Automatické rozpoznávání předpon a přípon s pomocí nástroje Affisix*

Jaroslava Hlaváčová, Michal Hrušecký

Matematicko-fyzikální fakulta Univerzity Karlovy v Praze, Česká Republika,
Hlavacova@ufal.mff.cuni.cz, Michal@Hrusecky.net

Abstrakt *Článek se zabývá problematikou segmentace slov. Ukáže, k čemu lze znalosti o segmentaci slov použít, a popíše několik metod pro automatickou segmentaci slov. Závěrem představí open source nástroj Affisix implementující představené metody.*

1 Motivace

Mnoho moderních metod pro zpracování přirozeného jazyka je založeno na statistickém přístupu, kdy natrénujeme jazykový model na trénovacích datech před jeho použitím. V reálných úlohách ale nelze předpokládat, že se v trénovacích datech objevila všechna možná slova. Zde nám znalost segmentace slov může pomoci. V případě, že při zpracování reálných dat narazíme na neznámé slovo, můžeme ho zkusit rozložit a získat tak alespoň nějaké informace.

Příklad 1 *Konstruujeme automatický tagger češtiny. V reálných datech se objevilo slovo* `rozpoznal`*, které se ovšem nevyskytovalo v trénovacích datech. Systém ale může zjistit, že segmentace tohoto slova může být* `roz-poznal`*. Může tedy zkusit ve slovníku nalézt slovo* `poznal`*. Budeme-li předpokládat, že předpony nemění mluvnické kategorie, které se snažíme rozpoznat, můžeme použít tag od slova* `poznal`*.*

Příklad 2 *Konstruujeme systém pro automatické rozpoznávání mluvené řeči a používáme jazykový model pro zpřesnění výstupu. Z rozpoznávače získáme, že uživatel pravděpodobně řekl* `eurostava`*. Druhé nejpravděpodobnější slovo bylo* `euroústava`*. Ani jedno z nich ovšem nebylo v trénovacích datech. Pokud ale víme, že* `euro-` *se chová jako předpona, a v trénovacích datech bylo slovo* `ústava`*, můžeme rozhodnout, že výstup* `euroústava` *je lepší.*

Příklad 3 *Konstruujeme sázecí program a chceme, aby uměl automaticky dělit slova. Narazíme-li na neznámé slovo, můžeme se pokusit ze znalostí o segmentaci nalézt vhodné dělení a navrhnout ho uživateli.*

* Tento článek prezentuje výsledky výzkumu podporovaného granty GAČR P202/10/1333, GAČR P406/2010/0875 a Centrem komputační lingvistiky, MŠMT ČR LC536.

V příkladech šlo vždy o lingvistické morfémy (příklad 1 a 2 se týkal předpon, 3. příklad morfémů obecně), při automatickém zpracování jazyka však nemusí být lingvistická motivace dělení slov nutná. Proto pro další použití budeme definovat předpony a přípony bez ohledu na lingvistické vlastnosti.

Definice 1 *Posloupnost písmen je* ***předpona****, pokud*

- *se vyskytuje na začátku mnoha slov*
- *ve slovech, kde se vyskytuje, může být nahrazena jinou posloupností písmen tak, že výsledek také tvoří slovo*

Ilustrace definice na příkladě předpony *euro-* (všechny příklady jsou z korpusu SYN [4]):

První podmínku dokumentují např. následující úryvky:

- *Třináct kritiků evropské integrace sepsalo "**kontra**ústavu" Evropské unie*
- *Hongkongská správa chce dál prosazovat novou **mini**ústavu*

Druhá podmínka je splněna existencí slov:

- *euro**politika**, euro**trh**, euro**zastoupení**, euro**expert***

Obdobně, jako jsme definovali předpony, budeme definovat i přípony:

Definice 2 *Posloupnost písmen je* ***přípona****, pokud*

- *se vyskytuje na konci mnoha slova*
- *ve slovech, kde se vyskytuje, může být nahrazena jinou posloupností písmen tak, že výsledek také tvoří slovo*

Ve výše uvedených definicích chybí určení toho, kolik je *mnoho*. To ovšem nelze obecně stanovit. Můžeme ale říci, jak moc která část slova tuto definici odráží. To se pokouší postihnout níže zmiňované metody.

