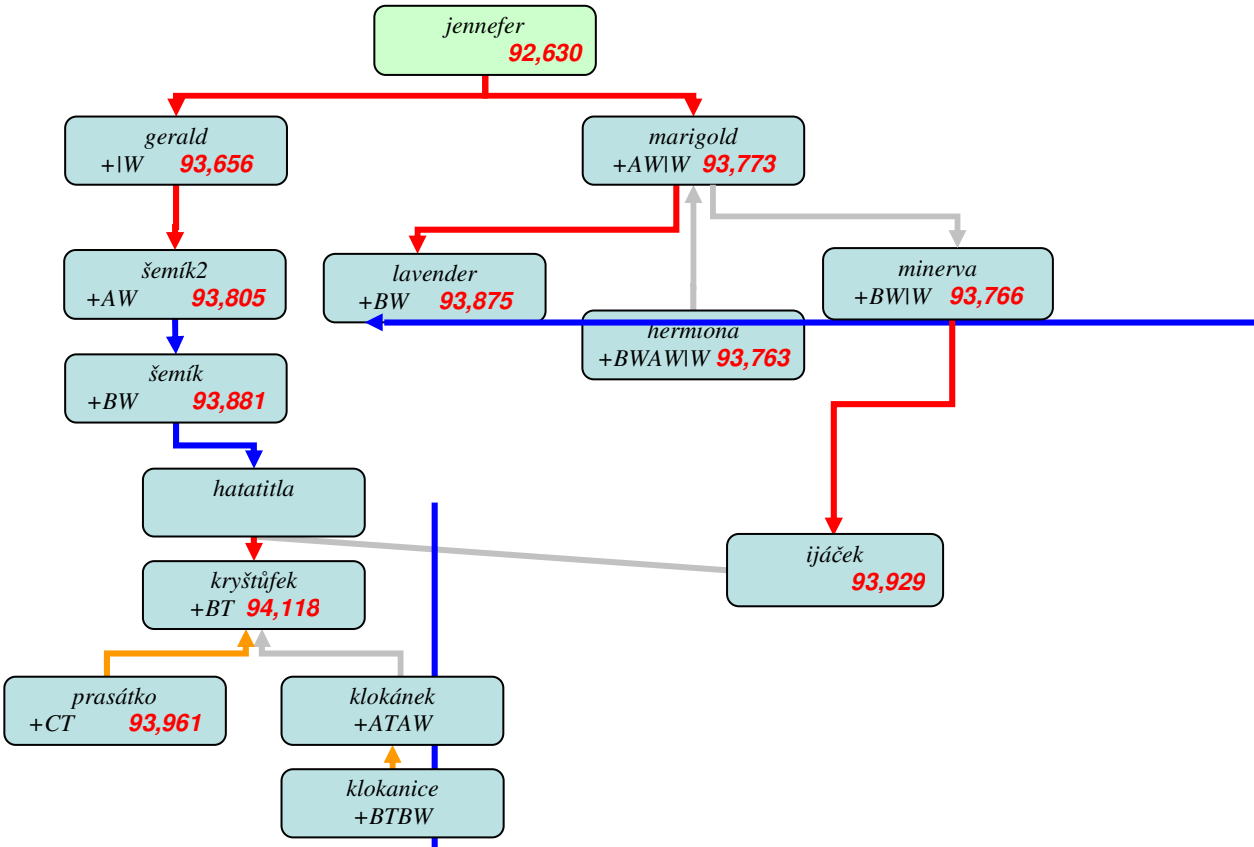
Také pokus přidat samostatně i značku tři pozice zpět se ukázal jako neúspěšný. Zřejmě přeskočení dvou slov již vede ke ztrátě podstatných informací a souvislost aktuální značky se slovem tři pozice zpět není příliš významná. Z této vývojové větve se tedy nejvíce osvědčila verze *kryštůfek*, která byla použita jako výchozí v dalších větvích.

verze	**rysy**	**počet**	**úspěšnost**
jennefer	|T, AT, ATBT	3	92,630
gerald	|T, AT, ATBT, |W	4	93,656
marigold	|T, AT, ATBT, AW|W	4	93,773
lavender	|T, AT, ATBT, AW|W, BW	5	93,875
hermiona	|T, AT, ATBT, AW|W, BWAW|W	5	93,763
minerva	|T, AT, ATBT, AW|W, BW|W	5	93,766
šemík2	|T, AT, ATBT, |W, AW	5	93,805
šemík	|T, AT, ATBT, |W, AW, BW	6	93,881
hatatitla	|T, AT, ATBT, |W, AW, BW, AW|W	7	93,929
ijáček	|T, AT, ATBT, |W, AW, BW, AW|W, BW|W	8	93,929
kryštůfek	|T, AT, ATBT, BT, |W, AW, BW, AW|W	8	94,118
prasátko	|T, AT, ATBT, BT, CT, |W, AW, BW, AW|W	9	93,961
klokánek	|T, AT, ATBT, BT, |W, AW, BW, AW|W, ATAW	9	94,109
klokanice	|T, AT, ATBT, BT, |W, AW, BW, AW|W, ATAW, BTBW	10	94,047

Tabulka 15. Přehled verzí používajících značky a slova na posledních třech pozicích

verze 1	verze 2	t	úroveň významnosti
jennefer	*gerald*	19,237	0,001
	marigold	23,498	0,001
marigold	*lavender*	10,420	0,001
	hermiona	-0,986	-
	minerva	-1,128	-
lavender	*hatatitla*	3,105	0,05
minerva	*ijáček*	9,101	0,001
gerald	*šemík2*	17,714	0,001
šemík2	*šemík*	3,602	0,05
šemík	*hatatitla*	3,248	0,05
hatatitla	*ijáček*	-0,015	-
	kryštůfek	42,801	0,001
kryštůfek	*prasátko*	-7,739	0,01
	klokánek	-0,668	-
klokánek	*klokanice*	-6,554	0,01

Tabulka 16. Srovnání verzí používajících značky a slova na posledních třech pozicích



Graf 7. Uspořádání verzí používajících značky a slova na posledních třech pozicích

8.3.2. Následující slova

Jestliže se předchozí slova ukázala jako významná pro předpověď aktuální značky, očekával jsem vliv i slova následujícího. Navíc slovní forma je jeden z mála údajů, který je o následujícím slovu bezpečně znám, neboť značka mu ještě nebyla vybrána.

Slovní forma následujícího slova vedla ke zlepšení pouze jako samostatný rys (*kolčava*). Slovní bigram z aktuálního a následujícího slova vedl jen k mírnému a statisticky nevýznamnému zlepšení – ať už ve verzi rané (*kolčava4*), tak při pozdějším přidání k verzi *nastěnka*. Tento bigram ve spojení s následujícím slovem samostatně dokonce vedl k mírnému zhoršení (*kolčava2*). Ani použití slovní formy dvě pozice vpřed nevedlo ke zlepšení (*kolčava3*).

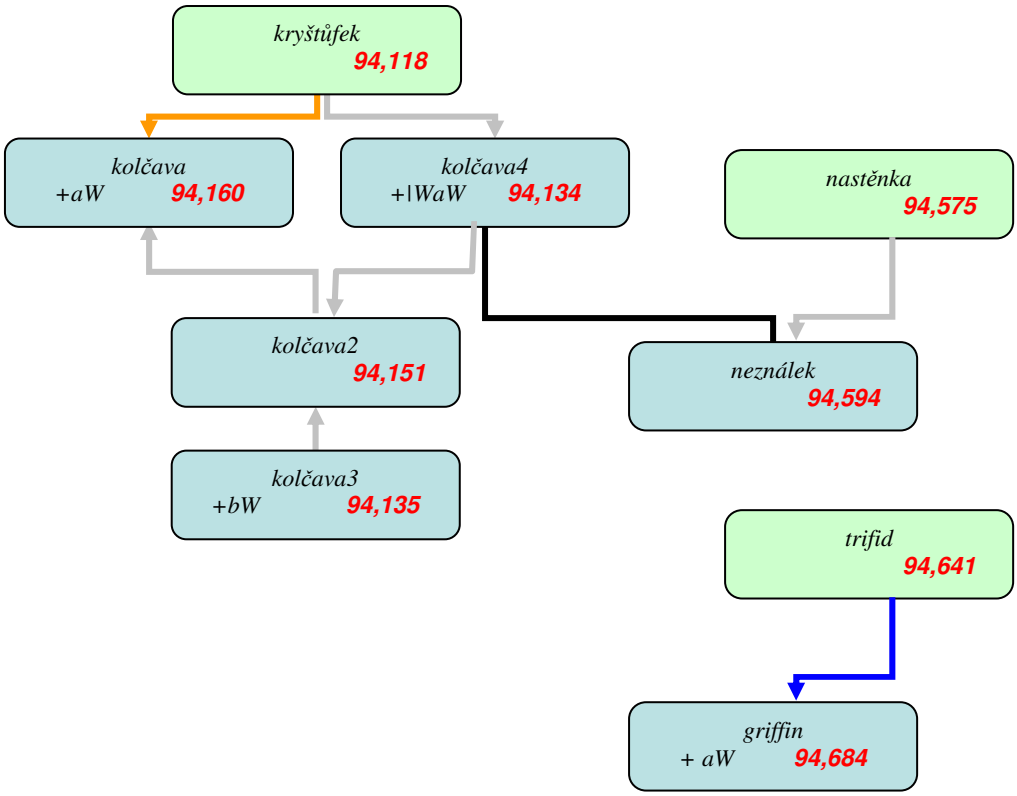
Pro další vývoj se tedy hodil pouze rys aW, který byl později přidán k verzi *trifid*, čímž vznikla úspěšná verze *griffin*.

verze	**rysy**	**počet**	**úspěšnost**
kryštůfek	|T, AT, ATBT, BT, |W, AW, BW, AW|W	8	94,118
kolčava	|T, AT, ATBT, BT, |W, AW, BW, AW|W, aW	9	94,160
kolčava4	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |WaW	9	94,134
kolčava2	|T, AT, ATBT, BT, |W, AW, BW, AW|W, aW, |WaW	10	94,151
kolčava3	|T, AT, ATBT, BT, |W, AW, BW, AW|W, aW, |WaW, bW	11	94,135
nastěnka	|T, AT, ATBT, |W, AW, BW, BT, AC, |O, VT, VL	11	94,575
neználek	|T, AT, ATBT, |W, AW, BW, BT, AC, |O, VT, VL, |WaW	12	94,594
trifid	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, UT, UL, AL	13	94,641
griffin	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, UT, UL, AL, aW	14	94,684

Tabulka 17. Přehled verzí používajících následující slova

verze 1	**verze 2**	**t**	**úroveň významnosti**
kryštůfek	*kolčava*	4,756	0,01
	kolčava4	0,995	-
kolčava4	*kolčava2*	1,156	-
	neználek	25,108	0,001
kolčava	*kolčava2*	-1,289	-
kolčava2	*kolčava3*	-1,553	-
nastěnka	*neználek*	1,777	-
trifid	*griffin*	3,282	0,05

Tabulka 18. Srovnání verzí používajících následující slova



Graf 8. Uspořádání verzí používajících následující slova

8.3.3. Lemma

Předpokládal jsem, že pokud použiji lemma aktuálního slova, algoritmus se naučí každému lemmatu přiřazovat v případě nerozhodnosti častější značku, se kterou se dané slovo vyskytuje. Ukázalo se však, že to není perspektivní. V rané verzi (*pú*) sice nepatrné zlepšení nastalo, ale později (*bazilišek)* už aktuální lemma vedlo dokonce významnému zhoršení. Zřejmě tedy aktuální lemma není dobrým rysem.

