Úvod do strojového učení (NPFL054)

Zápočtový úkol – určování koreferencí

Ondřej Dušek, IML, 2. roč., 2008/9

1 Cíl úlohy

Cílem úlohy je dosáhnout co nejpřesnějšího *určování koreferencí* v textu za pomoci algoritmů strojového učení. *Koreference* je dvojice slov v textu, které se vztahují ke stejnému mimojazykovému konceptu, sestává z *antecedentu* a *anafory*, tedy prvního, resp. druhého odkazu k onomu konceptu. V datech je vždy dána anafora a k ní možní kandidáti na antecedent, charakterizovaní mnoha lingvistickými vlastnostmi a vztahy k anafoře (viz bod 2). Úkolem je pak vybrat relevantní rysy antecedentů a co nejlépe vyladit dva různé algoritmy používané ve strojovém učení tak, aby s co nejmenším počtem chyb určovaly, zda daný kandidát je, či není antecedentem k dané anafoře. Vyhodnocení úspěšnosti se provádí na základě běžných evaluačních metrik, tj. *accuracy*, *precision*, *recall* a *f-measure* (viz [9]).

2 Data

2.1 Původ a charakter dat

Použitá data pocházejí původně z Pražského závislostního korpusu (PZK) [6], ale byla speciálně upravena za účelem určování koreferencí [5], kdy z jazykových dat v korpusu byly odvozeny potenciálně relevantní rysy. V PZK jsou anotovány dva druhy koreferencí – gramatické (vztažná slova, zvratná zájmena, doplňky, reciprocita a nevyjádřený podmět infinitivu) a textové (osobní zájmena, ukazovací zájmena), se kterými se pracuje i v práci [5] a na ně se tedy soustředí i následující pokusy.

Nepracovalo se s celým souborem dat z PZK, kde se podle [5] nachází celkem 36852 různých anafor, ale s podvýběrem celkem 10001 trénovacích příkladů a 3009 oddělených pro testování, náležejícím celkem 922, resp. 272 různým anaforám. Toto omezení bylo provedeno hlavně pro umožnění efektivního provedení pokusů v programovacím prostředí R. Pozitivních příkladů (tj. takových, kdy kandidát opravdu přísluší zkoumané anafoře) je v použitých datech cca 13%.

2.2 Popis jednotlivých rysů

Data sestávají z jednotlivých příkladů, kdy vždy jeden kandidát na antecedent k nějaké dané anafoře je popsán celkem 54 rysy (převzatými z [5]) a nakonec ohodnocen údajem typu 0/1 (*cílová proměnná*), zda opravdu přísluší k oné anafoře. Některé z rysů jsou číselné, většina (30)

však kategoriálních, tj. nabývajících několika málo hodnot z dané množiny (většinou označených názvem).

Rysy zahrnují mj. odlišení jednotlivých anafor (k jedné anafoře se může vztahovat víc potenciálních antecedentů), různé jazykové kategorie daného kandidáta, stejně jako jejich shodu s anaforou, případně další vlastnosti vzhledem k anafoře (vzdálenost apod.). Pocházejí z anotace PZK, jedná se tedy o morfologické a syntaktické kategorie kandidáta či anafory a jejich vztahy, stejně jako o vztahy a závislosti na úrovni významu (v PZK *tektogramatická* rovina).

Proměnnou identifikující anaforu je nutné od ostatních odlišit – pro určení správnosti antecedentu nemá žádný význam, protože nové příklady vždy budou posuzovány nezávisle na sobě, bez ohledu na to, zda přísluší stejné anafoře. Je zřejmé, že některé z ostatních rysů mají velký vliv na cílovou proměnnou; hodnoty mnohých proměnných naopak s touto vlastností nijak zvlášť nesouvisí (ani z lingvistického hlediska, ani podle různých statistik, např. těch zmíněných v bodu 4.2.1).

3 Hodnocení úspěšnosti pokusů

3.1 Hodnotící metrika

Výsledky pokůsů – predikce jednotlivých algoritmů – je možné porovnat s reálnými hodnotami pomocí několika metrik, zmíněných již v bodu 1.

První z možných, accuracy, je čistě procento správně ohodnocených příkladů. Vzhledem k tomu, že pozitivních příkladů je jen 13% (viz bod 2.1), bude taková míra značně nepřesná – např. i algoritmus, který by všechny příklady hodnotil negativně, dosáhne 87% accuracy.

Tabulka 1: Precision, recall a f-measure

	predikce	1	0
reálná hodnota	1	*tp* *	*fn*
	0	*fp*	*tn*

$P^{\dagger} = \frac{tp}{tp+fp}$ $\quad R = \frac{tp}{tp+fn}$ $\quad F = \frac{2P \cdot R}{P+R}$

* *tp* – správně ohodnocené pozitivní příklady, *fn* – špatně ohodnocené pozitivní příklady, *fp* – špatně ohodnocené negativní příklady, *tn* – správně ohodnocené negativní příklady

† P – precision, R – recall, F – f-measure

Míry precision a recall, uváděné většinou dohromady, jsou pro naše použití mnohem výhodnější, protože závisí zejména na správném ohodnocení pozitivních příkladů (viz Tabulka 1). Precision je poměr správně ohodnocených pozitivních příkladů ku všem příkladům ohodnoceným jako pozitivní, recall poměr správně ohodnocených pozitivních ke všem ve skutečnosti pozitivním.

Pro porovnávání výsledků je ale jednodušší použít jen jednu hodnotu. Nejběžněji uváděná pro pokusy tohoto typu je f-measure [9], harmonický průměr dvou předchozích hodnot (viz

Tabulka 1). Precision, recall a f-measure se používají i v práci [5] a pro binární klasifikační úlohy, což je i tento případ, jsou standardní metrikou.

3.2 Konfidenční intervaly pro hodnoty metriky

Aby bylo výsledky možné porovnávat, je nutné zjistit nejen hodnotu evaluační metriky, ale i její interval spolehlivosti (tj. interval, ve kterém se ona hodnota může s danou pravděpodobností pohybovat při použití algoritmu na různých datech). Pro stanovení tohoto rozmezí jsem použil metodu zvanou *bootstrap* (viz [11]), neboť její výpočet je rychlý a hodí se i pro pokusy s řídkými daty, jako je tento (tj. máme příliš mnoho možných hodnot atributů vzhledem k počtu trénovacích i testovacích příkladů), protože nemění množinu hodnocených dat příliš.

Odhad intervalů spolehlivosti pomocí *bootstrapu* probíhá nad hotovými výsledky predikce pro všechna testovací data. V mnoha bězích vybereme menší podmnožinu z trénovacích příkladů a spočítáme hodnotu evaluační metriky (např. f-measure) pouze za použití predikce a reálných data na této podmnožině. Z každého běhu dostaneme jednu hodnotu, takže máme ve výsledku množinu hodnot, u které je možné po jejím setřídění určit, ve kterém rozmezí se nachází zvolené procento prvků. Toto rozmezí je potom odhadem intervalu spolehlivosti.