2 Metody

V této části se budeme zabývat několika metodami pro automatickou segmentaci. Všechny popsané metody jsou jazykově nezávislé. Jediný požadavek na jazyk je, aby se text v psané formě dělil na slova, a předpony či přípony se přidávaly pouze na začátek či konec slova. Algoritmy pro svůj běh vyžadují pouze obsáhlý seznam slov z daného jazyka. Kromě metody diferenční entropie jsou metody převzaty z článku [1] o vytváření seznamu předpon a přípon ve španělštině pomocí korpusu.

V následujícím textu bude Ω značit množinu všech slov vyskytujících se v jazyce. Nechť slovo α je složeno z posloupnosti písmen $a_1, \ldots, a_n$. Potom budeme psát, že $\alpha = a_1 :: \ldots :: a_n$. Posloupnost písmen nazýváme také **řetězec**.

Operátor "::" budeme obdobným způsobem definovat i pro řetězce. Nechť slovo α je složeno z posloupnosti písmen $a_1, \ldots, a_n$ a nechť

$$p = a_1 :: \ldots a_l \quad \wedge \quad s = a_l :: \ldots :: a_n$$

Potom budeme psát, že $\alpha = p :: s$.

Definice 3 ***Dělením slova*** *budeme nazývat operaci, kdy ze slova w získáme dvě posloupnosti znaků p a s takové, že $w = p :: s$.*

Jelikož naším cílem je nalézt nejvhodnější dělení, všechny následující metody budou založeny na principu ohodnocování dělení. Pro každé možné dělení slova nám každá metoda vrátí číslo odpovídající vhodnosti tohoto dělení. Na nás poté je stanovit mez, při které dělení označíme za správné.

2.1 Metoda čtverců

Metoda čtverců je nejjednodušší metodou zde popisovanou. Využívá přímo definice předpon a přípon, uvedené v sekci 1.

Definice 4 ***Čtverec*** *je čtveřice řetězců (řetězec může být i prázdný) $< p_1, p_2, s_1, s_2 >$ taková, že platí všechny následující podmínky:*

- $p_1 :: s_1 \in \Omega$
- $p_1 :: s_2 \in \Omega$
- $p_2 :: s_2 \in \Omega$
- $p_2 :: s_1 \in \Omega$

Požadavkem na vytvoření čtverce tedy je, že každý počáteční segment musí s každým koncovým segmentem dohromady vytvářet slovo.

Definice 5 *Mějme slovo w a jeho dělení $w = p :: s$.* ***Počet dopředných čtverců*** *$S_f(p)$ je potom* $S_f(p) = |\{< p, p_2, s_1, s_2 >; < p, p_2, s_1, s_2 > \textit{je čtverec}\}|$.

Počet dopředných čtverců tedy říká, v kolika čtvercích se daná posloupnost písmen vyskytuje jako jeden z počátečních segmentů.

Obdobně budeme definovat počet zpětných čtverců jako počet čtverců, ve kterých se daná posloupnost písmen vyskytuje jako jeden z koncových segmentů.

Definice 6 *Mějme slovo w a jeho dělení $w = p :: s$.* ***Počet zpětných čtverců*** *$S_b(p)$ je potom* $S_b(p) = |\{< p_1, p_2, s, s_2 >; < p_1, p_2, s, s_2 > \textit{je čtverec}\}|$.

Vezmeme-li nyní v úvahu původní definice předpon a přípon, je patrné jak odpovídá metoda čtverců této definici. Počet dopředných čtverců nám řekne, v kolika slovech se posloupnost písmen vyskytovala na začátku a zároveň započítá jen slova, kde ji bylo možné nahradit něčím jiným.

Obdobným způsobem lze použít počet zpětných čtverců pro rozpoznávání přípon.

2.2 Entropie

Metoda entropie je založena na odlišném přístupu. Není zde již tolik patrná souvislost s definicemi v úvodu. Je založena na *entropii*, která je definována následujícím způsobem:

$$H(p) = -\sum_{s_i \in S} p(s_i|p) \log_2 p(s_i|p)$$

kde S je množina jevů, které mohou nastat po jevu p.

Entropie vyjadřuje míru nejistoty. Jestliže slovo začíná řetězcem p, potom čím je $H(p)$ vyšší, tím je těžší předpovědět, jakým řetězcem s_i bude slovo pokračovat. Je zřejmé, že za předponami bude entropie vysoká, protože předpony stojí na začátku mnoha slov. Podobně bude entropie vysoká před příponami, neboť k danému začátku slova lze obvykle připojit více přípon. Pro naše potřeby budeme počítat entropii dvěma způsoby.