Naproti tomu lemma na předchozí pozici se ukázalo jako prospěšné. Vedlo ke zlepšení ve verzi *pú2*, takže bylo později aplikováno i na komplikovanější verzi *kostěj2b*, opět úspěšně (verze *trifid*). Význam předchozího lemmatu je v tom, že určitá slova (bez ohledu na svůj tvar) vyžadují jistý tvar, který za nimi bude následovat. Typicky to mohou být například slovesa. Díky použití lemmatu se pak snižuje počet rysů, které z daného slova předpovídají následující značku. Tam, kde mohlo selhat použití předchozí slovní formy, protože se

nemusela v trénovacích datech vůbec objevit, může pomoci právě lemma, pokud se slovo v trénovacích datech objevilo v jiném tvaru.

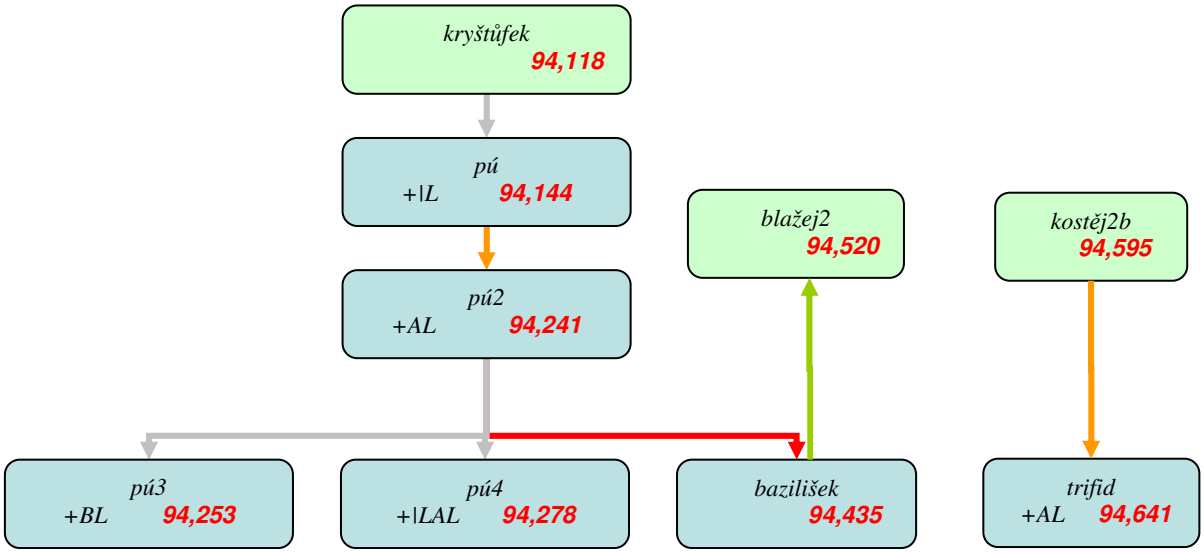
Komplikovanější využití lemmat (dvojice lemmat na předchozí a aktuální pozici – verze *pú4*) nepřineslo významné zlepšení, takže jsem od něj upustil. Ani lemma dvě pozice zpět nevedlo k výraznému zlepšení. Pro další vývoj se tedy z celé větve uplatnil pouze rys s aktuálním lemmatem.

verze	**rysy**	**počet**	**úspěšnost**
kryštůfek	lT, AT, ATBT, BT, lW, AW, BW, AWlW	8	94,118
pú	lT, AT, ATBT, BT, lW, AW, BW, AWlW, lL	9	94,144
pú2	lT, AT, ATBT, BT, lW, AW, BW, AWlW, lL, AL	10	94,241
pú3	lT, AT, ATBT, BT, lW, AW, BW, AWlW, lL, AL, BL	11	94,253
pú4	lT, AT, ATBT, BT, lW, AW, BW, AWlW, lL, AL, lLAL	11	94,278
blažej2	lT, AT, ATBT, lW, AW, BW, BT, lO, VT, VL	10	94,520
bazilišek	lT, AT, ATBT, lW, AW, BW, BT, lO, VT, VL, lL, AL	12	94,435
kostěj2b	lT, AT, ATBT, lW, AW, BW, BT, lO, VT, VL, UT, UL	12	94,595
trifid	lT, AT, ATBT, lW, AW, BW, BT, lO, VT, VL, UT, UL, AL	13	94,641

Tabulka 19. Přehled verzí používajících lemmata

verze 1	**verze 2**	**t**	**úroveň významnosti**
kryštůfek	*pú*	1,812	-
pú	*pú2*	7,987	0,01
pú2	*pú3*	0,703	-
	pú4	2,160	-
	bazilišek	11,389	0,001
blažej2	*bazilišek*	-4,312	0,02
kostěj2b	*trifid*	5,675	0,01

Tabulka 20. Srovnání verzí používajících lemmata



Graf 9. Srovnání verzí používajících lemmata

8.3.4. Pořadí slova ve větě

Určitou roli pro určování morfologické značky má také pořadí slova ve větě. Některé značky se objevují spíše na začátku věty, jiné jsou na začátku věty nepravděpodobné. Je ale potřeba pořadí slova ve větě nějak omezit, protože je zjevně jedno, je-li slovo ve větě dvacáté nebo dvacáté první. Nejvyšší vyzkoušenou hodnotou na omezení pořadí bylo číslo 15, při něm však došlo dokonce ke zhoršení výsledku. Z provedených pokusů se jako nejlepší ukázalo omezení pořadí na číslo 5. To znamená, že pro všechna slova, která měla pořadí 5 a vyšší, byl vygenerován rys lO5.

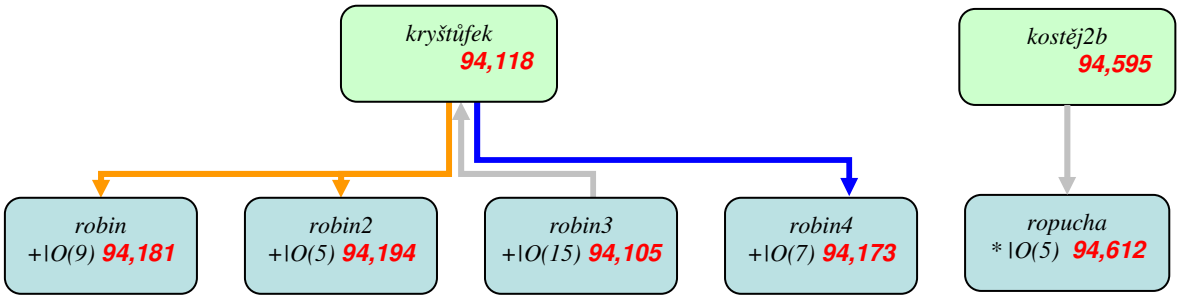
Na původních datech (PDT 1.0) ovšem těsně zvítězilo číslo 9, takže další větve obsahovaly omezení 9. Přestože zlepšení mezi verzemi *kostěj2b* a *ropucha* není statisticky významné, od následníků verze *kostěj2b* jsem se vrátil k číslu 5. Jedná se totiž o malou změnu rysu, nikoliv jeho přidání. Tím pádem není statistická významnost pro obhájení této změny potřebná.

verze	rysy	počet	úspěšnost
kryštůfek	|T, AT, ATBT, BT, |W, AW, BW, AW|W	8	94,118
robin	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O(9)	9	94,181
robin2	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O(5)	9	94,194
robin3	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O(15)	9	94,105
robin4	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O(7)	9	94,173
kostěj2b	|T, AT, ATBT, |W, AW, BW, BT, |O(9), VT, VL, UT, UL	12	94,595
ropucha	|T, AT, ATBT, |W, AW, BW, BT, |O(5), VT, VL, UT, UL	12	94,612

Tabulka 21. Přehled verzí používajících pořadí slova ve větě

verze 1	verze 2	t	úroveň významnosti
kryštůfek	*robin*	4,660	0,01
	robin2	4,638	0,01
	robin3	-0,600	-
	robin4	3,497	0,05
kostěj2b	*ropucha*	1,761	-

Tabulka 22. Srovnání verzí používajících pořadí slova ve větě



Graf 10. Uspořádání verzí používajících pořadí slova ve větě

8.3.5. Předchozí pády

Předchozí pád představuje čtvrtý znak značky na předchozí pozici. Zkusil jsem ho použít, abych posílil shodu jednak mezi jmény a jednak shodu předložky a následujícího jména. Protože se však v datech vyskytují poměrně časté sekvence jmen v různých pádech (například neshodné přívlastky, dvě jména zaplňující různé valenční pozice slovesa – „řekl otci pravdu“), je použití tohoto rysu sporné. Experiment ukázal, že předchozí pád vede pouze k nevýznamnému zlepšení. Významněji pomohlo spojení dvou předchozích pádů do jednoho rysu. Pád dvě pozice zpět samostatně opět výrazněji nepomohl.

Zkoušel jsem také vyhledat nejbližší definovaný pád (tedy jakoukoliv hodnotu čtvrtého znaku značky kromě „-“) až pět pozic zpět. Tento pokus však přinesl jen výrazné zhoršení (*zajíc*, *zajíc2*).

Pád na předchozí pozici byl vyzkoušen i na pozdějších verzích, kde většinou přinesl mírné, ale statisticky nevýznamné zlepšení (*baba*, *jaga*).