4 Postupy pro výběr rysů

Protože příliš vysoký celkový počet rysů a irelevance mnoha z nich na jednoduché algoritmy strojového učení při použití v této úloze působily negativně (viz výsledky za použití všech rysů v bodě 8), je třeba vybrat ty nejrelevantnější z nich. K nalezení optimální množiny rysů, která by dávala nejpřesnější určení koreference, by se dalo dojít jen pomocí vyzkoušení všech podmnožin rysů, což samozřejmě není vzhledem k jejich počtu (2^{54}) výpočetně proveditelné. Je proto nutné použít nějaké heuristiky, která najde sice suboptimální, přesto však „dostatečně vhodnou“ množinu rysů.

V těchto pokusech jsem pro výběr rysů zvolil dva různé přístupy: *hladový výběr* (viz bod 4.1) a *postupný výběr na základě statistik* (viz bod 4.2). Oba dva druhy algoritmů jsou tzv. *stepwise* – jedná se o postupné přidávání rysů do množiny již používaných pro predikci.

4.1 Hladový výběr

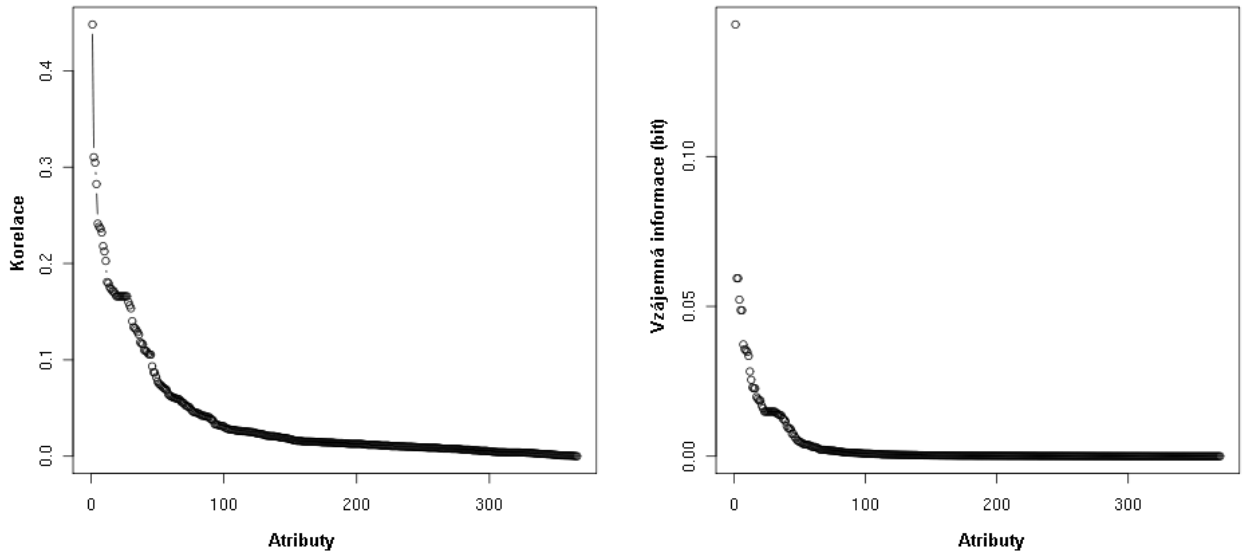
Jedním možným postupem je *hladový výběr* (viz např. [2, s. 316nn]). V každém kroku se vybere jeden rys, který dává nejlepší výsledky, tj. vyzkouší se v jednom kroku všechny možné rysy, ale již vybrané se nikdy nevrací zpět. K hodnocení se používá přímo natrénování testovaného algoritmu strojového učení a evaluace proti některému ze standardních skóre (v tomto případě, jak už bylo řečeno, f-measure), což je i tak výpočtně docela náročné; proto jsem udělal při praktickém použití tohoto způsobu určitou preselekci některých rysů (viz bod 6.1).

4.2 Výběr na základě statistik

Statistikou pro tento případ rozumíme nějaký měřitelný vztah jednotlivých rysů k cílové proměnné (korelace, vzájemná informace, *Minimum Redundancy – Maximum Relevance*; viz body 4.2.1 a 4.2.2). Nejprve se pro všechny rysy spočítá ona statistika a potom se podle ní seřadí (dostaneme tak posloupnost $r_1, \ldots, r_n$). Následně určíme iniciální množinu R_0 využívaných rysů (např. $r_1, \ldots, r_k$). Pak v j-tém kroku $R_j := R_{j-1} \cup \{r_{k+j}\}$, pokud algoritmus strojového učení dosahuje lepší výsledky[1] s $R_{j-1} \cup \{r_{k+j}\}$ než s R_{j-1}. Jinak $R_j := R_{j-1}$. Když projdeme všechny rysy (po $n - k$ krocích), R_{n-k} je výsledná množina používaných rysů.

4.2.1 Výběr na základě korelace a vzájemné informace

Obrázek 1: Velikost korelace, resp. vzájemné informace jednotlivých rysů k cílové proměnné



Jako metriku, hodnotící vhodnost jednotlivých rysů pro použití v algoritmu strojového učení, je možné uvažovat jejich *korelaci* (velikost lineární závislosti; viz [10]) nebo *vzájemnou informaci* (viz např. [3, s. 66]) s cílovou proměnnou. Z grafů (Obr. 1) a Tabulky 2 je vidět, že tímto způsobem lze vydělit jen několik málo rysů a ostatní ignorovat – naprostá většina rysů má jak korelaci, tak vzájemnou informaci s cílovou proměnnou blízkou nule. Je tedy možné dále omezit počet vůbec uvažovaných rysů na nižší číslo, aby se výpočet zrychlil.

4.2.2 Minimum Redundancy – Maximum Relevance

Přesnější, ale také výpočetně mnohem náročnější je algoritmus *Minimum Redundancy – Maximum Relevance* (mRMR, podrobnosti viz [4]). Snaží se na základě vzájemné informace vybrat

[1] Vzhledem k evaluační metrice, tedy zde f-measure.

Tabulka 2: 10 nejvýznamnějších atributů podle korelace a vzájemné informace

atribut* podle:	**korelace**†	**vzáj. informace**
1.	*gen_agree*	*gen_agree*
2.	*cand_subj*	*cand_subj*
3.	*cand_fun.ACT*	*cand_fun.ACT*
4.	*fun_agree*	*num_agree*
5.	*cand_afun.AuxY*	*cand_akt*
6.	*num_agree*	*fun_agree*
7.	*cand_akt*	*akt_agree*
8.	*cand_gen.anim*	*cand_tfa.f*
9.	*cand_tfa.f*	*cand_afun.AuxY*
10.	*akt_agree*	*cand_gen.anim*

*Pro srovnání korelace a vzájemné informace jsou atributy převedeny z kategoriálních na spojité. Podrobnější informace viz bod 6.

†Korelace i vzájemná informace s cílovou proměnnou jsou měřeny na trénovacích datech.

ty rysy, které jsou *co nejrelevantnější* pro cílovou proměnnou (mají s ní co největší vzájemnou informaci), ale *nejsou redundantní* (mají co nejnižší celkovou vzájemnou informaci s již vybranými rysy). Postupné vybírání rysů, které mají v každém kroku nejvyšší hodnotu funkce:

$$mRMR = I(x_j, y) - \frac{1}{m-1} \sum_{x_i \in S_{m-1}} I(x_i, x_j) \tag{1}$$

kde I značí vzájemnou informaci, y je cílová proměnná, S_m je množina vybraných rysů v m-tém kroku, X množina všech rysů a $x_j \in X \setminus S_{m-1}$, dává opět pořadí rysů, podle kterého můžeme na základě zlepšení přesnosti algoritmu vzhledem k evaluační metrice přidávat postupně další do množiny těch skutečně použitých.

