Word Sense Disambiguation

Matouš Macháček

Únor 2013

1 Popis úlohy

Rozpoznání významu slova je klasická úloha počítačové lingvistiky. Cílem této úlohy je pro dané víceznačné slovo vybrat správný význam, který je dán kontextem daného slova.

V této úloze máme pro každé ze slov *line*, *serve* a *hard* vyvinout automatický klasifikátor, který pro dané víceznačné slovo použité ve větě určí jeho význam.

2 Data

Pro každé víceznačné slovo, pro které budeme vyvýjet klasifikátor, máme k dispozici trénovací data. Tato data obsahují instance použití daného slova. Každá taková instance obsahuje tokenizovanou větu, morfologickou analýzu věty (včetně lemat), výstup ze Stanfordského závislostního parseru a správný význam slova. V tabulce 1 můžete vidět počty instancí pro jednotlivá slova.

Testovací data určená pro měření finální úspěšnosti nemáme zatím k dispozici. Uváděné výsledky budou proto měřeny na vyčleněné vývojové testovací sadě, nebo pomocí křížové validace.

3 Použité metody

Pří vývoji klasifikátorů jsem použil tři různe algoritmy strojového učení a další metody, které v této sekci teoreticky popíšu.

Cílové slovo	**Počet instancí**
hard	3834
line	3647
serve	3879

Tabulka 1: Počet trénovacích instancí

3.1 Naive Bayes

Metoda Naive Bayes předpokládá, že hodnoty jednotlivých rysů jsou na sobě nezávislé. Klasifikátor vybere třídu, která má nejvyšší pravděpodobnost:

$$\begin{aligned} c_{best} &= \arg\max_c P(c|f_1, f_2, \ldots, f_n) \\ &= \arg\max_c \frac{P(c) \cdot P(f_1, f_2, \ldots, f_n|c)}{P(f_1, f_2, \ldots, f_n)} \\ &= \arg\max_c P(c) \cdot P(f_1, f_2, \ldots, f_n|c) \end{aligned}$$

Protože však předpokládáme, že jsou hodnoty jednotlivých rysů na sobě nezávislé, můžeme napsat:

$$c_{best} = \arg\max_c P(c) \cdot P(f_1|c) \cdot P(f_2|c) \cdot \ldots \cdot P(f_n|c)$$

Parametry pravděpodobnostního modelu se v metodě Naive Bayes určují podle principu maximální věrohodnosti: spočítají se výskyty dané hodnoty a tento počet se vydělí počtem všech výskytů.

3.2 K Nearest Neighbours

Tato metoda, na rozdíl od většiny ostatních metod strojového učení, nevytváří v trénovací fázi žádný model. Pouze si uloží všechny trénovací instance. Při klasifikování pak pouze nalezne K nejbližších instancí a vybere z nich nejčastější třídu. Další variantou je pak vážené hlasování: třídy jednotlivých instancí vážíme podle vzdálenosti od klasifikované instance.

3.3 Support Vector Machine

Metoda Support Vector Machine je jedna z nejsilnějších metod strojového učení. Tato metoda hledá v prostoru trénovacích instancí nadrovinu, která nejlépe odděluje pozitivní a negativní instance. Jinými slovy: vzdálenost nejbližších instancí od oddělující nadroviny (tzv. *support vectors*), je co největší.

Pokud nejsou trénovací data lineárně separabilní, používá se takzvaná *cost function*, která penalizuje instance vyskytující se na špatné straně nadroviny. Při trénování pak hledáme takovou nadrovinu, která minimalizuje součet takových penalizací.

Dalším zobecněním této metody je to, že prostor instancí nejdříve transformujeme pomocí jádrové funkce s danými parametry.

3.4 Křížová validace

Křížová validace se používá v případech, kdy chceme získat více testovacích sad, abychom otestovali model na různých datech, a při tom použít co nejvíce trénovacích dat.

V této metodě rozdělíme všechna data náhodně do k souborů. Pak vždy použijeme jeden z těchto souborů jako testovací sadu a ostatní soubory použijeme pro natrénování modelu. Tímto způsobem získáme k různých výsledků, ze kterých pak můžeme vypočítat průměrnou hodnotu a střední odchylku.