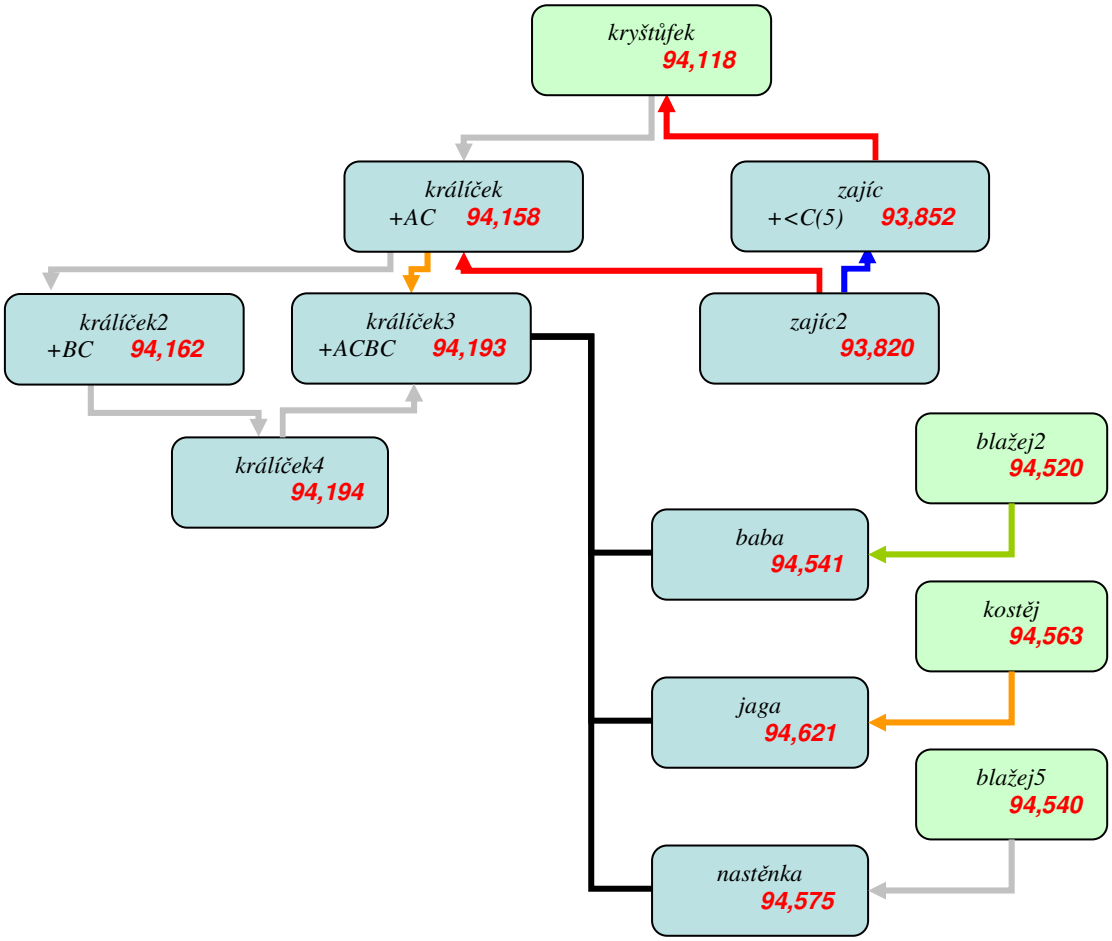
Použití předchozích pádů případně i pro předpověď aktuálního pádu se nakonec se nakonec ukázalo jako vhodné až teprve ve spojení s hodnotou slovního poddruhu (druhý znak značky).

verze	**rysy**	**počet**	**úspěšnost**
kryštůfek	|T, AT, ATBT, BT, |W, AW, BW, AW|W	8	94,118
králíček	|T, AT, ATBT, BT, |W, AW, BW, AW|W, AC	9	94,158
králíček2	|T, AT, ATBT, BT, |W, AW, BW, AW|W, AC, BC	10	94,162
králíček3	|T, AT, ATBT, BT, |W, AW, BW, AW|W, AC, ACBC	10	94,193
králíček4	|T, AT, ATBT, BT, |W, AW, BW, AW|W, AC, BC, ACBC	11	94,194
zajíc	|T, AT, ATBT, BT, |W, AW, BW, AW|W, <C(5)	9	93,852
zajíc2	|T, AT, ATBT, BT, |W, AW, BW, AW|W, AC, <C(5)	10	93,820
blažej2	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL	10	94,520
baba	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, AC	11	94,541
kostěj	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, UT, UL	12	94,563
jaga	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, UT, UL, AC	13	94,621
blažej5	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL	10	94,540
nastěnka	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, AC	11	94,575

Tabulka 23. Přehled verzí používajících předchozí pády

verze 1	**verze 2**	**t**	**úroveň významnosti**
kryštůfek	králíček	2,339	-
	zajíc	-12,422	0,001
zajíc	zajíc2	-3,178	0,05
králíček	králíček2	0,556	-
	králíček3	4,698	0,01
	zajíc2	-100,609	0,001
	baba	17,278	0,001
	jaga	18,862	0,001
	nastěnka	15,492	0,001
králíček2	králíček4	2,285	-
králíček3	králíček4	0,039	-
blažej2	baba	4,005	0,02
kostěj	jaga	4,882	0,01
blažej5	nastěnka	1,703	-

Tabulka 24. Srovnání verzí používajících předchozí pády



Graf 11. Srovnání verzí používajících předchozí pády

8.3.6. Předchozí sloveso

Předchozí sloveso má zcela jistě zásadní vliv především na pády následujících jmen. Svou roli hraje nejenom značka předchozího slovesa, ale také jeho lemma. Ukázalo se, že význam lemmatu je dokonce větší. To plyne zřejmě z toho, že lemma slovesa určuje jeho valenci, a tím i určuje pád, který by měl následovat. Důležitým parametrem, na kterém závisí úspěšnost slovesných rysů, je maximální vzdálenost, do jaké se má sloveso vyhledávat, a případně hranice, které se nepřekročí ani tehdy, není-li maximální vzdálenost dosažena.

Samozřejmým a nijak netestovaným omezením byla hranice věty. Vliv slovesa přes tuto hranici je minimální, a byl proto od začátku vyloučen. Testované maximální vzdálenosti byly 5, 10, 20 a 30 slov zpět. 5 slov se ukázalo jako nedostačujících (pouze značka: *tygr*), 10 slov už vedlo k výraznému zlepšení (pouze značka: *štrosmajer*, pouze lemma: *cvach*, obojí: *blažej*), ale 20 slov vedlo k ještě lepším výsledkům (pouze lemma: *cvach2*, obojí: *blažej2*).

Naproti tomu 30 slov už je zřejmě příliš, vedlo to k nepatrnému zhoršení (*blažej4*), a proto byla hranice pro další vývoj zafixována na 20 slov.

Protože se zdálo, že na vzdálenost 20 slov může sloveso působit i kontraproduktivně (například přes hranici vedlejších vět a podobně), provedl jsem několik experimentů s dalšími omezeními. Omezení vyhledávání přes jakoukoliv interpunkci vedlo k významnému zhoršení (*blažej3*), důvodem jsou zřejmě několikanásobné jmenné fráze, vsuvky a podobně, které působení slovesa neruší.

Nepatrné zlepšení (avšak nevýznamné) přineslo omezení vyhledávání přes vztažná zájmena (*blažej5*). Tím jsem chtěl docílit toho, aby sloveso věty nadřazené nemělo vliv na značky ve větě podřazené alespoň tam, kde lze tuto hranici rozpoznat. To se právě zdálo možné u vedlejších vět začínajících vztažným zájmenem, kterých je relativně hodně. Druhým krokem pak bylo zamezení vyhledávání přes podřadící spojky (*blažej8*). Přestože oba dva kroky (omezení přes vztažná zájmena i přes podřadící spojky) nevedly ke statisticky významnému zlepšení, porovnání verzí *blažej2* a *blažej8* už statisticky významné zlepšení dává.

Dalším pokusem bylo negenerování rysů s nejbližším slovesem tam, kde nebylo nalezeno (*blažej6*). To přineslo zhoršení, neboť patrně i fakt, že se v blízkosti žádné sloveso nenachází, má svůj význam. Pokusil jsem se také zařadit mezi slovesa i gerundivní tvar adjektiva (*blažej7*). I to však vedlo k mírnému zhoršení.

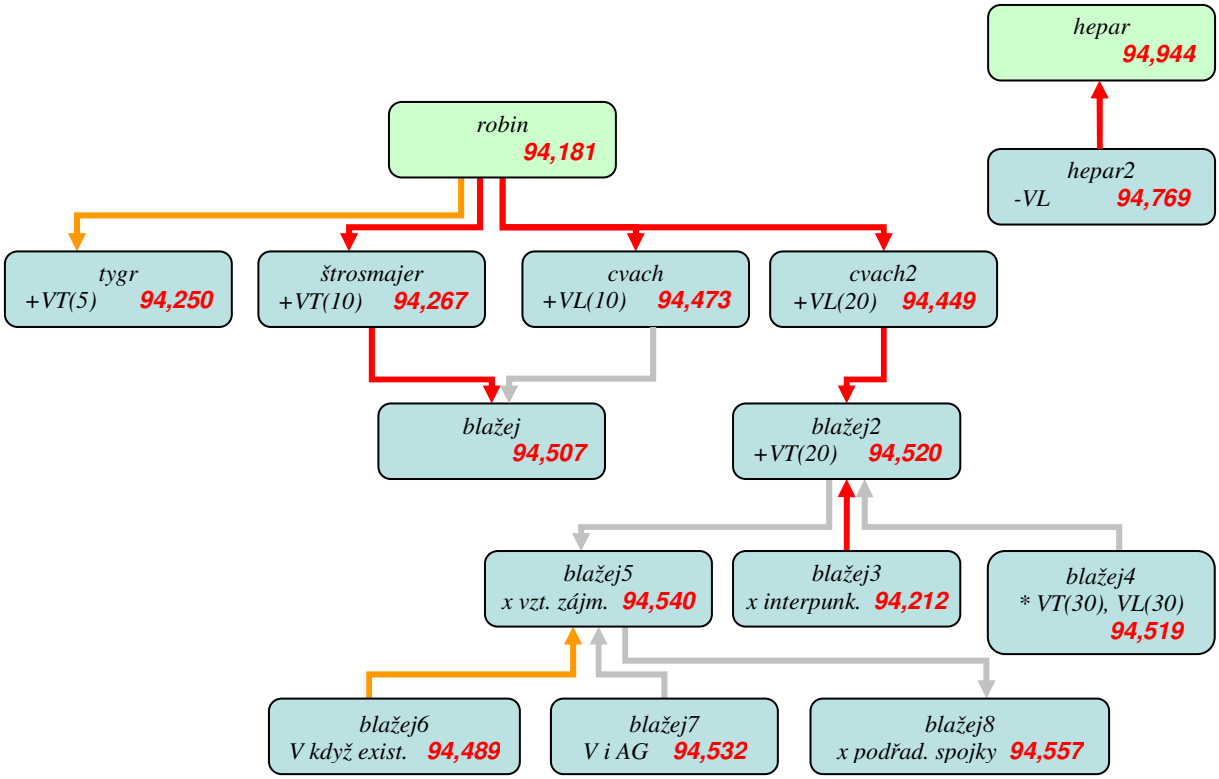
Pro další experimenty byla použita verze *blažej2*, která se ukázala jako nejlepší na původních datech.