5 Algoritmy strojového učení

Algoritmy strojového učení používané v této práci vždy předpokládají nějaká *trénovací data*, kde jsou zadány jednotlivé příklady, jejich rysy a výsledné správné ohodnocení. Pomocí těchto příkladů se algoritmus *natrénuje*, tj. nastaví svoje parametry pro další používání, a může tak rozhodovat i o dalších, doposud neviděných příkladech (proti rysům vydá „správné“ ohodnocení příkladu podle svého natrénování).

5.1 K-nearest neighbor

Algoritmus *K-nearest neighbor* (K-NN, detailní popis viz [1, s. 230nn]) patří do třídy postupů učení založeného na příkladech. Uchovává totiž v paměti všechny trénovací příklady a vyhledává

v nich pro nově zadaný příklad ty s nejpodobnějšími rysy a klasifikuje ho podle nich; neuvažuje tedy všechny trénovací příklady pro každý nový příklad. Míra podobnosti se určuje jako blízkost v zadané metrice, implementace použitá v této práci používá standardní euklidovskou metriku (viz [7]). Formálně pro nový příklad x_q a uložená trénovací data E vypadá postup následovně:

1. Je-li $x_q \in E$, potom přímo vydej hodnocení.
2. Jinak najdi $x_1, \ldots, x_k \in E$, které jsou podle euklidovské metriky nejbližší x_q (mají nejnižší $d(x, x_q) = \sqrt{\sum_{i=1}^{n}(a_i(x) - a_i(x_q))^2}$).
3. Najdi:

$$\hat{f}(x_q) = \text{argmax}_{v \in V} \sum_{i=1}^{k} w_i \cdot \delta(v, f(x_i)), \text{ kde: } \delta(x, y) = \begin{cases} 1 \text{ pro } x = y \\ 0 \text{ jinak} \end{cases} \quad \text{a } w_i = \frac{1}{d(x_q, x_i)^2} \tag{2}$$

   Tj. přiřaď x_q takovou hodnotu, kterou má nejvíce z k nejbližších testovacích příkladů (vážených v závislosti na jejich vzdálenosti)).

5.2 Rozhodovací stromy

Rozhodovací stromy jsou sekvence vnořených větvení, které na základě rozhodnutí založených na hodnotách jednotlivých rysů dojdou až k finální klasifikaci nového příkladu (přesnější pojednání viz např. [1, s. 52nn]). Celkově se klasifikační funkce dá popsat jako disjunkce mnoha konjunkcí.

Pro tuto metodu strojového učení znamená natrénování právě vytvoření statického rozhodovacího stromu s pomocí trénovacích dat; nové příklady se již klasifikují jen jeho průchodem. V každém kroku se rozhoduje na základě hodnoty právě jednoho rysu, a to toho, který je z nějakého hlediska pro cílovou klasifikaci nejzásadnější (např. v této práci použitá implementace *rozhodovacích stromů* v prostředí R *rpart* ([8]) vybírá při konstrukci stromu rys podle *Giniho koeficientu* na trénovacích datech). Další větvení se staví jen na základě podmnožiny trénovacích dat, která již splňuje požadavky na hodnoty dříve vybraných rysů.

Při vytváření rozhodovacího stromu na základě trénovacích dat hrozí riziko *overfittingu*, tedy přílišné specifičnosti stromu, který pak funguje dobře jen na trénovacích datech a na jakýchkoliv jiných selhává. Řešením tohoto problému je *prořezávání*, kdy se větve stromu, které operují jen s příliš malou podmnožinou trénovacích dat, odstraní a nahradí přímou klasifikací nejčastější hodnotou výsledné funkce na oné podmnožině.

6 Provedené testy

Veškeré výpočty jsem provedl v prostředí R, použil jsem kromě základní ještě knihovny *class* a *rpart*. Všechny testy a pomocné skripty, na které odkazuji v této kapitole, jsou součástí přílohy k této práci a jde o moje vlastní implementace. Testoval jsem zmiňované metody výběru rysů

(viz bod 4) i jiné možnosti vyladění nejprve pro algoritmus *K-nearest neighbor* a potom pro *rozhodovací stromy*. Čistě pro srovnání uvádím i výsledky při použití úplně všech rysů.

Nejprve jsem mírně upravil datový format vstupu. Zadaná vstupní trénovací a testovací data jsem předem pomocí jednoduchého perlovského skriptu (`prepare.pl`) doplnil o názvy jednotlivých rysů. Protože u obou metod jsou jednotlivé příklady posuzovány odděleně, bylo by zbytečné uvažovat příslušnost několika kandidátů ke stejné anafoře a výsledky by tím mohly být zkreslené. Rys, který tuto informaci obsahuje, byl proto při všech testech ignorován.

U algoritmu K-NN dále bylo nutné převést kategoriální rysy na spojité (více viz bod 7.1). Následně byly provedeny tyto testy výběru nejvhodnějších rysů pro oba algoritmy, podle postupů popsaných v bodě 4:

(g) *Hladový výběr rysů* pro K-NN i *rozhodovací stromy* (v tomto testu byly použity jen některé rysy, podrobnosti viz 6.1),

(c) *Výběr rysů na základě korelace* jen pro K-NN, protože korelace nemá pro kategoriální rysy smysl (postup viz 6.2),

(i) *Výběr na základě vzájemné informace* pro oba algoritmy (postup je stejný, pouze pracujeme se vzájemnou informací místo korelace – viz 6.2),

(m) *Minimum Redundancy – Maximum Relevance* výběr rysů taktéž pro oba algoritmy (a taktéž se stejným postupem, jen při použití atributů podle pořadí z výstupu algoritmu mRMR, také viz 6.2)

Každý z těchto testů byl proveden u obou uvažovaných algoritmů[2] strojového učení několikrát s různým nastavením. Detailní postup testů je popsán v bodě 7.

V rámci jednotlivých postupů pro výběr rysů se zlepšení výsledků považuje zvýšení f-measure na testovacích datech o více než 0.005; příliš malé zvýšení by totiž mohlo být způsobeno čistě charakterem konkrétních testovacích dat. Pro nejlepší výsledky obou algoritmů se uvádí i hodnoty v rámci dalších metrik – accuracy, precision, recall a f-measure. Moje implementace těchto funkcí v prostředí R, použitá pro výpočet v projektu, se nachází ve skriptu `p-r-f.R`.

Pro nejlepší výsledky z obou algoritmů jsem nakonec určil intervaly spolehlivosti pomocí *bootstrapu* (podle bodu 3), aby je bylo možné porovnat. Pro tento algoritmus jsem zvolil 1000 opakování na náhodných podmnožinách velikosti 300 (cca 10% celkového počtu testovacích příkladů) a na základě jejich výsledků vybral 90% intervaly spolehlivosti. Spočítání konfidenčních intervalů metodou *bootstrap* jsem naprogramoval ve skriptu `bootstrap.R`.