3.5 Bootstrapping

Pro výpočet konfidenčních intervalů jsem použil metodu zvanou bootstrapping. V případě omezené testovací sady nám tato metoda pomůže vygenerovat mnoho nových virtuálních testovacích sad. Každou novou virtuální testovací sadu vyrobíme tak že náhodně vybíráme instance (s opakováním) z původní testovací sady. Takových sad můžeme snadno a rychle vyrobit třeba 1000. Na každé sadě vyočítáme hodnotu evaluační metriky. Z těchto hodnot už snadno vypočítáme průměnou hodnotu a když zahodíme nejhorších 25 % a nejlepších 25 % výsledků získáme snadno i konfidenční interval.

3.6 Výběr rysů

Důležitou součástí vývoje klasifikátoru je výběr rysů, které bude klasifikátor používat. Ne vždy totiž platí: čím více rysů, tím lépe. Model, který používá méně rysů, se rychleji natrénuje, zabírá méně paměti, provádí rychleji klasifikaci a je méně náchylný k přetrénování. Může tedy dávat lepší výsledky, než klasifikátor používající více rysů.

V této práci jsem pro výběr rysů použil hladový algoritmus, který začíná s prázdnou množinou a v každé iteraci přibere vždy rys, který nejvíce zvýší přesnost klasifikátoru (přesnost je měřena pomocí křížové validace).

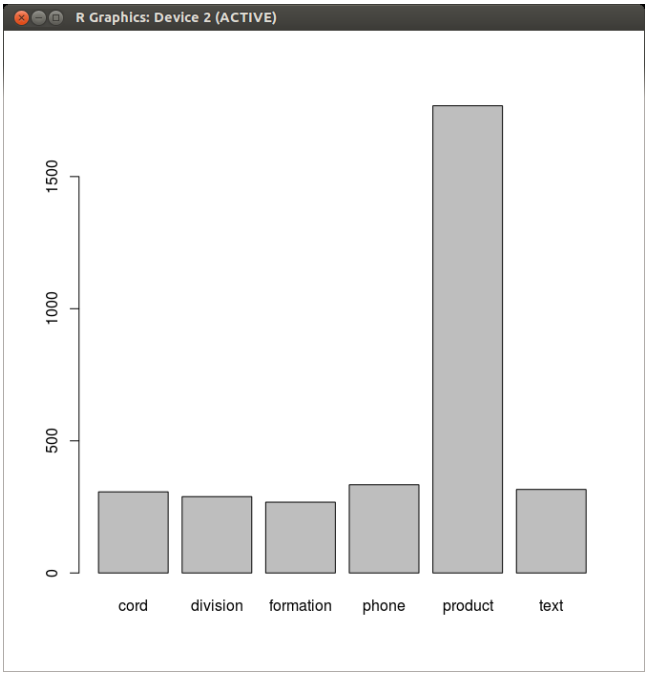
4 Extrahované rysy

Z trénovacích dat jsem extrahoval následující skupiny kategoriálních rysů:

- Lemmata kontextových slov – celkem 9 rysů. Například: *LEMMA_M2* (lemma druhého slova před cílovým slovem), *LEMMA_P1* (lemma prvního slova za cílovým slovem), *LEMMA* (lema cílového slova).
- Morfologické značky kontextových slov – celkem 9 rysů. Například: *TAG_M1* (značka prvního slova před cílovým slovem), *TAG_P2* (značka druhého slova za cílovým slovem).
- Forma cílového slova, značka *FORM*.
- Analitické funkce cílového slova a jeho rodiče – celkem 2 rysy: *AFUN* a *PARRENT_AFUN*.
- Analitické funkce kontextových slov – celkem 4 rysy: *AFUN_M2*, *AFUN_M1*, *AFUN_P1*, *AFUN_P2*.

Metoda	Accuracy
hard	0.810 [0.770,0.851]
line	0.582 [0.530,0.635]
serve	0.397 [0.346,0.450]