Při rozboru verze *lucifer4*[1] se ukázalo, že odebrání rysu VL (lemma slovesa) může dokonce přinést nepatrné zlepšení. Proto u nejlepší verze na nových datech (*hepar*) jsem provedl obdobný pokus (*hepar2*). Výsledkem však bylo podstatné zhoršení, takže rys VL byl v nejlepší verzi ponechán.

[1] viz 8.3.13.

verze 1	verze 2	t	úroveň významnosti
robin	*tygr*	6,903	0,01
	cvach	17,698	0,001
	cvach2	10,654	0,001
	štrosmajer	8,842	0,001
cvach	*blažej*	1,757	-
štrosmajer	*blažej*	13,315	0,001
cvach2	*blažej2*	18,696	0,001
blažej2	*blažej3*	-11,252	0,001
	blažej4	-0,061	-
	blažej5	1,289	-
	blažej8	4,676	0,01
blažej5	*blažej6*	-7,850	0,01
	blažej7	-1,191	-
	blažej8	0,907	-
hepar	*hepar2*	-16,295	0,001

Tabulka 25. Srovnání verzí používajících nejbližší předcházející sloveso



Graf 12. Uspořádání verzí používajících nejbližší předcházející sloveso

verze	**rysy**	**počet**	**úspěšnost**
robin	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O	9	94,181
tygr	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VT(5)	10	94,250
cvach	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VL(10)	10	94,473
cvach2	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VL(20)	10	94,449
štrosmajer	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VT(10)	10	94,267
blažej	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VT(10), VL(10)	11	94,507
blažej2	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VL(20), VT(20)	11	94,520
blažej3	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VT(20), VL(20) *VT a VL se nevyhledává přes interpunkci*	11	94,212
blažej4	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VT(30), VL(30)	11	94,519
blažej5	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VT(20), VL(20) *VT a VL se nevyhledává přes vztažná zájmena*	11	94,540
blažej6	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VT(20), VL(20) *VT a VL se nevyhledává přes vztažná zájmena* *a generuje se jen tam, kde je nalezeno*	10 - 11	94,489
blažej7	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VT(20), VL(20) *VT a VL se nevyhledává přes vztažná zájmena* *za sloveso je považován i gerundivum (značka AG...)*	11	94,532
blažej8	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VT(20), VL(20) *VT a VL se nevyhledává přes vztažná zájmena* *a přes podřadící spojky*	11	94,557
hepar	|T, AT, ATBT, |W, AW, BW, BT, |O, VT(20), VL(20), UT, UL, AL, aW, |Q, AQ|Q, AQBQ|Q, |U(0,1), |V(0,1,2)	19	94,944
hepar2	|T, AT, ATBT, |W, AW, BW, BT, |O, VT(20), UT, UL, AL, aW, |Q, AQ|Q, AQBQ|Q, |U(0,1), |V(0,1,2)	18	94,769

Tabulka 26. Přehled verzí používajících nejbližší předcházející sloveso

8.3.7. Následující sloveso

Vzhledem k tomu, že čeština má poměrně dost volný slovosled, stává se často, že sloveso, které určuje tvar jmen, se nachází ve větě až za nimi. Proto bylo zapotřebí vyhledávat také slovesa následující za aktuálním slovem. Problémem ovšem je, že následující značky nejsou Viterbiho algoritmem dosud určeny, a tak je možné vyhledávat pouze možná slovesa, tedy slova, mezi jejichž možnými značkami je alespoň jedna značka slovesná. To sice není tak spolehlivé, jako vyhledávání sloves předcházejících, nicméně jisté zlepšení to přineslo.

Nevyhneme se bohužel případům, kdy je jako sloveso vyhledáno úplně jiné slovo (například číslovka „tři“ jako rozkazovací způsob slovesa „třít“), ale algoritmus tyto chybné údaje zřejmě sám vyloučí tím, že jim nepřiřadí velkou váhu.

Opět zde hrají roli omezení, přes která se sloveso již vyhledávat nebude. Samozřejmým omezením byla hranice věty. Jako maximální vzdálenost se však ukázala lepší hranice pouze 10 slov (*kostěj*) než hranice 20 slov (*kostěj2*). To je zřejmě dáno jednak zmíněnou nejistotou v určování následujících sloves a do jisté míry může být i vlastností češtiny obecně, že slova, jejichž tvar je určen slovesem, se od tohoto slovesa nemohou vyskytovat tak daleko vlevo jako vpravo.

Dalším pokusem bylo vyhledávat sloveso vpravo jen tam, kde nebylo nalezeno žádné sloveso vlevo (*blažej5*). To ovšem ke zvýšení úspěšnosti nevedlo.

Ve verzích *kostěj* a *kostěj2* byla zahrnuta do vyhledávání i aktuální pozice. Ve verzi *kostěj2b* byla aktuální pozice vynechána, což dalo téměř shodný výsledek, ale pro další vývoj byla použita právě tato verze, protože namátkovým otestováním[1] na dalších podobných verzích se ukázalo, že zahrnutí aktuální pozice do vyhledávání nepatrně škodí.

Pro další vývoj byla použita verze *kostěj2*.

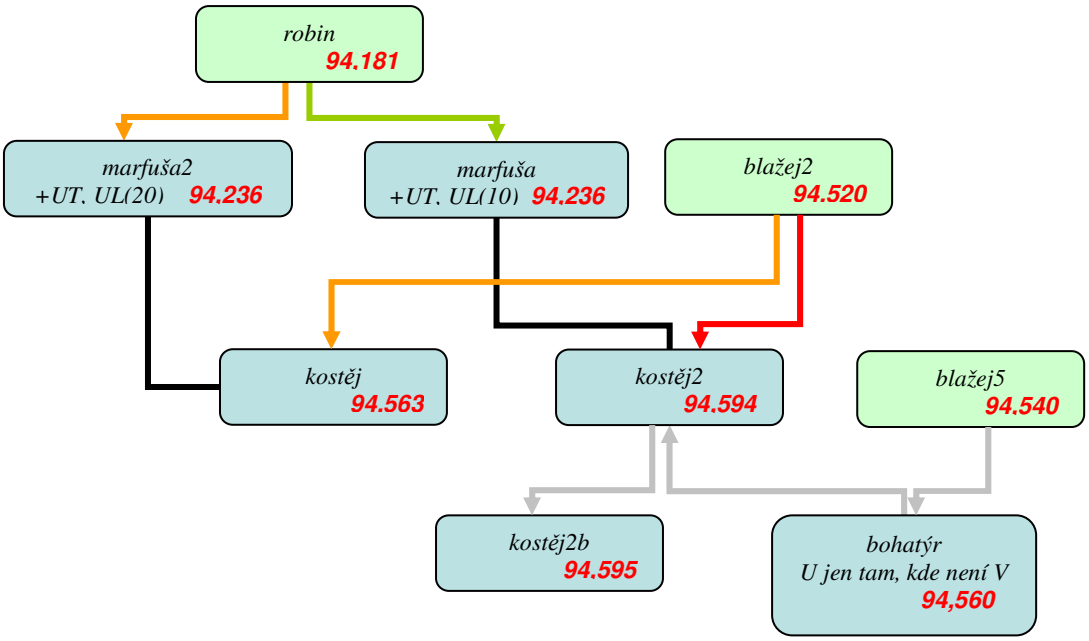
[1] Toto otestování však nebylo provedeno na všech pěti blocích dat, takže jej v přehledech neuvádím.

verze	rysy	počet	úspěšnost
robin	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O	9	94,181
marfuša	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, UT(10), UL(10)	11	94,236
marfuša2	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, UT(20), UL(20)	11	94,236
blažej2	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VL(20), VT(20)	11	94,520
kostěj	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VL(20), VT(20), UL(20), UT(20)	13	94,563
kostěj2	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VL(20), VT(20), UL(10), UT(10) *do UT a UL zahrnuto i aktuální slovo*	13	94,594
kostěj2b	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VL(20), VT(20), UL(10), UT(10)	13	94,595
blažej5	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VT(20), VL(20) *VT a VL se nevyhledává přes vztažná zájmena*	11	94,540
bohatýr	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VT(20), VL(20), UT(10), UL(10) *VT a VL se nevyhledává přes vztažná zájmena* *UT a UL jen tam, kde není VT a VL*	11	94,560

Tabulka 27. Přehled verzí používajících nejbližší následující sloveso

verze 1	verze 2	t	úroveň významnosti
robin	*marfuša*	3,968	0,02
	marfuša2	5,823	0,01
blažej2	*kostěj*	4,779	0,01
	kostěj2	10,537	0,001
marfuša	*kostěj2*	17,143	0,001
marfuša2	*kostěj*	15,716	0,001
kostěj2	*bohatýr*	-2,260	-
blažej5	*bohatýr*	1,496	-
kostěj2	*kostěj2b*	0,064	-

Tabulka 28. Srovnání verzí používajících nejbližší následující sloveso



Graf 13. Uspořádání verzí používajících nejbližší následující sloveso

8.3.8. Slovesa a aktuální pád

Tato část vývoje vychází ze snahy prohloubit vazbu mezi slovesem a pádem, který sloveso může vyžadovat.

Na původních datech se osvědčilo použití předchozího slovesa pro předpověď aktuálního pádu. Aktuální pád byl spojen jak se značkou předchozího slovesa, tak s jeho lemmatem. Teprve v závěru vývoje na původních datech se ukázalo jako prospěšné spojení aktuálního pádu se značkou slovesa opět odebrat (původně nejúspěšnější verze *lucifer4*).