6.1 Preselekce rysů pro hladový výběr

Protože opakované zkoušení všech rysů podle hladového algoritmu (viz bod 4.1) by při jejich vysokém počtu bylo neúměrně výpočetně náročné, vybral jsem **ručně** na základě svých lin-

[2] S výjimkou výběru rysů na základě korelace, jak už bylo zmíněno.

gvistických znalostí a náhledu na trénovací data pro postup výběru rysů hladovým způsobem následujících 13 rysů:

- shodu v rodě a čísle a shodu v lemmatu – *gen_agree, num_agree, lemma_agree* (sice podle dat jsou tyto jevy mnohem častější než koreference, ale měly by ji pomoci určit – vyloučit negativní případy; většina koreferencí má v tomto shodu (88%, resp. 95% a 75%))
- přítomnost ve stejné koordinaci a sourozenectví (předp. neg. vymezení) – *app_in_coord, sibl*
- shodu v aktantu, shodu vlastnosti „být podmětem“ – *akt_agree, subj_agree* (u kladných případů v trénovacích datech je nadprůměrná)
- vzdálenost (jak po větách, tak po klauzích, tak po slovech) – *sent_dist, clause_dist, cand_ord* (všechny hodnoty jsou u kladných případů nižší než průměr, tedy většina koreferencí nedosahuje na příliš velkou vzdálenost)
- TFA kandidáta (podle trénovacích dat předp. „t“ – kontextová zapojenost) – *cand_tfa*
- přítomnost ve stejné kolokaci a vícenásobný výskyt kandidáta – *coll, cand_freq* (u kladných případů jsou obě častější)

V tomto pokusu jsou jsou všechny ostatní rysy ignorovány. Hladový algoritmus tedy má k dispozici jen jejich zúžený počet, na rozdíl od všech ostatních, které pracují s plnou sestavou rysů.

6.2 Výběr rysů na základě statistik

Rysy byly napřed seřazeny podle jedné z metrik (korelace, vzájemná informace, mRMR) a následně se v tomto pořadí od „nejdůležitějšího“ testovalo, zda přidání rysu mezi uvažované přinese signifikantní zlepšení. V případě, že ano, byl tento rys přidán do množiny používaných. Každý rys byl takto testován jen jednou podle předem daného pořadí, jedním průchodem přes rysy se takto získala množina používaných rysů a její hodnocení (tato metoda se někdy označuje jako *forward stepwise*).

Pro testy tohoto typu jsem v prostředí R implementoval algoritmus pro výpočet vzájemné informace dvou proměnných a také na něm založený algoritmus mRMR. Oba jsou vypsány v souboru `mRMR.R`.

7 Praktický postup s jednotlivými algoritmy strojového učení

7.1 Výběr rysů s K-nearest neighbor

Pokusy s tímto algoritmem jsem prováděl za pomoci jeho implementace v prostředí R v knihovně *class* [7].

Protože algoritmus pracuje výhradně se spojitými proměnnými, bylo nejprve třeba kategoriální data převést. Použil jsem jednoduchou metodu, kdy se pro každou klasifikaci kategoriální proměnné zavede nový rys s hodnotami 1 a 0, podle toho, zda daný příklad má, či nemá u oné proměnné právě žádanou klasifikaci (viz implementace této funkce v souboru `categ.R`). Tím se celkový počet rysů zvýšil na 370.

Výpočty pomocí implementace tohoto algoritmu v prostředí R jsou částečně založeny na generátoru náhodných čísel (algoritmus nerozhodné případy řeší náhodně, srov. [7]), před každým testem jsem proto pevně nastavil random seed, aby bylo možné výsledky opakovat. Navíc se každé měření f-measure provádí 5× za sebou, aby bylo jisté, že zlepšení je opravdu signifikantní. Výsledné statistiky jsou vypisovány na základě jednoho dalšího výpočtu se stejnou podmnožninou rysů; je totiž nutné vypsat i ostatní skóre. Při zkoušení K-NN se také občas vyskytuje příliš vysoký počet stejně hodnocených příkladů, který způsobí zastavení této implementace algoritmu v prostředí R[3]. Proto bylo nutné vyhodnocovat až na základě několika testů a nezačínat jen s jednou proměnnou, aby se skripty vůbec provedly.

Byly provedeny následující testy s výběrem rysů, všechny navíc pro různé hodnoty k (tj. počtu uvažovaných „nejbližších“ trénovacích příkladů), $k \in \{1, 3, 5, 7, 9, 10\}$.

(g) *Hladový výběr*
Hladový výběr (implementovaný v `knn-greedy.R`) začíná se dvěma rysy v množině použitých pro klasifikaci, *clause_dist* a *cand_ord*, které zajistí, že algoritmus K-NN nikdy neselže. Postupně jsou přidávány rysy z podmnožiny popsané v bodě 6.1 – v každém kroku takový, který vede k nejlepšímu zvětšení f-measure; pokud přidání žádného z rysů nevede k signifikantnímu zlepšení, algoritmus se zastaví a vydá výsledek pro dané k. Jediný rys, který bylo nutné převést na spojitý v tomto pokuse, byl *cand_tfa* (získáme tak 3 nové rysy).

(c) *Výběr na základě korelace*
Tento pokus probíhá pro každé k přesně podle popisu z bodu 6.2, kritériem řazení rysů je absolutní hodnota korelace s cílovou proměnnou. Počet skutečně uvažovaných rysů byl pro rychlejší průběh pokusu snížen na 100; nedá se předpokládat, že by další rysy s příliš nízkou korelací výsledek nějak výrazně vylepšily. Aby se algoritmus K-NN nezastavil z důvodu příliš vysokého počtu stejně hodnocených prvků, bylo nutné začínat s 10 rysy

[3] Algoritmus selže s chybovou hláškou `Too many ties in k-nn`.

s nejvyšší korelací (experimentálně ověřeno). Implementace pokusu se nachází v souboru `knn-cor.R`.

(i) *Výběr na základě vzájemné informace*
Pro tento druh pokusu platí úplně totéž, co pro předchozí, s tím rozdílem, že kritériem řazení rysů byla velikost vzájemné informace s cílovou proměnnou; je popsán v `knn-mi.R`.

(m) *Výběr na základě Minimum Redundancy – Maximum Relevance*
Zde byly rysy řazeny pro pokusné přidávání do množiny používaných (bod 6.2) podle kritéria daného výpočtem mRMR. Algoritmus je upraven tak, že začíná s jedním rysem a přidává další, dokud K-NN selhává; pak přidává už jen na základě zlepšení výsledků[4]. Příslušný skript je v souboru `knn-mRMR.R`.

7.2 Výběr rysů s rozhodovacími stromy

Pro pokusy s *rozhodovacími stromy* byla použita implementace v R v knihovně *rpart* ([8]). Rozhodovací stromy je možné postavit (především) nad kategoriálními rysy, nebylo tedy třeba žádného převodu. Taktéž celý proces probíhá naprosto deterministicky, takže nastavení generátoru náhodných čísel ani opakování není třeba.