Definice 7 ***Dopředná entropie*** *posloupnosti písmen r je definována jako*

$$H_f(r) = -\sum_{s_i; r::s_i \in \Omega} p_f(s_i|r) \log_2 p_f(s_i|r)$$

kde $p_f(s_i|r)$ označuje pravděpodobnost, že slovo začínající r bude končit s_i.

Definice 8 ***Zpětná entropie*** *posloupnosti písmen r je definována jako*

$$H_b(r) = -\sum_{s_i; s_i::r \in \Omega} p_b(s_i|r) \log_2 p_b(s_i|r)$$

kde $p_b(s_i|r)$ označuje pravděpodobnost, že slovo končící r bude začínat s_i.

V definici předpony jsme požadovali, aby řetězec mohl pokračovat mnoha různými způsoby. Pokud máme mnoho různých konců, je nejistota, co bude následovat po předponě, výrazně vyšší, než pokud máme jen jeden možný konec. Čím vyšší má tedy úvodní řetězec dopřednou entropii, tím více odpovídá naší definici předpony. Dopředná entropie ovšem nijak explicitně nezahrnuje druhou podmínku — aby předponu bylo možno odtrhnout a nahradit jinou tak, aby výsledek byl také slovem v daném jazyce.

O tom by nám něco mohla prozradit zpětná entropie, ale ta bývá na začátku slov poměrně nízká. Pro splnění této podmínky můžeme použít i jiné metody (například zkusit najít alespoň jeden čtverec), nebo jen spoléhat, že když má počáteční řetězec tolik možných pokračování, tak že ho lze i nahradit.

Pro odhalování předpon se tedy hodí dopředná entropie, kterou je možné ještě trochu vylepšit pomocí dodatečných podmínek. Definice přípony se od předpony liší jen odtrháváním z druhého konce slova. A jelikož se zpětná entropie od dopředné liší taktéž pouze opačným směrem zpracování slova, lze stejným způsobem, jakým lze použít dopřednou entropii pro rozpoznávání předpon, použít zpětnou entropii pro rozpoznávání přípon.

2.3 Diferenční entropie

Podíváme-li se na výsledky *metody entropie*, je patrné, že u dopředné entropie jsou hodnoty na začátku slova výrazně vyšší než na jeho konci. Aby bylo možné rozumně porovnávat vhodnost dělení slova na začátku i na konci, je třeba použít nějakou formu automatické normalizace.

Tu nám nabízí metoda *diferenční entropie*.

Definice 9 ***Dopředná diferenční entropie*** *posloupnosti písmen $r = r_r :: r_1$ je definována jako*

$$H_{df}(r) = H_f(r) - H_f(r_r)$$

kde r_1 je právě jedno písmeno.

Definice 10 ***Zpětná diferenční entropie*** *posloupnosti písmen $r = r_1 :: r_r$ je definována jako*

$$H_{db}(r) = H_f(r) - H_f(r_r)$$

kde r_1 je právě jedno písmeno.

Neměříme tedy už nejistotu, s jakou lze předpovědět, co bude pokračovat, ale nárůst této nejistoty. Budeme-li postupovat od začátku slova k jeho konci, bude dopředná entropie obecně klesat, neboť čím více se při zpracování blížíme ke konci slova, tím méně možností pokračování můžeme očekávat. Dopředná diferenční entropie zachytí okamžiky, kdy entropie oproti očekávání vzroste, a to i tehdy, když její hodnota není absolutně nejvyšší. To nám umožní zachytit ta místa ve slově, kde je nejistota pokračování neobvykle vysoká. Opět se tedy snažíme o splnění první podmínky z definice, kdy požadujeme, aby řetězec (předpona) měl mnoho možností pokračování. V tomto případě "mnoho" není globální limit, ale znamená více možností, než je obvyklé.

Druhá podmínka, totiž možnost odtrhnout předponu a nahradit ji jinou, není ani u této metody zohledněna, stejně jako u metod entropie. Lze se s ní vypořádat i stejným způsobem. Podobně jako u metod entropie se dopředná diferenční entropie hodí pro rozpoznávání předpon a zpětná diferenční entropie k rozpoznávání přípon.