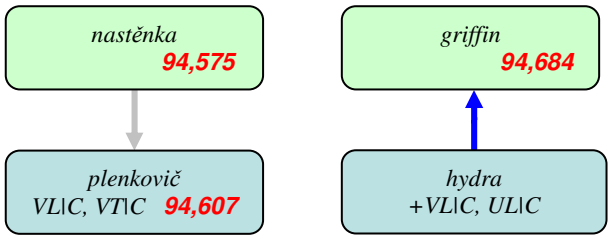
Ověření na nových datech prospěšnost předpovědi aktuálního pádu ze slovesa nepotvrdilo. Přidání vazby mezi předchozím slovesem a aktuálním pádem k verzi *nastěnka* sice vedlo k mírnému, avšak statisticky nevýznamnému zlepšení (*plenkovič*). K pozdější verzi *griffin* (která používá slovesa předchozí i následující) jsem zkusil přidat vazbu pouze lemmat obou sloves na aktuální pád. Výsledkem je statistický významné zhoršení (*hydra*), takže jsem od této větve vývoje zcela upustil.

verze	rysy	počet	úspěšnost
nastěnka	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, AC	11	94,575
plenkovič	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, AC, VL|C, VT|C	13	94,607
griffin	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, UT, UL, AL, aW	14	94,684
hydra	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, UT, UL, AL, aW, VL|C, UL|C	16	94,588

Tabulka 29. Přehled verzí používajících nejbližší sloveso a aktuální pád

verze 1	verze 2	t	úroveň významnosti
nastěnka	plenkovič	2,078	-
griffin	hydra	-3,450	0,05

Tabulka 30. Srovnání verzí používajících nejbližší sloveso a aktuální pád



Graf 14. Uspořádání verzí používajících nejbližší sloveso a aktuální pád

8.3.9. Zvratné zájmeno „se“

Nejčastější chybou bývá záměna prvního a čtvrtého pádu u jmen. Z chybových výpisů vyplynulo, že k ní dochází často tam, kde se objevuje zvratná forma slovesa (například „...on stahoval **králíka**...“ a „...v neděli se stahoval **králík**...“). Právě výskyt zvratného zájmena „se“ může přinést důležitou informaci pro rozhodování mezi prvním a čtvrtým pádem.

„se“ se v češtině zpravidla vyskytuje na počátečních pozicích ve větě a ovlivňuje pád následujících jmen. Proto bylo vyhledáváno od aktuálního slova pouze doleva, nejdále 20 pozic a nikoliv za hranicí věty. V textu se však často vyskytuje i „se“ jako předložka sedmého

pádu. Za touto předložkou se však nutně vyskytuje i jméno v sedmém pádu, takže vyhledávání zvratného „se“ se zarazilo i v případě nalezení sedmého pádu.

Základní verze *agáta* přinesla pouze nepatrné a statisticky nevýznamné zlepšení. Proto byla provedena řada pokusů s dalšími omezeními na tento rys.

Podle chybových matic se sice snížil počet chyb v záměně prvního a čtvrtého pádu, ale jinde počet chyb zase narostl. Prvním omezením tedy bylo generování rysu pouze pro značky, které na pozici pádu obsahují 1 nebo 4 (*agáta2)*. To se osvědčilo, i když statisticky nevýznamně. Dále bylo generování omezeno pouze na podstatná a přídavná jména (*agáta3*), to se však neosvědčilo, takže od tohoto omezení jsem v dalších krocích ustoupil. Ve verzi *agáta4* bylo oproti verzi *agáta2* hledání navíc zastaveno na jakékoliv interpunkci. Ve verzi *agáta5* se hledání zastavilo i při nalezení jakéhokoliv prvního pádu. Ve verzi *agáta6* se hledání zastavilo při nalezení jakéhokoliv podstatného jména – to bylo motivováno úvahou, že „se“ má vliv pouze na první následující jméno, nikoliv už na další. V navazující verzi *agáta8* byl rys generován jen tam, kde zvratné „se“ bylo nalezeno. Ve verzi *agáta7* platila stejná omezení jako v *agáta6*, ale rys byl generován jen pro značky obsahující první pád.

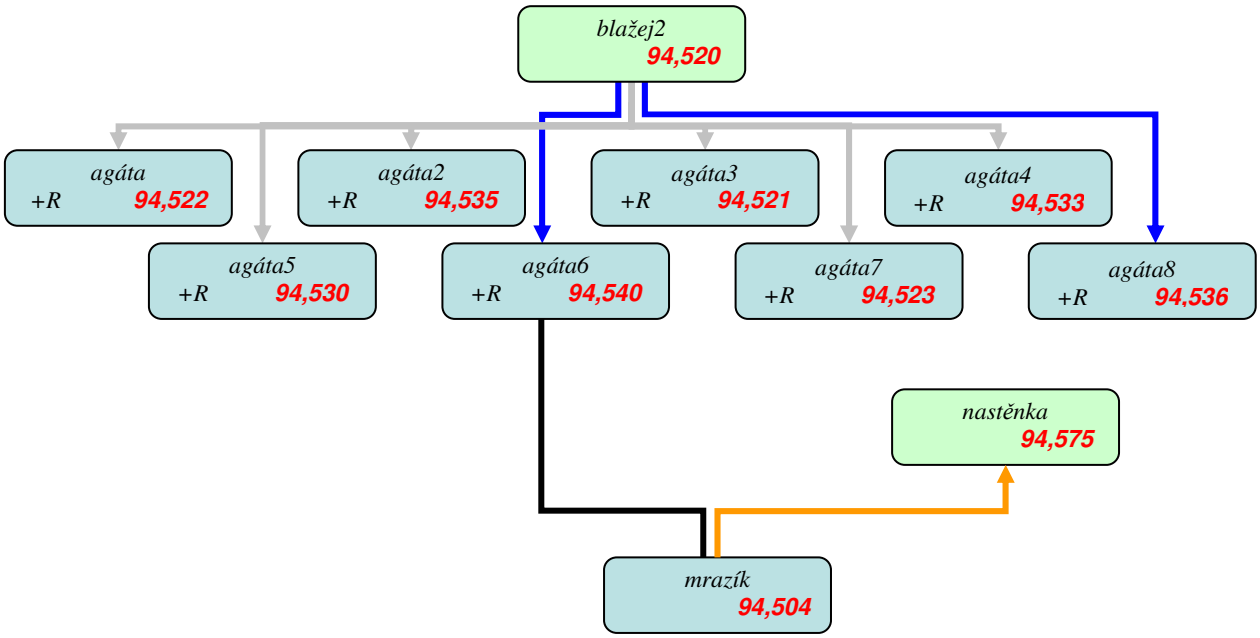
Celkový výsledek této série pokusů je nepatrné zlepšení. Jediné dvě verze dokázaly přinést zlepšení, které je proti verzi *blažej2* statisticky významné (*agáta6* a *agáta8*). Přestože intuitivně se zdálo, že by tyto rysy mohly řešit určený problém, jejich přínos byl nepatrný. Po zkombinování alespoň trochu přijatelné verze *agáta6* s pozdější verzí *nastěnka* došlo dokonce ke statisticky významnému zhoršení. V dalších experimentech proto bylo od tohoto rysu upuštěno.

verze 1	**verze 2**	**t**	**úroveň významnosti**
blažej2	*agáta*	0,173	-
	agáta2	2,312	-
	agáta3	0,065	-
	agáta4	2,744	-
	agáta5	1,617	-
	agáta6	3,347	0,05
	agáta7	0,374	-
	agáta8	3,130	0,05
agáta6	*mrazík*	-3,084	0,05
nastěnka	*mrazík*	-8,399	0,01

Tabulka 31. Srovnání verzí používajících zvratné „se“

verze	**rysy**	**počet**	**úspěšnost**
blažej2	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VL(20), VT(20)	11	94,520
agáta	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VL(20), VT(20), R	12	94,522
agáta2	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VL(20), VT(20), R *R generováno jen pro 1. a 4. pád*	11 - 12	94,535
agáta3	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VL(20), VT(20), R *R generováno jen pro 1. a 4. pád pro slovní druhy A a N*	11 – 12	94,521
agáta4	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VL(20), VT(20), R *R generováno jen pro 1. a 4. pád, nehledá se přes interpunkci*	11 – 12	94,533
agáta5	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VL(20), VT(20), R *R generováno jen pro 1. a 4. pád, nehledá se přes interpunkci a 1. pád*	11 – 12	94,530
agáta6	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VL(20), VT(20), R *R generováno jen pro 1. a 4. pád, nehledá se přes interpunkci a 1. pád a přes podstatná jména*	11 – 12	94,540
agáta7	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VL(20), VT(20), R *R generováno jen pro 1. pád, nehledá se přes interpunkci a 1. pád a přes podstatná jména*	11 – 12	94,523
agáta8	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VL(20), VT(20), R *R generováno jen tam, kde je pozitivní a jen pro 1. a 4. pád, nehledá se přes interpunkci a 1. pád a přes podstatná jména*	11 – 12	94,536
nastěnka	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, AC	11	94,575
mrazík	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, AC, R *R generováno jen pro 1. a 4. pád, nehledá se přes interpunkci a 1. pád a přes podstatná jména*	12	94,504

Tabulka 32. Přehled verzí používajících zvratné „se“



Graf 15. Uspořádání verzí používajících zvratné „se“

8.3.10. Velikost písmen

Velikost písmen hraje důležitou roli v tom, že umožňuje rozlišit vlastní jména. K zohlednění velikosti písmen mě přivedlo pročítání chybových výpisů, kde se velice často chybovalo právě ve vlastních jménech. Další početnou skupinu v chybových výpisech tvořily zkratky.

Zvolil jsem dvě cesty: jednak rys založený na velikosti písmen slovní formy (což způsobuje zahrnutí prvních slov ve větě a všech slov ve větách psaných velkými písmeny), jednak rys založený na velikosti písmen lemmatu (tím se zmíněné nežádoucí případy odstraní).

V obou případech mohl rys popisující velikost písmen nabývat tří hodnot:

- 0: první písmeno je malé
- 1: první písmeno je velké
- 2: druhé písmeno je velké

Hodnota 1 u lemmatu ve většině případů označuje vlastní jméno. Hodnota 2 v drtivé většině případů znamená, že se jedná o slovo celé psané velkými písmeny. Řídké případy, kdy

druhé písmeno je velké z jiného důvodu, jsem zanedbal. U lemmatu pak hodnota 2 signalizuje zkratku.

V obou případech jsem vyzkoušel tři varianty použitých hodnot: 0, 1, 2; 0, 1 a 0, 2. U slovní formy stejně jako u lemmatu se jako nejlepší ukázalo použití všech tří hodnot. Rozdíly při vynechání hodnoty 2 však byly v obou případech zcela nepatrné.

Na původních datech ovšem u slovní formy zvítězila varianta 0, 1. Proto byla spojena varianta úspěšná varianta *ilja* a *muromec* do jedné verze *lucifer*. Ta přinesla opět zlepšení, byť v porovnání s verzí *muromec* nebylo statisticky významné. Při zběžném otestování na verzích pozdějších se rozdíl mezi použitím hodnot 0, 1 a 0, 1, 2 na slovní formu ukázal jako zcela nevýznamný.