Kromě výběru vhodné podmnožiny jsem zkoušel nastavení různé úrovně prořezávání (v implementaci *rpart* nastavení *complexity*$\in \{0.05, 0.01, 0.005, 0.001, 0.0005\}$), abych nalezl optimum mezi přílišnou obecností a overfittingem.

Protože se pracovalo s kategoriálními rysy, nebylo možné provést postupný výběr na základě korelace s cílovou proměnnou jako u K-NN – tu lze spočítat jen na základě číselných hodnot. Ostatní pokusy bylo možné provést obdobně (jsou jeden po druhém naprogramovány ve skriptu `rpart.R`):

(g) *Hladový výběr*
Algoritmus začíná klasifikovat za pomoci tří rysů, které jsou nejvýhodnější z hlediska mRMR (*gen_agree*,*num_agree* a *cand_tfa*); je to nejnižší počet, pro který postavený rozhodovací strom vůbec klasifikuje nějaké instance jako pozitivní. Jinak probíhá stejně jako u K-NN, ke klasifikaci se přidává ten, který nejvíce zvýší f-measure, dokud je zvýšení signifikantní.

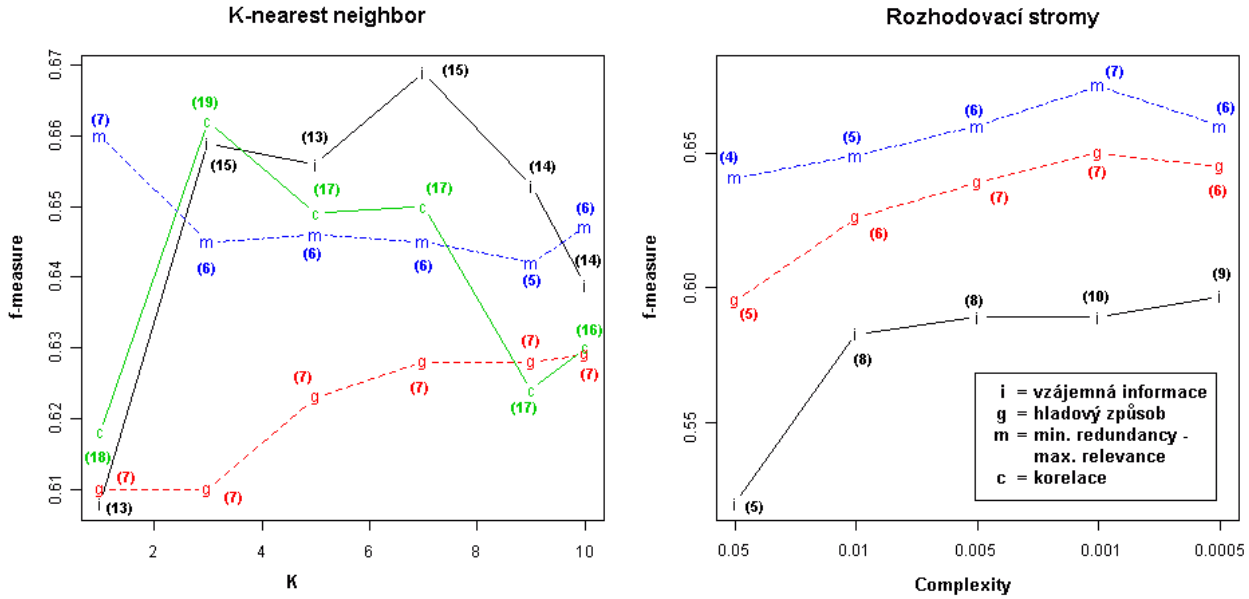
(i) *Výběr na základě vzájemné informace*
Postup je opět podobný jako u K-NN, nebylo tu ale nutné omezovat počet rysů a testují se postupně všechny; začíná se s jedním rysem a přidávají se další v pořadí podle velikosti vzájemné informace s cílovou proměnnou, dokud vytvořený strom neklasifikuje nějaké instance jako pozitivní, dále se přidávájí jen takové, které způsobí signifikantní zlepšení.

[4] V tomto případě totiž minimální počet rysů potřebných k úspěšnému proběhnutí K-NN závisí na k.

(m) *Výběr na základě Minimum Redundancy – Maximum Relevance*
Tento pokus probíhá úplně stejně jako předchozí, jen pořadí rysů je podle výstupu algoritmu mRMR.

8 Výsledky

Obrázek 2: Výsledná f-measure všech pokusů s oběma algoritmy v různém nastavení



Čísla v závorkách označují celkový počet použitých rysů v daném případě.

8.1 K-nearest neighbor

Když jsem algoritmus *K-nearest neighbor* zkusil natrénovat s použitím úplně všech rysů, maximální dosažená f-measure byla přibližně 0.54 (v závislosti na k a nastavení generátoru náhodných čísel), tedy nepříliš vysoká. Všem testům na výběry atributů (viz body 6 a 7.2) se tento výsledek povedlo zlepšit.

U všech provedených testů bylo dosaženo f-measure vyšší než 0.60. Nejnižší hodnotu z nich vykázal test s hladovým výběrem *(g)* – pouze 0.62. Omezení počtu rysů bylo pro tento algoritmus zřejmě příliš přísné. Nejlepší výsledky všech ostatních testů (výběru podle korelace *(c)*, vzájemné informace *(i)* i mRMR *(m)*) leží nad 0.65 (viz Obrázek 2).

Ukazuje se, že pokud uvažujeme jen několik málo rysů, má zvýšení k pozitivní vliv, lze ho ale vyvážit použitím více proměnných (ačkoliv $k = 1$ nefunguje dobře skoro nikdy, vysoký výsledek testu s algoritem mRMR pro $k = 1$ je zřejmě závislý na konkrétních testovacích datech). Nejlepšího skóre (viz Tabulka 4) bylo dosaženo pro $k = 7$ a přidávání rysů v pořadí podle vzájemné informace *(i)*; redundantní rysy se touto metodou zřejmě dobře odstraní samy,

Tabulka 3: Nejlepší výsledky všech typů testů pro K-NN

Test:	(c)	(i)	(m)	(g)
Nejlepší K:	3	7	1	10
F-measure:	0.662	0.669	0.629	0.660
Seznam rysů:	*gen_agree** *cand_subj* *cand_fun.ACT* *fun_agree* *cand_afun.AuxY* *num_agree* *cand_akt* *cand_gen.anim* *cand_tfa.f* *akt_agree* cand_apers.undef tfa_agree cand_afun.Coord *cand_ord* cand_acase.6 epar_fun_agree *sibl* cand_epar_fun.ACT cand_agen.Y	*gen_agree* *cand_subj* *cand_fun.ACT* *num_agree* *cand_akt* *fun_agree* *akt_agree* *cand_tfa.f* *cand_afun.AuxY* *cand_gen.anim* *file_deepord_dist* cand_acase.1 cand_agen.M *sibl* cand_acase.7	*gen_agree* *cand_akt* *num_agree* *file_deepord_dist* *cand_gen.anim* *sibl* anaph_gen.nr	clause_dist *cand_ord* *gen_agree* *cand_tfa.f* *akt_agree* *num_agree* *sibl*
Počet rysů:	19	15	7	7

*Vícekrát použité rysy jsou vyznačeny kurzívou.

proto byl výsledek dokonce lepší než pro mnohem náročněji vypočítané pořadí metodou mRMR *(m)*.