3 Affisix

`Affisix` je open source nástroj pro experimenty se segmentací slov. Je psán v jazyce C++ a distribuován pod licencí GPLv3. Obsahuje pouze rozhraní pro příkazovou řádku. Díky tomu je ale snadno přenositelný a umožňuje snadné začlenění do složitějších scriptů pro zpracování dat.

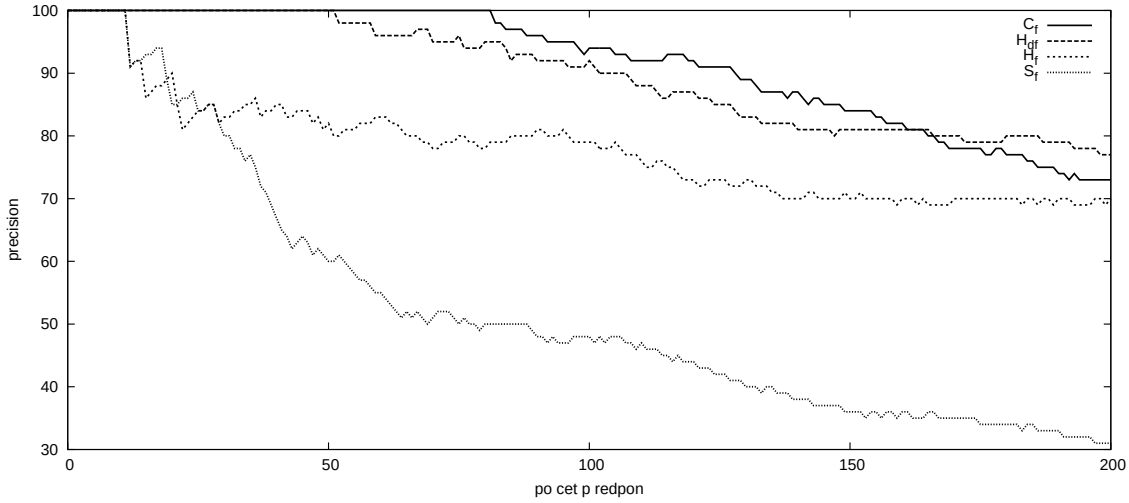
Hlavní výhodou programu `Affisix` je jeho univerzálnost. Implementuje nejen uvedené metody, ale zároveň i jednoduchý interpreter. Díky němu lze metody a jednoduché pomocné funkce libovolně kombinovat. Uživatel tak může snadno testovat různé možnosti kombinace metod bez nutnosti rekompilace.

`Affisix` podporuje dva různé módy. První mód je *collector*, kdy jsou vstupní data použita pouze pro vygenerování seznamu rozpoznaných segmentů. Druhý mód je *filter*. V tomto módu `Affisix` projde vstupní text a pouze označí místa vhodná pro segmentaci.

4 Výsledky

Pro srovnání metod jsme provedli několik experimentů. Ty spočívaly v použití Affisixu pro rozpoznání předpon existujících v češtině.

Jako vstupní byla použita filtrovaná lemmata z korpusu SYN2000 [4]. Odstranili jsme všechna slova začínající velkým písmenem (typicky vlastní jména), dále slova obsahující tři stejná písmena za sebou (různé chyby, které se dostaly do korpusu) a nakonec i slova obsahující pro češtinu netypická písmena "q", "w" a



Obrázek 1. Srovnání precision při rozpoznávání předpon

"x". Každé ze zbylých lemmat bylo započítáno pouze jednou.

Na vstupní data jsme použili zmiňované metody v kombinaci s požadavkem, že předpona netvoří více než polovinu slova, a získali tak kandidáty na české předpony. Výsledné seznamy byly zkontrolovány člověkem, který stanovil, které řetězce jsou skutečnými předponami a které ne.

Pro porovnávání metod jsme použili *precision*.