Tabulka 2: Výsledky baseline metody



Obrázek 1: Rozdělení tříd v trénovacích datech pro slovo *line*

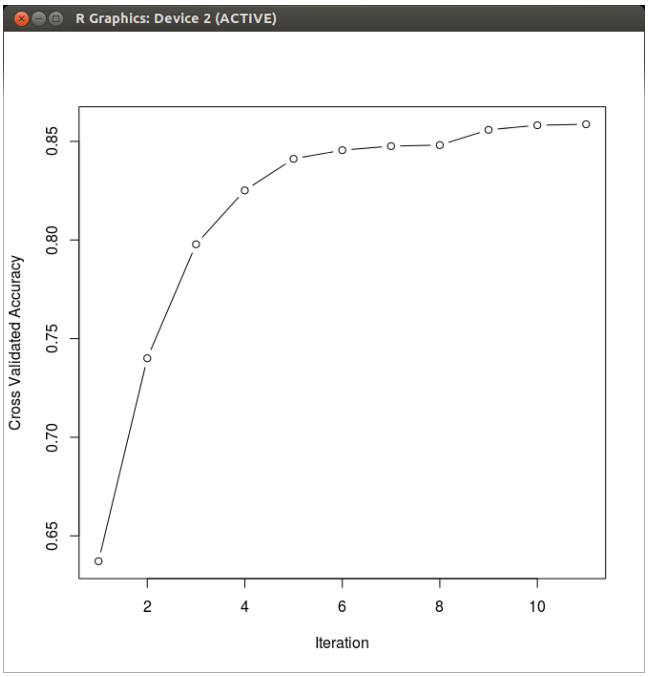
5 Provedené experimenty a výsledky

5.1 Baseline

Jako baseline jsem zvolil nejjednodušší metodu: Klasifikátor vždy vybere nejčastější třídu, kterou viděl v trénovacích datech. Na obrázku 1 můžete vidět, že i takový klasifikátor může fungovat. V tabulce 2 naleznete výsledky této metody.

5.2 Naive Bayes

Pro natrénování modelu Naive Bayes jsem použil funkci *naiveBayes* z balíčku *e1071*. Tato funkce nemá žádné metaparametry, proto jsem pustil rovnou hladový algoritmus pro výběr rysů. Na obrázku 2 můžete vidět postup algoritmu na slově *serve*. V tabulce 3 naleznete vybrané rysy pro každé ze tří slov a celkové přesnosti jednotlivých klasifikátorů spočítané pomocí křížové validace.



Obrázek 2: Vývoj algoritmu pro výběr rysů pro slovo *line*

Cílové slovo	**Vybrané rysy**	**Accuracy**
hard	*LEMMA_P1*, *LEMMA_M1*, TAG, LEMMA_M2, LEMMA	0.890 [0.859,0.919]
line	*LEMMA_M1*, *LEMMA_M2* *LEMMA_P1*, *LEMMA_M3*, *LEMMA*	0.812 [0.769,0.852]
serve	*LEMMA_P1*, *LEMMA_M1*, *LEMMA_P2*, *LEMMA_P3*, *LEMMA_M2*, *LEMMA_P4*, *LEMMA_M4*, *AFUN_P1*, *AFUN*, *LEMMA_M3*, *PARRENT_AFUN*	0.853 [0.817,0.886]

Tabulka 3: Rysy vybrané hladovým algoritmem pro metodu Naive Bayes. Rysy jsou seřazené tak, jak je algoritmus vybral.

Cílové slovo	Vybrané rysy	Accuracy
hard	*AFUN_P2*	0.809 [0.770,0.849]
line	*TAG_M1*	0.588 [0.536,0.637]
serve	*LEMMA_P1*, *TAG_P1*, *LEMMA_M1*, *TAG_M1*, *AFUN_M1*, *TAG*, *AFUN_P1*, *TAG_M2*	0.625 [0.576,0.672]

Tabulka 4: Rysy vybrané hladovým algoritmem pro metodu SVM s defaultními parametry. Rysy jsou seřazené tak, jak je algoritmus vybral.

Cílové slovo	Cost	Accuracy
hard	1000	0.904 [0.872,0.930]
line	1000	0.819 [0.780,0.854]
serve	10000	0.869 [0.835,0.902]

Tabulka 5: Vysledky ladění parametru *cost* u lineárních SVM modelů používající rysy vybrané pro Naive Bayes

5.3 Support Vector Machine

Nejdříve jsem chtěl postupovat podobně jako u metody Naive Bayes: Chtěl jsem vybrat rysy pomocí hladového algoritmu, který bude zkoušet trénovat SVM s lineárním jádrem. V tabulce 4 naleznete rysy, které vybral algoritmus, a přesnosti natrénovaných klasifikátorů. Je patrné, že tento postup moc nefunguje: nepodařilo se dosáhnout o moc vyšších výsledků než baseline a vybrané rysy byly nesmyslné. Tento neúspěch je způsoben nejspíše tím, že SVM modely s defaultními parametry nedávají moc dobré výsledky.