Ta ale naopak měla výhodu, že jí stačilo vybrat mnohem méně rysů než ostatním sekvenčním testům (hladový způsob pomineme). U těch bylo 10 minimální počet vůbec pro to, aby algoritmus K-NN neselhal (viz bod 7.1). Počet skutečně vybraných pro nejlepší výsledky se pak pohyboval mezi 13-15 pro výběr podle vzájemné informace *(i)* a dokonce mezi 16-19 pro výběr podle korelace *(c)*. Při vybírání rysů v pořadí podle výstupu mRMR stačilo 6-7 rysů. Seznam vybraných rysů a jejich počet pro jednotlivé metody ve z hlediska f-measure nejúspěšnějším nastavení je uveden v Tabulce 3; je z ní možné i vyčíst, které rysy byly vybrány ve více případech.

Tabulka 4: Nejlepší dosažené výsledky pro K-nearest neighbor

K: 7

Metoda výběru rysů: Vzájemná informace s cílovou proměnnou *(i)*

Použité rysy (15)*: ***gen_agree***, *cand_subj, cand_fun.ACT,* ***num_agree***, ***cand_akt***, *fun_agree, akt_agree, cand_tfa.f, cand_afun.AuxY, cand_gen.anim,* ***file_deepord_dist***, *cand_acase.1, cand_agen.M, sibl, cand_acase.7*

F-measure: 0.669 Accuracy: 0.918

Precision: 0.716 Recall: 0.629

90% konfidenční interval f-measure: $\langle 0.563, 0.763 \rangle$

		predikovaná 1	0
reálná	1	249	147
	0	99	2514

*Nejdůležitější, popisované v bodě 9.1, jsou vyznačeny tučně.

Jedním z aspektů, které bylo nutno zohlednit při testech s algoritmem K-NN, byla nedeterminističnost jeho implementace v prostředí R. Je zřejmé, že při jiném nastavení náhodného generátoru by na základě jiných výsledků f-measure mohly všechny zkoušené postupy přidat do množiny uvažovaných i jiné rysy než ty, které se ve výsledku objevily, a dosáhnout tak jiných hodnot evaluační metriky. Tento efekt jsem se snažil zredukovat přidáváním rysů na základě průměru f-measure z 5-ti po sobě jdoucích běhů, takže volba rysů pravděpodobně není podmíněna nahodilými výsledky, nelze to ale úplně vyloučit. Neočekávám tedy, že by výsledky v jiném nastavení byly výrazně lepší než ty uvedené.

Algoritmus K-NN vykazoval další zajímavou vlastnost – nastavení k, tedy počtu testovaných nejbližších vzorků, jde proti počtu použitých proměnných. Tedy čím větší k bylo nastaveno, tím menší počet rysů byl vybrán (ve většině případů všech druhů testů). Vyšší k je tak v jistém smyslu výhodnější, když v tomto nastavení algoritmus K-NN potřebuje menší počet proměnných pro podobné výsledky. Toto chování je vidět i na Obrázku 2: výkyvy v f-measure jsou často způsobeny různým počtem používaných rysů pro různá k.

8.2 Rozhodovací stromy

Výsledek použití *rozhodovacích stromů* se všemi rysy byla f-measure rovná 0.556, tedy situace o něco málo lepší než pro K-NN (zřejmě díky prořezávání), ale také neuspokojivá. Vhodný výběr podmnožiny rysů (viz body 6 a 7.2) ale vede k radikálnímu zlepšení.

Rozhodovací stromy mohly pracovat (i díky zachování kategoriálnosti) s mnohem méně rysy než K-NN, ukázaly se ale velmi citlivé na metodu výběru rysů. Z prováděných pokusů dopadl zdaleka nejhůře ten s přidáváním proměnných na základě vzájemné informace *(i)*, výsledná f-measure se pro žádné z testovaných nastavení nedostala nad 0.60. Omezení počtu rysů nemělo až tak negativní vliv, pokud si algoritmus z nich mohl vybrat v libovolném pořadí ty, které přináší nejlepší výsledky; hladový algoritmus *(g)* tedy dosáhl až na f-measure rovnou 0.65.

Nejvíce se tu ale projevil vliv algoritmu mRMR. V tomto pokusu *(m)* bylo dosaženo z hlediska f-measure úplně nejlepšího výsledku, přesahujícího 0.67 (viz Tabulka 6 a Obrázek 3). Je tedy vidět, že pro *rozhodovací stromy* je pořadí výběru rysů zásadní.

Rysy použité jednotlivými postupy výběru *(i), (m), (g)*, jejich počet a dosažený výsledek jsou uvedeny v tabulce 5. Důležité jsou jednak ty rysy, které byly vybrány shodně několika postupy a jednak ty, které jsou shodné mezi K-NN a *rozhodovacími stromy* (srov. i Tabulku 3).

Tabulka 5: Nejlepší výsledky všech typů testů *rozhodovací stromy*

Test:	(i)	(m)	(g)
Complexity:	5e-004	0.001	0.001
F-measure:	0.597	0.675	0.650
Seznam rysů:	*gen_agree**	*gen_agree*	*gen_agree*
	cand_fun	cand_akt	*num_agree*
	cand_afun	*num_agree*	cand_tfa
	cand_subj	file_deepord_dist	sibl
	cand_acase	epar_fun_agree	epar_lemma_agree
	num_agree	cand_coord	akt_agree
	cand_agen	tfa_agree	cand_ord
	app_in_coord		
	anaph_gen		
Počet rysů:	9	7	7

*Vícekrát použité rysy jsou vyznačeny kurzívou.

Změny nastavení *rozhodovacích stromů* dopadlo naprosto podle předpokladů – nižší míra prořezání způsobí „složitější“ rozhodovací strom a tedy i více v něm obsažených proměnných. Všechny testy také ukázaly, že nejlepší úroveň prořezání je 0.001. Při větší složitosti stromů úspěšnost poklesla, projevil se tedy overfitting.