Na původních datech se nakonec ukázala jako absolutně nejlepší verze *lucifer4* vzniklá spojením verzí *ilja* a *muromec3* (tedy používající pro slova hodnoty 0, 2), navíc neobsahovala rys VT|C (nejbližší sloveso vpravo a aktuální pád). Důkladnější rozbor (už na nových datech) jsem tedy prováděl právě na ní[1], protože nejúspěšnější verze na nových datech ještě nebyla známa. Teprve na závěr vývoje verzí na nových datech jsem oba rysy přidal k úspěšné verzi *epiglotis*, čímž vznikla nejlepší verze *hepar*.

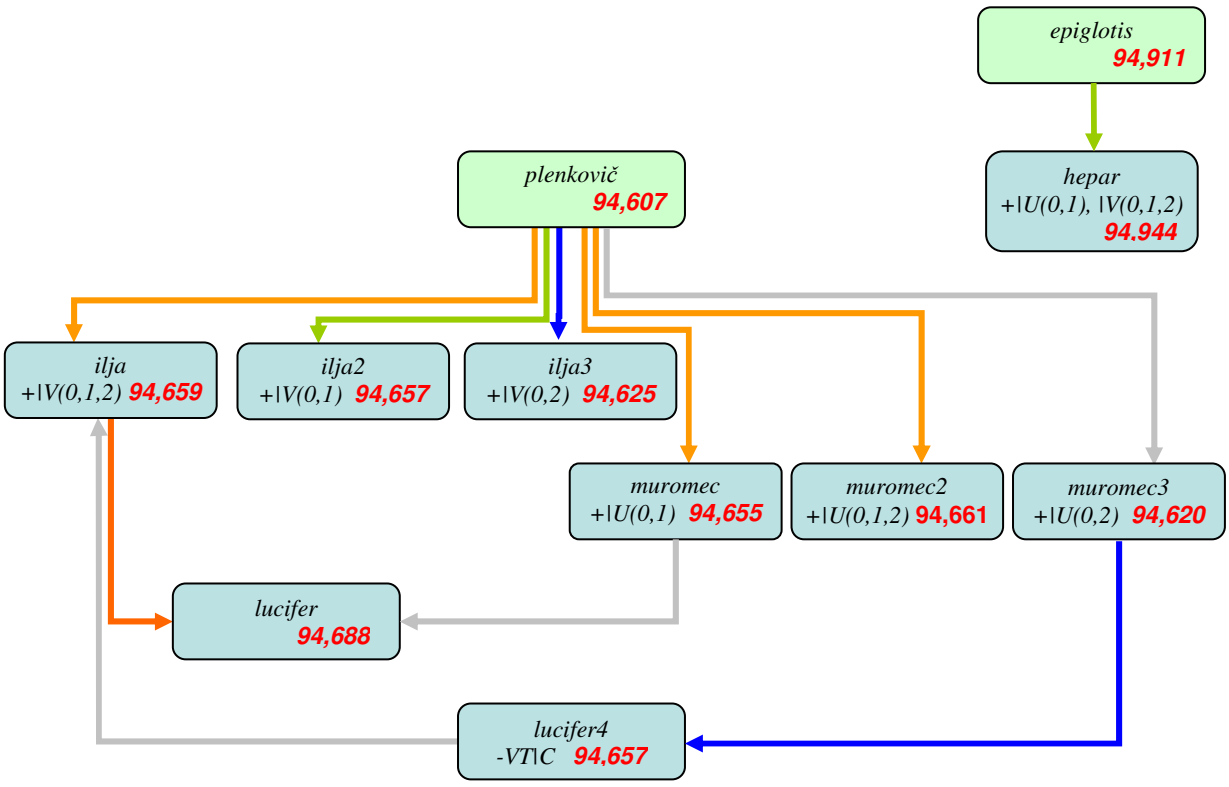
verze 1	**verze 2**	**t**	**úroveň významnosti**
plenkovič	*ilja*	5,553	0,01
	ilja2	3,983	0,02
	ilja3	2,913	0,05
	muromec	4,898	0,01
	muromec2	4,714	0,01
	muromec3	1,359	-
ilja	*lucifer*	2,895	0,05
	lucifer4	-0,194	-
muromec	*lucifer*	0,187	-
muromec3	*lucifer4*	3,316	0,05
epiglotis	*hepar*	3,971	0,02

Tabulka 33. Srovnání verzí používajících velikost písmen

[1] viz 8.3.13.

verze	rysy	počet	úspěšnost
plenkovič	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, AC, VT|C, VL|C	13	94,607
ilja	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, AC, VT|C, VL|C, |V(0,1,2)	14	94,659
ilja2	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, AC, VT|C, VL|C, |V(0,1)	14	94,657
ilja3	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, AC, VT|C, VL|C, |V(0,2)	14	94,625
muromec	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, AC, VT|C, VL|C, |U(0,1)	14	94,655
muromec2	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, AC, VT|C, VL|C, |U(0,1,2)	14	94,661
muromec3	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, AC, VT|C, VL|C, |U(0,2)	14	94,620
lucifer	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, AC, VT|C, VL|C, |V(0,1,2), |U(0,1)	15	94,688
lucifer4	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, AC, VL|C, |V(0,1,2), |U(0,2)	14	94,657
epiglotis	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, UT, UL, AL, aW, |Q, AQ|Q, AQBQ|Q	17	94,911
hepar	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, UT, UL, AL, aW, |Q, AQ|Q, AQBQ|Q, |U(0,1), |V(0,1,2)	19	94,944

Tabulka 34. Přehled verzí používajících velikost písmen



Graf 16. Uspořádání verzí používajících velikost písmen

8.3.11. Následující sloveso - předchozí označkování Morčetem

Hledání nejbližšího následujícího slovesa je problematické v tom, že značky vpravo od aktuální pozice ještě nejsou vybrány, takže se na každé pozici musí prohledávat všechny možné značky. To může ale způsobit i nežádoucí výsledky – například slovo „tři“ bude vždy vyhodnoceno jako nejbližší následující sloveso vpravo, protože se může jednat o imperativ slovesa třít. Dá se však očekávat, že mnohem častěji půjde o číslovku.

Aby se zamezilo podobným omylům, lze využít i nějaké předcházející označkování textu. Rozhodl jsem se použít předchozí označkování Morčetem, verzí *lucifer4*, kde úspěšnost v určování slovních druhů byla 98,872% (na původních evaluačních datech).

K tomuto účelu byl kód upraven tak, aby dokázal zpracovat i data Morčetem již jednou označkovaná a uložil příslušné značky do svých datových struktur.[1] Po této úpravě je možné využívat informaci o předcházejícím označkování.

Několik verzí, které používají rysy s nejbližším možným následujícím slovesem jsem upravil tak, že se vyhledává nejbližší následující sloveso z předchozího označkování. Má-li se použít jeho lemma, pak se vybere standardním způsobem jako první lemma, které odpovídá zvolené značce.

Jak se však ukázalo, ve všech otestovaných verzích vedlo využití předchozího značkování k podstatnému zhoršení výsledku. Pravděpodobnou příčinou je to, že vyhledávání nejbližšího následujícího slovesa zahrnovalo i aktuální pozici. Navíc byl algoritmus při prvním značkování trénován i použit na těch samých datech, takže úspěšnost byla značně vyšší. Tím pádem pro všechna slovesa nutně musely všechny rysy typu UT|T (nejbližší sloveso vlevo a aktuální značka) nabýt vysokých váhových koeficientů – předpovídaly totiž s téměř stoprocentní jistotou aktuální značku na základě aktuální značky. Tím pádem se ovšem ostatní rysy nenatrénovaly dostatečně dobře (byly tímto silným rysem vytlačeny). V důsledku toho na testovacích datech, která byla oproti trénovacím hůře označkována i v prvním kole, došlo k výraznému zhoršení.

Aby se tento jev odstranil, ve verzi *kostěj2* jsem vyloučil aktuální značku při hledání nejbližšího následujícího slovesa (verze *kostěj2b*). Tyto dvě verze (s původním hledáním slovesa mezi možnými značkami) byly prakticky ekvivalentní. Při použití předchozího značkování (*kostěj2b*_x) ovšem opět došlo ke zhoršení. Možnou variantou by ještě bylo rozdělení trénovacích dat na dvě části, aby předchozí označkování na nových trénovacích datech vykazovalo přibližně stejnou chybovost jako na datech testovacích. Tím by se však

[1] viz [4]

trénovací data pro druhé kolo značkování podstatně zmenšila. Z těchto důvodů jsem od dalšího využití předchozího značkování upustil.

původní verze	**úspěšnost**	**nová verze**	**úspěšnost**	**t**	**úroveň významnosti**
bohatýr	94,560	*bohatýr_x*	94,349	-17,596	0,001
kostěj	94,563	*kostěj_x*	94,219	-3,180	0,05
kostěj2	94,594	*kostěj2_x*	94,015	-13,106	0,001
kostěj2b	94,595	*kostěj2b_x*	94,401	-18,538	0,001
marfuša	94,236	*marfuša_x*	93,623	-19,923	0,001
marfuša2	94,236	*marfuša2_x*	93,576	-31,036	0,001

Tabulka 35. Přehled a srovnání verzí využívajících předchozího označkování sloves

8.3.12. Předpovídání značek po částech

Při podrobnější analýze počtu výskytů značek[1] se ukázalo, že z celkového počtu 1132 značek se jich téměř 300 vyskytuje v textu méně než třikrát, takže pro ně neexistuje po odfiltrování žádný rys. Pro mnoho dalších značek s relativně malým počtem výskytů pak existuje rysů jen velmi málo. Z tohoto důvodu jsem vytvořil také sady rysů, které nepředpovídají aktuální značku celou, jako tomu bylo doposud, ale jen její určitou část. Vytvořil jsem tedy jen částečné značkové unigramy, bigramy a trigramy. Zaměřil jsem se přitom především na pád a číslo.

Vyšel jsem z úvahy, že pro předpověď pádu hrají roli nejen pády předchozí, ale významně také slovní druh. Například za předložkou, která se váže se sedmým pádem, nemůže prakticky následovat nic jiného než sedmý pád („šel na mě se sekerou“). Naproti tomu za podstatným jménem v sedmém pádě může následovat i pád druhý („šel na mě se sekerou hajného“), předložka se třetím pádem („šel na mě se sekerou k lesu“) a podobně. Základní jednotkou pro jednu pozici tedy bylo spojení slovního poddruhu a pádu (druhá a pátá pozice ve značce). Z těchto jednotek jsem pak utvořil unigram, bigram a trigram. Tím vznikla verze *epiglotis*, která se velmi výrazně osvědčila (o více jak 0,2%). Protože jsem však přidal tři rysy najednou, zkusil jsem také jeden odebrat – a sice ten, který pouze předpovídal aktuální dvojici slovní poddruh – pád (|Q) bez jakéhokoliv předchozího kontextu (*psýché*).