Definice 11 *Nechť Γ je množina předpon získaných automatickou metodou a nechť Ψ je množina skutečných předpon existujících v daném jazyce. Potom* ***precision*** *P je definována jako*

$$P = \frac{|\Gamma \cap \Psi|}{|\Gamma|} \times 100\%$$

Dále jsme používali pomocnou metodu C_f utvořenou jako součet normalizovaných hodnot jednotlivých metod:

Definice 12 *Nechť p je řetězec a nechť Υ je množina všech řetězců. Dále nechť:*

$$||H_f(p)|| = H_f(p) / \max_{s \in \Upsilon}\{H_f(s)\}$$
$$||H_{df}(p)|| = H_{df}(p) / \max_{s \in \Upsilon}\{H_{df}(s)\}$$
$$||S_f(p)|| = \log(S_f(p)) / \max_{s \in \Upsilon}\{log(S_f(s))\}$$

Potom:

$$C_f(p) = ||H_f(p)|| + ||H_{df}(p)|| + ||S_f(p)||$$

V tabulce 1 je vidět 25 nejlepších rozpoznaných předpon, seřazených podle hodnoty C_f.

	předpona	C_f	$\|\|H_f\|\|$	$\|\|H_{df}\|\|$	$\|\|S_f\|\|$
1.	super	2.682	0.967	0.978	0.737
2.	pseudo	2.602	0.963	0.961	0.677
3.	mikro	2.532	0.921	0.917	0.693
4.	sebe	2.505	0.898	0.849	0.757
5.	rádoby	2.480	0.983	1.000	0.496
6.	deseti	2.437	0.917	0.865	0.653
7.	mimo	2.423	0.968	0.801	0.653
8.	hyper	2.420	0.906	0.894	0.619
9.	anti	2.393	0.942	0.706	0.744
10.	roz	2.390	0.922	0.530	0.937
11.	severo	2.384	0.966	0.839	0.578
12.	jiho	2.370	0.944	0.848	0.577
13.	makro	2.366	0.935	0.886	0.544
14.	elektro	2.364	0.931	0.797	0.635
15.	jedno	2.361	0.923	0.726	0.711
16.	nízko	2.359	0.940	0.919	0.499
17.	mega	2.341	0.904	0.861	0.575
18.	vnitro	2.332	0.939	0.862	0.530
19.	spolu	2.327	0.921	0.711	0.695
20.	dvou	2.317	0.929	0.659	0.728
21.	euro	2.311	0.918	0.791	0.601
22.	velko	2.309	0.857	0.820	0.631
23.	auto	2.306	0.896	0.737	0.673
24.	ultra	2.306	0.873	0.888	0.544
25.	devíti	2.304	0.858	0.833	0.612

Tabulka 1. Ukázka výsledků

V tabulce 2 je k dispozici srovnání jednotlivých metod podle precision. Precision byla počítána pro 10, 50, 100, 150 a 200 nejlepších předpon dle dané metody. Jak precision postupně klesá přidáváním dalších kandidátů na předpony, je vidět na obrázku 1.

metoda	10	50	100	150	200
C_f	100%	100%	94%	84%	73%
H_{df}	100%	100%	92%	81%	77%
H_f	100%	82%	79%	70%	70%
S_f	100%	60%	48%	36%	31%

Tabulka 2. Srovnání precision při rozpoznávání předpon

Aktuální výsledky [2] ukazují, že pomocí kombinace metod lze dosáhnout dobrých výsledků při pořizování seznamu předpon. Na druhou stranu praktické využití výsledků při automatickém překladu [3] zatím nepřineslo očekávané výsledky. V té době ale nebyl ještě k dispozici mód *filter*, a program měl tedy k dispozici méně informací, než by měl dnes.

Cílem do budoucna je přidávat další slibné metody a rozšířit možnosti jejich kombinací.

Reference

1. Urrea, A. M.: Automatic Discovery of Affixes by means of a Corpus: A Catalog of Spanish Affixes. Journal of Quantitative Linguistics **7** (2000) 97–114
2. Hlaváčová, J., Hrušecký, M.: Affisix: Tool for Prefix Recognition. Text, Speech and Dialogue **5246** (2008) 85–92
3. Bojar, O., Straňák, P., Zeman, D., Jain, G., Hrušecký, M., Richter, M., Hajič, J.: English-Hindi Translation—Obtaining Mediocre Results with Bad Data and Fancy Models. Proceedings of the 7th International Conference On Natural Language Processing (ICON-2009)
4. Ústav Českého národního korpusu FF UK: Český národní korpus - SYN. http://ucnk.ff.cuni.cz
5. Hrušecký, M.: Affisix. http://affisix.sf.net