Rozhodl jsem se proto, že vezmu rysy vybrané metodou Naive Bayes a použiju je pro trénování linárních SVM modelů, na kterých vyladím parametr parametr *cost*, který výrazně ovlivňoval výslednou kvalitu klasifikátoru. V tabulce 5 můžete vidět nejlepší hodnoty parametru *cost* s příslušným accuracy. Nejlepší hodnoty parametru *cost* jsem pak použil při poušění hladového algoritmu pro výběr rysů pro lineární SVM modely. Výsledky tohoto experimentu i s vybranými rysy můžete vidět v tabulce 6.

Zkoušel jsem i jiné jádrové funkce, ale bohužel ladění parametrů pro tyto jádrové funkce už bylo velmi výpočetně náročné a nedočkal jsem se skoro žádných výsledků.

5.4 KNN

Tato metoda neumí pracovat s kategoriálními rysy, proto jsem musel všechny kategoriální rysy binarizovat (tedy pro každou hodnotu vytvořit nový rys indikující přítomnost hodnoty). Tím však vznikne z každého rysu tolik nových rysů, kolik je jeho unikátních hodnot. Z toho důvodu jsem v této metodě nemohl použít lexikální rysy (lemmata kontextových slov).

Cílové slovo	Vybrané rysy	Accuracy
hard	*LEMMA_P1*, *LEMMA_M2*, *LEMMA_M1*, *TAG*, *LEMMA_P2*, *TAG_M1*,	0.904 [0.872,0.932]
line	*LEMMA_M1*, *LEMMA_P1*, *LEMMA_M2*, *LEMMA_M3*, *TAG_P2*, *AFUN*	0.819 [0.780,0.857]
serve	*LEMMA_P1*, *LEMMA_P3*, *LEMMA_M1*, *LEMMA_P2*, *LEMMA_M2*, *LEMMA_P4*, *TAG*, *TAG_P1*,	0.868 [0.835,0.899]

Tabulka 6: Rysy vybrané hladovým algoritmem pro metodu SVM s vyladěným parametrem *const*. Rysy jsou seřazené tak, jak je algoritmus vybral.

Metoda	Hard	Line	Serve
Baseline	0.810 [0.770,0.851]	0.582 [0.530,0.635]	0.397 [0.346,0.450]
Naive Bayes	0.890 [0.859,0.919]	0.812 [0.769,0.852]	0.853 [0.817,0.886]
Linear SVM	**0.904 [0.872,0.932]**	**0.819 [0.780,0.857]**	**0.868 [0.835,0.899]**
KNN	0.815 [0.773,0.851]	0.622 [0.569,0.670]	0.705 [0.659,0.749]

Tabulka 7: Výsledky nejlepších klasifikátorů

Zkoušel jsem nastavit parametr k (počet nejbližších sousedů, které beru v úvahu). Už pro hodnotu 2 jsem však dostával chybu, že se při hlasování vyskytuje příliš mnoho remíz. Je vidět, že takto extrahované rysy nejsou vhodné pro metody jako knn.

Výsledky této metody můžete vidět v tabulce 7.

6 Závěr

Cílem práce bylo vytvořit tři klasifikátory pro slova *hard*, *line* a *serve*. Pro tento účel jsem použil metody strojového učení *Naive Bayes* a *SVM* a *KNN*.

V tabulce 7 jsou shrnuté nejlepší výsledky z mých experimentů. V každém sloupci je zvýrazněn výsledek nejlepšího klasifikátoru. Nejlepší metodou se tedy zdá být lineární SVM model, avšak jednoduchá metoda Naive Bayes není o moc horší a zdá se, že se pro úlohu Word Sense Disambiguation hodí také.

Z konfidenčních intervalů můžeme usoudit, že obě metody Naive Bayes a Linear SVM jsou signifikantně lepší než baseline. Už ale nemůžeme říci, že metoda Linear SVM je signifikantně lepší než Naive Bayes.

Výsledky hladového algoritmu ukázaly, že nejdůležitější rysy jsou lemata kontextových slov, zejména nejbližších dvou slov. Trochu pomáhají také morfologické značky. Na druhou stranu výstup z parseru, nepomáhá skoro vůbec.

Nemožnost použití lemat kontextových slov u metody KNN, je tedy hlavní důvod neúspěchu této metody. U slova *hard* tato metoda sotva překonala Baseline.