[1] viz 6.3.

To se však ukázalo jako horší varianta, zřejmě i rys |Q má svůj význam v tom, že předpovídá nejčastější dvojici poddruh – pád i tam, kde není znám žádný kontext.

V dalších krocích jsem k verzi *epiglotis* přidával další obdobné rysy, které však měly předpovídat aktuální číslo (čtvrtá pozice ve značce). Použil jsem jednak samotný unigram (*fysis*), jednak samotný bigram (*fysis3*), a pak také spojení předchozí dvojice poddruh – číslo s aktuální značkou (*fysis2*). Dále jsem přidal unigram, bigram a trigram zároveň (*glotis*). Ve verzích *fysis2* a *fysis3* sice určité nepatrné zlepšení nastalo, ale nebylo statisticky významné, a proto jsem od tohoto směru upustil.

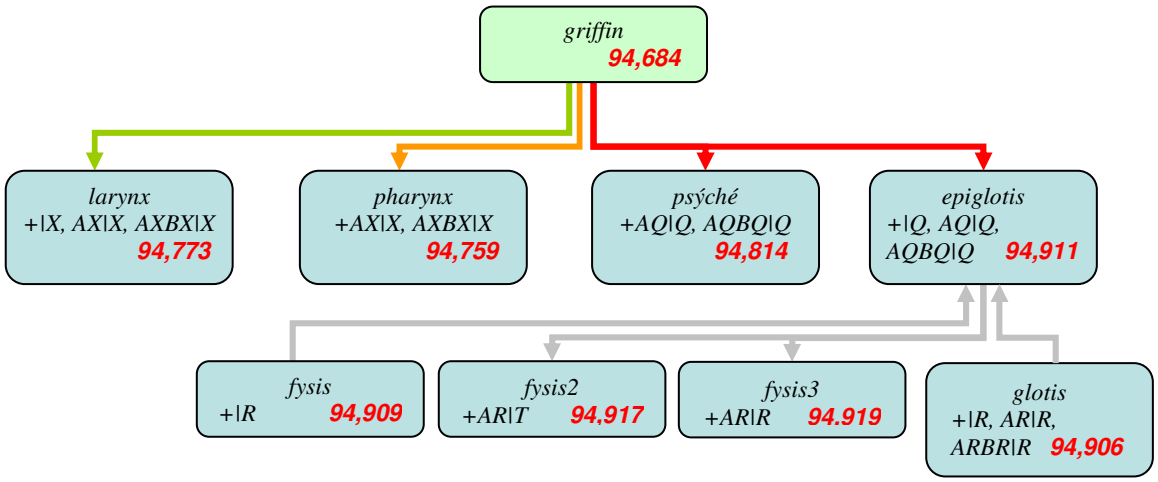
Při vytváření verzí jsem se dopustil jedné chyby, jejíž výsledek však pro zajímavost také uvádím. Omylem jsem vytvořil dvě verze, které spojovaly dvojici rod – pád (*larynx*, *pharynx*). Jak se ukázalo, i tato dvojice má svůj význam, neboť její použití vedlo ke zlepšení. Protože však dvojice poddruh – pád vykázala výraznější zlepšení, na dvojici rod – pád jsem v dalších experimentech nenavázal.

verze	**rysy**	**počet**	**úspěšnost**
griffin	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, UT, UL, AL, aW	14	94,684
larynx	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, UT, UL, AL, aW, |X, AX|X, AXBX|X	17	94,773
pharynx	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, UT, UL, AL, aW, AX|X, AXBX|X	16	94,759
psýché	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, UT, UL, AL, aW, AQ|Q, AQBQ|Q	16	94,814
epiglotis	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, UT, UL, AL, aW, |Q, AQ|Q, AQBQ|Q	17	94,911
fysis	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, UT, UL, AL, aW, |Q, AQ|Q, AQBQ|Q, |R	18	94,909
fysis2	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, UT, UL, AL, aW, |Q, AQ|Q, AQBQ|Q, AR|T	18	94,917
fysis3	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, UT, UL, AL, aW, |Q, AQ|Q, AQBQ|Q, AR|R	18	94,919
glotis	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, UT, UL, AL, aW, |Q, AQ|Q, AQBQ|Q, |R, AR|R, ARBR|R	20	94,906

Tabulka 36. Přehled verzí využívajících částečné n-gramy

verze 1	verze 2	t	úroveň významnosti
griffin	*larynx*	4,159	0,02
	pharynx	5,541	0,01
	psýché	10,841	0,001
	epiglotis	13,112	0,001
epiglotis	*fysis*	-0,161	-
	fysis2	0,356	-
	fysis3	0,615	-
	glotis	-0,747	-

Tabulka 37. Srovnání verzí využívajících částečné n-gramy



Graf 17. Uspořádání verzí využívajících částečné n-gramy

8.3.13. Rozbor verze *lucifer4*

Na původních datech byla nejúspěšnější verze *lucifer4*. Protože vývoj verzí na nových datech probíhal neustále a nebylo tedy známo, která verze, či dokonce která vývojová větev povede k úspěchu, podrobil jsem analýze právě verzi *lucifer4*, abych prozkoumal význam jednotlivých rysů. Zároveň jsem chtěl ověřit, že během vývoje této verze nedošlo k tomu, že se některý rys stal nadbytečným, a zda by jeho odebráním nedošlo ke zlepšení úspěšnosti. Otestoval jsem tedy verzi *lucifer4* postupně s jedním rysem vynechaným. Opět jsem použil pětinásobnou crossvalidaci na nových datech a kromě úspěšnosti jsem také určil statistickou významnost výsledku pomocí t-testu.

verze	odebraný rys	úspěšnost	t (srovnání s *lucifer4*)	úroveň významnosti
lucifer4		94,657		
lucifer4_m1	|T	94,641	-1,693	-
lucifer4_m2	AT	94,189	-35,351	0,001
lucifer4_m3	ATBT	94,606	-2,673	-
lucifer4_m4	BT	94,549	-9,447	0,001
lucifer4_m5	|W	94,219	-23,997	0,001
lucifer4_m6	AW|W	94,607	-13,435	0,001
lucifer4_m7	AW	94,623	-2,562	-
lucifer4_m8	BW	94,653	-0,268	-
lucifer4_m9	|O	94,615	-4,124	0,02
lucifer4_m10	AC	94,605	-3,267	0,05
lucifer4_m11	|U	94,622	-4,053	0,02
lucifer4_m12	|V	94,618	-6,541	0,01
lucifer4_m13	VT	94,628	-2,234	-
lucifer4_m14	VL	94,668	1,235	-
lucifer4_m15	VL|C	94,635	-1,593	-

Tabulka 38. Přehled verzí vzniklých z lucifer4 odebráním rysu

Z tabulky je patrné, že odebrání jednoho rysu neohrozí úspěšnost verze zásadním způsobem. Algoritmus je zřejmě schopen význam jednoho rysu po odebrání přesunout na rysy ostatní a vzniklou chybu tím značně snížit.

Výraznější zhoršení (přes 0,4%), a tudíž zřejmě největší význam vykazují rysy |W (aktuální slovní forma) a AT (předchozí značka). Stejně jako v experimentu s automatickým skládáním rysů[1] se potvrdil stěžejní význam značkového bigramu (AT). Statisticky významné zhoršení přineslo také vynechání značky dvě pozice zpět (BT), slovního bigramu (AW|W), pořadí slova ve větě (|O), předchozí pád (AC) a velikost písmen slova a lemmatu (|V, |U).

Odebrání značkového trigramu (ATBT) či samostatné aktuální značky (|T) překvapivě ke statisticky významnému zhoršení nevedlo. Ukázal se zcela minimální význam slovní formy dvě pozice zpět, což částečně potvrzují i výsledky při ručním vývoji verzí.

V jednom případě dokonce došlo ke zlepšení – po odebrání lemmatu nejbližšího slovesa vpravo (VL). Toto zlepšení však nebylo statisticky významné, takže mu nelze přikládat velký význam. Odebrání VL však bylo později ověřeno i na nejlepší verzi na nových datech – *hepar*. Tam se však projevilo výrazným zhoršením.

Celkově se však neukázalo, že by nějaký rys významně zhoršoval výsledek, takže rozumný výběr rysů ve verzi *lucifer4* se potvrdil.

[1] viz 7.6.1.

8.3.14. Vývojová cesta k nejlepší verzi *hepar*

Předchozí kapitoly byly řazeny podle jednotlivých typů rysů, se kterými se pracovalo. Z tohoto členění však není patrná žádná vývojová cesta k nejlepší verzi *hepar*. Následující tabulka tedy uvádí hlavní větev vývoje od počáteční verze *jennefer* až po závěrečnou verzi *hepar*. Verze jsou řazeny v pořadí tak, že na následujícím řádku je vždy uveden bezprostřední následník řádku aktuálního. Zároveň tabulka ukazuje hodnotu *t* pro t-test a úroveň významnosti při srovnání vždy s předcházející verzí. Jak je vidět, každý krok této větve vedl ke statisticky významnému zlepšení, což odpovídá předpokladu, který jsem si stanovil na začátku.

verze	**rysy**	**počet**	**úspěšnost**	**t**	**úroveň významnosti**
jennefer	|T, AT, ATBT	3	92,630		
gerald	|T, AT, ATBT, |W	4	93,656	19,237	0,001
šemík2	|T, AT, ATBT, |W, AW	5	93,805	17,714	0,001
šemík	|T, AT, ATBT, |W, AW, BW	6	93,881	3,602	0,05
hatatitla	|T, AT, ATBT, |W, AW, BW, AW|W	7	93,929	3,248	0,05
kryštůfek	|T, AT, ATBT, BT, |W, AW, BW, AW|W	8	94,118	42,801	0,001
robin	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O(9)	9	94,181	4,660	0,01
cvach2	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VL(20)	10	94,449	10,654	0,001
blažej2	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VL, VT(20)	11	94,520	18,696	0,001
kostěj2b	|T, AT, ATBT, BT, |W, AW, BW, AW|W, |O, VL, VT, UL(10), UT(10) *do UT a UL aktuální slovo nezahrnuto*	13	94,595	9,563	0,001
trifid	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, UT, UL, AL	13	94,641	5,675	0,01
griffin	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, UT, UL, AL, aW	14	94,684	3,282	0,05
epiglotis	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, UT, UL, AL, aW, |Q, AQ|Q, AQBQ|Q	17	94,911	13,112	0,001
hepar	|T, AT, ATBT, |W, AW, BW, BT, |O, VT, VL, UT, UL, AL, aW, |Q, AQ|Q, AQBQ|Q, |U(0,1), |V(0,1,2)	19	94,944	3,971	0,02

Tabulka 39. Přehled verzí vývoje nejúspěšnější verze hepar

9. Shrnutí

V této kapitole shrnu jednak vypozorované obecné vlastnosti algoritmu, jednak dosažené výsledky ve značkování. V závěru kapitoly pak uvedu další možné směry pro pokračování.