Tabulka 6: Nejlepší dosažené výsledky pro rozhodovací stromy

Complexity: 0.001

Metoda výběru rysů: *Minimum Redundancy – Maximum Relevance (m)*

Použité rysy (7)*: ***gen_agree***, ***num_agree***, ***cand_akt***, ***file_deepord_dist***, *tfa_agree*, *epar_fun_agree*, *cand_coord*

F-measure: 0.675 Accuracy: 0.913

Precision: 0.663 Recall: 0.687

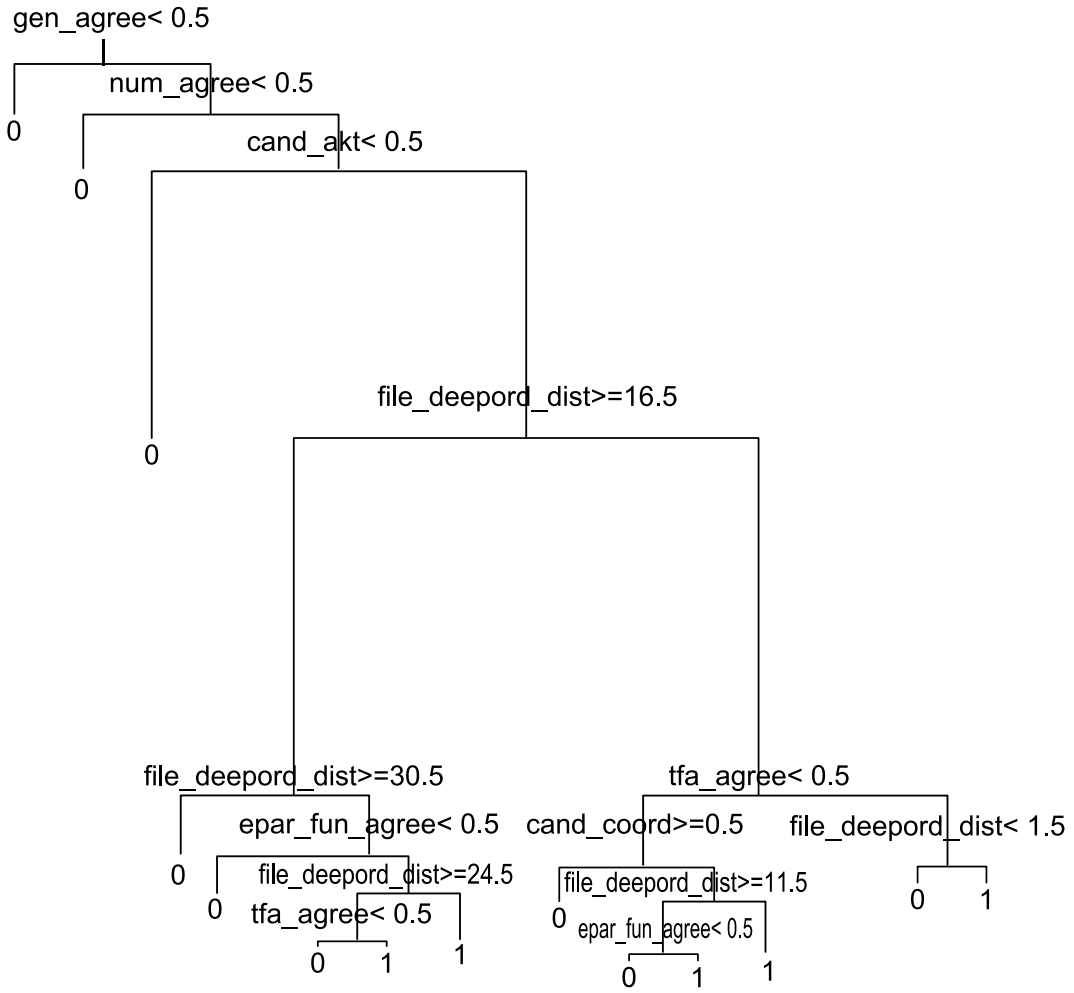
90% konfidenční interval f-measure: $\langle 0.568, 0.768 \rangle$

		predikovaná 1	0
reálná	1	272	124
	0	138	2475

*Použité rysy jsou řazeny podle pořadí důležitosti (viz Obrázek 3). Nejdůležitější, popisované v bodě 9.1, jsou vyznačeny tučně.

Obrázek 3: Nejlepší nalezený rozhodovací strom

Za povšimnutí stojí zejména pořadí vybraných rysů – nejvýše v grafu stojí ty z hlediska algoritmu nejdůležitější.



8.3 Srovnání

Výsledky testů s oběma algoritmy strojového učení jsou podobné. Rozdíl f-measure cca 0.006 ve prospěch *rozhodovacích stromů* není signifikantní, neboť metodou *bootstrapu* odhadnuté 90% konfidenční intervaly se překrývají (a to z převážné části; viz Tabulky 4 a 6). Znamená to, že pro jiná testovací data by algoritmus K-NN mohl dosáhnout lepších výsledků než *rozhodovací stromy*.

Na druhou stranu je vidět, že *rozhodovací stromy* jsou pro tento typ úlohy vhodnější. Nejdůležitější jejich předností je potřeba mnohem nižšího počtu rysů k dosažení dobrých výsledků. Mimo o něco málo vyšší průměrné dosažené f-measure a více správně rozpoznaných pozitivních klasifikací v nejlepším případě (také viz Tabulky 4 a 6) pro ně mluví i to, že proměnné není třeba převádět z kategoriálních na spojité a algoritmus dává v každém opakování přesně stejný výsledek. Natrénování a predikce rozhodovacího stromu také probíhá znatelně rychleji než stejný proces u K-NN, kde je ho nutné navíc pro větší přesnost opakovat.

Jedinou nevýhodou *rozhodovacích stromů* zůstává jejich vysoká citlivost na pořadí výběru rysů; mRMR se pro tuto úlohu prokázal jako velmi dobře použitelný způsob (přes svou vysokou výpočetní náročnost). I pro K-NN při zachování podobných výsledků vybere méně rysů než ostatní metody.

9 Zajímavá pozorování

9.1 Nejdůležitější rysy pro určování koreference

Srovnání seznamu rysů ve výsledku použitých oběma algoritmy ukazuje, které z lingvistických vlastností jsou zřejmě významné pro přítomnost či nepřítomnost koreference. Za nejdůležitější lze považovat shodu v rodě u anafory a antecedentu (*gen_agree*), shodu v čísle (*num_agree*); potom použití kandidáta jako aktantu (povinného valenčního doplnění slovesa; *cand_akt*) a vzdálenost antecedentu a ananafory (v uspořádání na tektogramatické rovině, tedy ne přímo slovní vzdálenost; *file_deepord_dist*). Tyto čtyři figurují mezi proměnnými vybranými oběma algoritmy a v nejlepším rozhodovacím stromě dokonce stojí na nejvyšších příčkách, tedy jako nejdůležitější (viz Obrázek 3).

Důležité je patrně také aktuální členění (kontextová zapojenost či nezapojenost anafory i antecedentu), protože pro nejlepší dosažené výsledky obou algoritmů byly použity rysy, které právě tyto vztahy zachycují. Všechny jmenované důležité rysy (s výjimkou *file_deepord_dist*) mají vysokou hodnotu korelace i vzájemné informace s cílovou proměnnou (viz Tabulka 2).

9.2 Srovnání ručního předvýběru rysů a automatického výběru

Zajímavé je i srovnání ruční selekce rysů podle lingvistických vlastností (viz 6.1) pro výběr hladovým způsobem *(g)* a selekce, kterou provedly automatické algoritmy podle jednotlivých

metrik *(c)*, *(m)* a *(i)*.