9.1. Obecné vlastnosti algoritmu

Pozorování učiněná již během práce na projektu s původními daty byla potvrzena a prohloubena. K oprávněnosti následujících tvrzení přispěly i výsledky automatického vývoje verzí.

- **Algoritmus „nesnáší“ zahlcení informacemi.**

Celý projekt byl sice implementován tak, aby bylo možno použít velké počty rysů (řádově miliony, tj. stovky na jeden Viterbiho přechod mezi stavy), ale ukázalo se, že mnohem lepší výsledky dává poměrně malá sada rysů, ovšem pečlivě zvolená. Jedná se o necelých 20 rysů na jeden přechod, celkově stovky tisíc použitých rysů.

Nestačí tedy rysy pouze mechanicky přidávat a spoléhat, že si trénovací algoritmus sám zvolí ty správné rysy přiřazením vyšších vah. Různé sady rysů a vztahy mezi nimi je třeba podrobně zkoumat. Díky této vlastnosti však práce s tímto algoritmem přinesla lingvisticky zajímavé informace o tom, co je a co není pro daný úkol podstatné.

- **Algoritmus „nesnáší“ komplikované rysy.**

Přestože komplikované rysy obsahují více informací, a jsou tudíž specializovanější (přesněji popisují daný kontext), ukázalo se jako perspektivní používání spíše elementárních rysů a jejich kombinací nejvýše do počtu tři. To je patrně dáno tím, že komplikované rysy příliš dobře popisují kontext na trénovacích datech, takže jim jsou přiřazeny velké váhy. Váhy jednodušších rysů jsou pak druhotné. Při použití na datech testovacích se však zpravidla komplikované rysy neuplatní, neboť speciální kontext, který popisují, se nevyskytuje příliš frekventovaně. Pak se tedy uplatní pouze jednodušší, ale nepříliš dobře natrénované jednoduché rysy.

Používání komplikovaných rysů tedy může vést k výraznému zhoršení. Pozorování při ručním i automatickém vývoji tuto úvahu jednoznačně potvrzují.

- **Rysy si navzájem mohou „nezdravě“ konkurovat.**

Přidáním jednoho rysu může dojít i ke zhoršení výsledku. Sloučením dvou relativně úspěšných, leč typově rozdílných verzí může také dojít ke zhoršení. U některých rysů, které v ranějších verzích vedly ke zlepšení, může v pozdějších verzích vést ke zlepšení jejich vypuštění. Z těchto důvodů je třeba pro každý přidávaný rys zkoumat jeho vliv na vybraných jednoduchých i komplikovanějších verzích a také pečlivě zkoumat slučitelnost jednotlivých rysů.

9.2. Výsledky a srovnání nejlepší verze *hepar*

Před zahájením vývoje verzí na nových datech jsem změřil úspěšnost do té doby nejlepší verze *lucifer4*. Přetrénoval jsem tuto verzi na nových trénovacích datech a otestoval ji na nových evaluačních datech. Tím jsem získal hodnotu, se kterou je možno porovnávat úspěšnost nejlepší verze po skončení práce. Bohužel doposud nebyly k dispozici žádné jiné výsledky na nových datech.

Tabulka shrnuje dosažené výsledky na obou verzích. Verze *lucifer4* byla testována na 7. iteraci, verze *hepar* na 9. iteraci.

verze	**minimální počet výskytů rysu pro filtrování**	**úspěšnost na testovacích datech**	**úspěšnost při pětinásobné crossvalidaci**	**výsledná úspěšnost na evaluačních datech**
lucifer4	3	94,811	94,657	**94,474**
hepar	3	95,035	94,944	**94,702**
rozdíl		0,224	0,287	**0,228**

Tabulka 40. Srovnání úspěšností verzí lucifer4 a hepar

Jak je z tabulky patrné, dosáhl jsem během práce zlepšení o 0,228%, což je v oblasti 94% úspěch. Na druhou stranu je nutno přiznat, že k tomu, aby se Morče stalo nejlepším taggerem pro češtinu to patrně nebude postačující. To však nebylo podmínkou této práce.

Následující tabulka ještě shrnuje úspěšnosti verze *hepar* na jednotlivých pozicích značky (opět na evaluačních datech).

Úspěšnost ve volbě správné značky i lemmatu	93,820
Úspěšnost ve volbě značky	94,702
Úspěšnost ve volbě lemmatu	97,635
Úspěšnost ve volbě druhu (POS)	98,553
Úspěšnost ve volbě poddruhu (SUBPOS)	98,482
Úspěšnost ve volbě rodu (GENDER)	97,634
Úspěšnost ve volbě čísla (NUMBER)	97,913
Úspěšnost ve volbě pádu (CASE)	96,274
Úspěšnost ve volbě rodu vlastníka (POSGENDER)	99,971
Úspěšnost ve volbě čísla vlastníka (POSNUMBER)	99,988
Úspěšnost ve volbě osoby (PERSON)	99,911
Úspěšnost ve volbě času (TENSE)	99,940
Úspěšnost ve volbě stupně (GRADE)	99,669
Úspěšnost ve volbě negace (NEGATION)	98,890
Úspěšnost ve volbě slovesného rodu (VOICE)	99,940
Úspěšnost ve volbě varianty (VAR)	99,470

Tabulka 41. Shrnutí úspěšností verze hepar pro jednotlivé pozice značky

9.3. Možnosti dalšího vývoje

Výsledky této práce jsou omezeny časem, který je potřeba pro provádění experimentů. Bylo otestováno pouze několik typů rysů a domnívám se, že možnosti ručního vývoje verzí ještě vyčerpány nebyly. Na druhou stranu s rostoucí úspěšností roste i potřebné úsilí pro jakékoliv zlepšení.

Zcela jistě by bylo potřeba zevrubněji prozkoumat rysy, které předpovídají značky pouze částečně. To však obnáší opravdu velké množství verzí k prověření. K tomu by bylo možno využít automatický vývoj verzí, ovšem za předpokladu dostatečně dlouhého času pro výpočty (řádově týdny až měsíce).

Pro další výraznější zlepšování by pak mohlo být prospěšné využít další možné informace o textu – například označkování jinými taggery a podobně.

9.4. Závěr

Cíl této práce – stanovit metodu pro vývoj verzí a důkladně zmapovat co největší počet verzí – byl v rámci daných možností splněn. Popis vývoje verzí může sloužit jako vodítko pro další experimenty s jinými taggery. Navíc se podařilo úspěšnost taggeru Morče statisticky významně zvýšit. Výsledkem práce je také doplnění programu o nástroje pro automatický vývoj verzí.

Použitá literatura

[1] Collins M. (2002): *Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms.* EMNLP.

[2] Jelinek, F. (1998): *Statistical Methods for Speech Recognition*. The MIT Press.

[3] Lowry R. (1999-2005): *Concepts and Applications of Inferential Statistics.* http://faculty.vassar.edu/lowry/webtext.html

Další zdroje

[4] *Dokumentace projektu Morče* (připojena na přiloženém CD).

[5] *Popis značkovacího systému češtiny.* http://quest.ms.mff.cuni.cz/pdt/Morphology_and_Tagging/Doc/docc0pos.pdf

[6] *Specifikace standardu CSTS.* http://quest.ms.mff.cuni.cz/pdt/Corpora/PDT_1.0/Doc/csts/ALL-ELEM.html

Příloha A – obsah připojeného CD

Součástí této práce je i přiložené CD obsahující zdrojové kódy, skripty a data náležící k projektu, a dále kompletní souhrnné výsledky.

CD obsahuje tyto části:

- adresář *morce*, ve kterém se nachází kompletní projektový balík morce.tar.gz a původní uživatelská dokumentace projektu, podle níž je možné projekt nainstalovat
- adresář *dipl*, ve kterém se nachází tato práce ve formátu pdf
- adresář *vysledky*, ve kterém se nachází tyto soubory:

- vysledky.xls: souhrn výsledků všech verzí na druhém listu, na prvním listu je pak možno po zadání názvů verzí vidět výpočet t-testu
- komplet.html: kompletní výsledky všech verzí a všech iterací
- kombinovani.txt: souhrn výsledků automatického kombinování rysů
- skladani.txt: souhrn výsledků automatického skládání rysů

Příloha B – Změny a doplňky kódu

Instalace projektu zůstala ve své základní podobě nezměněna, byly v ní však provedeny následující změny:

- parser upraven tak, aby do interních struktur načítal i předchozí značku a lemma vybranou Morčetem
- přidány programy master, master2 a slave, které slouží k automatickému vývoji rysů. Jejich použití je přesně popsáno v kapitole 7.
- odpovídajícím způsobem byl upraven i hlavní Makefile
- v adresáři *morce/versions* se nacházejí generátory všech použitých verzí
- programy make_ftrs, train a test obsahují navíc volitelný parametr –k, kterým je možno definovat inicializační soubor verze při automatickém testování
- přidány skripty pro kompletní natrénování, otestování a vyhodnocení verze při pětinásobné crossvalidaci, jsou však vytvořeny přímo na míru stávajících dat. Jedná se o interaktivní skript xtrain, který využívá ke svému běhu pomocný skript do_xtrain, do_xtest a do_xeval.
- přidán skript do_atrain, sloužící pro automatický vývoj. Jeho použití je popsáno v kapitole 7.
- v adresáři *morce/data* jsou uložena nová data z PDT 2.0 (trénovací c4049a-t.cst, testovací c4049a-d.cst a evaluační c4049a-e.cst)
- v datovém repozitáři je uložen spojený trénovací a testovací soubor pod názvem ts2, pod stejným názvem je veden i jako testovací (použitá část pro crossvalidaci se rozliší odpovídajícími parametry)
- v datovém repozitáři je uložen nový trénovací soubor pod názvem t2, testovací pod názvem s2 – pro ně je v repozitáři uložena natrénovaná a otestovaná verze hepar