Důležitost shody v rodě a čísle (*gen_agree* a *num_agree*) se potvrdila, kdežto shoda v lemmatu zřejmě není pro určení korefence testovanými algoritmy tak podstatná – ani v testech s hladovým výběrem nebyla většinou použita. Stejně tak přítomnost v koordinaci a sourozenectví ani shoda v aktantu nebo bytí podmětem či přítomnost v kolokacích a vícenásobný výsky, které jsem ručně vybral, nemají zřejmě příliš velký vliv. Jinak je tomu ale s aktuálním členěním věty – tady se můj ruční výběr s automatickým shoduje, i když jen částečně (kontextová nezapojenost kandidáta *cand_tfa.t* u K-NN, rozhodovací stromy používají shodu v akt. členění *tfa_agree*). Také se ukázalo, že rysy, týkající se vzdálenosti anafory a antecedenta, jsou správnou volbou. Důležitější než ručně vybrané proměnné, které zohledňovaly spíše syntaktické vzdálenosti, se však jeví vzdálenost na rovině významu (*file_deepord_dist*).

Ze srovnání bodu 6.1 a Tabulek 4 a 6 a Tabulek 3 a 5 je ale vidět, že ručně jsem nevybral několik rysů, které automatické testy označily za relevantní (alespoň v některých případech). Jsou to *tfa_agree* a *cand_akt* (a několik dalších, méně důležitých), u nichž jsem významnější vliv na cílovou proměnou jsem nepřepokládal. Pokud by je měly testy s hladovým výběrem *(g)* k dispozici, pravděpodobně by skončily s lepšími výsledky.

9.3 Srovnání výpočtu se spojitými a kategoriálními rysy

Provedl jsem ještě kontrolní testy *(m)*, *(i)* a *(g)* s rozhodovacími stromy, ale nad spojitými proměnnými. Převod kategoriálních rysů na spojité byl proveden stejně jako u testů s K-NN, viz bod 7.1. Po něm byly výsledky mnohem méně variabilní v závislosti na použité metodě, více se však lišily vzhledem k nastavení úrovně prořezání a počet používaných rysů se ve většině případů zvýšil. Testy s mnohem větším počtem proměnných byly navíc výrazně pomalejší. Zdá se tedy, že kategoriální rysy jsou pro tento typ úlohy užitečnější. Přesné dosažené hodnoty f-measure a počet použitých rysů lze najít v Tabulce 7.

Tabulka 7: Výsledky *rozhodovacích stromů* nad spojitými proměnnými

Test:	**Nejhorší výsledek:**			**Nejlepší výsledek:**		
	complexity	počet rysů	f-measure	complexity	počet rysů	f-measure
(i)	0.05	3	0.520	5e-004	13	0.669
(m)	0.05	4	0.642	0.005	6	0.660
(g)	0.05	6	0.592	0.001	8	0.650

9.4 Další vlastnosti úlohy

Ze všech provedených testů se ukazuje, že největší vliv na výsledek nemá konkrétní algoritmus strojového učení, ale spíš výběr rysů. Oba dva algoritmy dosahují velmi podobných výsledků –

srovnatelně dobrých v nejlepším nastavení a srovnatelně špatných, pokud nedojde ke správné volbě proměnných.

Srovnáme-li zkoušené metody volby proměnných (*(c)*, *(m)*, *(i)*, *(g)*), uvidíme zjevné výhody postupu mRMR *(m)*. Pro oba algoritmy strojového učení dává nejstabilnější výsledky, ukazuje se tedy jako nejvíce robustní. U *rozhodovacích stromů* se projevil jako nejsilnější, co se dosažené f-measure týče, u K-NN zase vykazuje nejstálejší (a velmi dobré) hodnocení v závislosti na nastavení a potřebuje k tomu nejméně rysů (viz Obrázek 2). Jedinou nevýhodou mRMR je fakt, že řazení rysů podle relevance a redundance je časově mnohem náročnější než podle korelace nebo vzájemné informace (mnou implementovaný postup má kvadratickou časovou složitost, takže s vyšším počtem rysů délka výpočtu roste velmi rychle).

Proti tomu test s výběrem rysů podle vzájemné informace *(i)* sice u K-NN skončil s nejlepší zaznamenanou f-measure, ale výsledky se v závislosti na nastavení algoritmu velmi lišily (viz Obrázek 2). Navíc pro *rozhodovací stromy* nad kategoriálními proměnými stejný postup výběru dopadl nejhůře ze všech prováděných testů (nejlepší dosažená f-measure je skoro o 0.10 níže než pro mRMR). Jak je vidět z bodu 9.3, pro spojité proměnné je možné dosáhnout dobrých výsledků i s *rozhodovacími stromy*, a tak je tento postup výběru zřejmě vhodný spíše pro spojité rysy.

10 Závěr

Oba algoritmy strojového učení pracují srovnatelně dobře a f-measure 0.669 u K-NN, resp. 0.675 u *rozhodovacích stromů* je pro tuto velikost trénovacích dat dobrý výsledek, ač pro čistě automatické určování koreferencí nedostačuje; něco takového se však neočekávalo (tato úloha patří mezi velmi obtížné problémy lingvistické analýzy). Rozdíl v f-measure obou metod sice není signifikatní, jiné faktory jako rychlost výpočtu, stabilita výsledků a menší počet rysů potřebných k jejich dosažení však hovoří ve prospěch *rozhodovacích stromů*.

Mnohem větší vliv než použití daného algoritmu strojového učení však na výsledek měla volba rysů. Natrénování algoritmů za použití vhodných proměnných vedlo k výraznému zlepšení predikce. Testy tak poskytly velmi zajímavé informace o relevanci jednotlivých rysů (a tedy s nimi spojených lingvistických vlastností) pro určování koreferencí. Na prvních místech v pořadí důležitosti stojí shoda anafory a antecedentu v rodě a čísle, zapojení antecedentu jako aktantu, „vzdálenost“ obou členů na tektogramatické rovině a také jejich vztahy na poli aktuálního členění věty.

Literatura

[1] Tom M. Mitchell: *Machine Learning.*
NYC: McGraw-Hill, 1997.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein: *Introduction To Algorithms 2nd Edition.*
NYC: McGraw-Hill, 2001.
`http://www.dleex.com/read/?42489` (9. února 2009)

[3] Christopher D. Manning, Hinrich Schütze: *Foundations of Statistical Natural Langugage Processing.*
Cambridge, Massachusetts: MIT Press, 2000.

[4] Hanchuan Peng, Fuhui Long, Chris Ding: *Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy.*
In: IEEE Transactions on Pattern Analysis and Machine Intelligence 8, vol. 27, 2005.

[5] Nguy Giang Linh: *Návrh souboru pravidel pro analýzu anafor v českém jazyce.* Praha: ÚFAL MFF UK, 2006.

[6] *Prague Dependency Treebank 2.0.*
`http://ufal.mff.cuni.cz/pdt2.0/` (9. února 2009)

[7] *R Class Package Documentation: knn.*
`http://stat.ethz.ch/R-manual/R-patched/library/class/html/knn.html`
(9. února 2009)

[8] *Recursive Partitioning and Regression Trees.*
`http://stat.ethz.ch/R-manual/R-patched/library/rpart/html/rpart.html`
(12. února 2009)

[9] *Wikipedia: F-measure.*
`http://en.wikipedia.org/wiki/F-score` (9. února 2009)

[10] *Wikipedia: Correlation.*
`http://en.wikipedia.org/wiki/Correlation` (9. února 2009)

[11] *Wikipedia: Bootstrapping (statistics).*
`http://en.wikipedia.org/wiki/Bootstrapping_(statistics)` (26. února 2009)