Počítačové zpracování přirozeného jazyka

Vyhodnocování úspěšnosti

Daniel Zeman

http://ufal.mff.cuni.cz/course/popj1/

Úspěšnost zpracování jazyka

- Jak ověřit, že program funguje správně?
- 2 části:
  - programátorská (nepadá to, necyklí se to)
  - věcná (jazyk zpracován „správně“)
- Programátorská část:
  - ~~„Existují data, pro která to nepadá.“~~
  - Zkoušet, zkoušet, zkoušet… Množina možných vstupů je nekonečná.

Jazyková část vyhodnocení

- Objektivnost hodnocení
  - objektivní (měřitelné a srovnatelné s okolím)
  - subjektivní (závislé na názoru uživatele)
- Někdy zcela objektivní metoda neexistuje!
  - strojový překlad
- Někdy je dokonce vhodné zahrnout subjektivní hodnocení (ale většího počtu uživatelů)
  - „Jak dobře a snadno se vám se systémem pracovalo?“

Černá a skleněná skříňka

- Máme možnost vidět dovnitř systému?
  - Černá skříňka (black box)
    - Nevidíme dovnitř, hodnotíme chování a výstup celku.
  - Skleněná skříňka (glass box)
    - Umíme systém rozdělit na komponenty a hodnotit každou zvlášť.
    - Umíme říct nejen ke kolika chybám došlo, ale i jaké druhy chyb to byly.

Příklad černé skříňky

- Program pro zodpovídání dotazů na databázi
  - Hodnotíme správnost odpovědí.
  - Při chybě nevíme, zda k ní došlo kvůli
    - neporozumění některému slovu v dotazu
    - neporozumění celé větě v dotazu
    - technické chybě v programu
    - chybě v databázi

Černá a skleněná skříňka

- Statistické metody
  - Obvykle fungují jako černé skříňky
  - Není snadné je rozdělit na menší části
  - U chybného výstupu je těžké vystopovat, proč k chybě došlo
- Pravidlové metody
  - Obvykle lze vystopovat pravidlo-viníka (podobně jako při ladění programu)

Černá a skleněná skříňka: pro a proti

- Vyhodnocování černou skříňkou je to jediné důležité z hlediska koncového uživatele.
  - Až na to, že i pro něj je zajímavé vědět, jaké druhy chyb očekávat.
- Během vývoje je však potřeba vidět dovnitř kvůli snadnější opravě chyb.

Varovný příklad: ELIZA

- Program simulující psychoterapeuta (předpokládá, že uživatel je pacient)
- Komunikuje v přirozeném jazyce (EN)
- Vyvinut v 60. letech (!) 20. století na MIT
- Jestliže nevíte, na čem je založená, na chvíli uvěříte, že funguje dobře.
- Na první pohled mimořádně uměle inteligentní… Tedy:
  - na 1. pohled mimořádně inteligentní
  - na 2. pohled mimořádně uměle ☺

ELIZA

U: Muži jsou všichni stejní.

E: V jakém smyslu?

U: Pořád nás něčím otravují.

E: Máte nějaký konkrétní příklad?

U: Tak třeba můj přítel mě přiměl sem jít.

E: Váš přítel vás přiměl sem jít.

U: Říká, že mám pořád deprese.

ELIZA

E: To je mi líto, že máte deprese.

U: Je to tak, jsem nešťastná.

E: Myslíte, že vaše návštěva zde vám pomůže, abyste nebyla nešťastná?

U: Nějakou pomoc každopádně potřebuju.

E: Co vám to přinese, když dostanete nějakou pomoc?

ELIZA

U: Snad bych se mohla naučit vycházet se svou matkou.

E: Povězte mi více o své rodině.

U: Má matka se o mě stará.

E: Kdo jiný ve vaší rodině se o vás stará?

U: Můj otec.

E: Váš otec.

ELIZA

U: Vy jste v něčem trochu jako můj otec.

E: Jakou podobnost vidíte?

…

„Technologie“ za Elizou

- Porovnávání vzorů (pattern matching)
- Klíčová slova
  - V uživatelově větě *U* najít klíčové slovo, jehož vzor (pattern) odpovídá větě *U*.
  - Odpovídá-li věta vzorům více klíčových slov, vybrat podle jejich priority.
  - Podle věty *U* a specifikace u klíčového slova zkonstruovat reakci.
  - Nebyla-li nalezena žádná klíčová slova, odpovědět něco obecného jako „Řekněte mi o tom více.“ nebo „Pokračujte.“

Eliza: Příklad vzoru

- Otazníkem začínají proměnné, jejichž obsah lze zkopírovat na výstup.
- Klíčové slovo: jsou
- Vzor: ?X jsou ?Y
- Výstup: Co kdyby nebyli ?Y?

Eliza: další příklady vzorů

Slovo	Pri	Vzor vstupu	Výstupy
stejní	10	?X	V jakém smyslu? Jakou podobnost vidíte?
je 	3 3	?X je ?Y ?X ona je ?Y	Co kdyby nebyl ?Y? Byla byste radši, kdyby nebyla ?Y?
vždy	5	?X	Máte nějaký konkrétní příklad? Kdy? Opravdu vždy?
co	2	?X	Proč se ptáte? Zajímá vás to?

Další triky

- Má-li Eliza více výstupů pro jeden vzor, vybere pokaždé jiný, aby se moc neopakovala.

Hodnocení Elizy

- Program nerozumí konverzaci, které se účastní.
- Uživatel to nemusí poznat ihned.
  - Zvláště, je-li skutečně pacientem zralým pro psychiatra!
- V jiných doménách by byl program méně úspěšný
  - Nepotřebuje znalost světa, protože nic netvrdí, s ničím nesouhlasí, na nic neodpovídá.
  - Raději klade sérii otázek.
  - Přímé otázky odráží protiotázkami typu „Proč se ptáte?“ Nelze ji přinutit říct cokoli konkrétního o čemkoli.

Hodnocení Elizy 2

- I v takto omezeném prostoru lze celkem snadno ukázat, že Eliza je hlupák:
  U: Opatrnost je matka moudrosti.
  E: Povězte mi více o své rodině.
- Nevadí jí nesmyslné vstupy:
  U: Zelené ten adzabak je ten kuk běžel čtyři.
  E: Co kdyby nebyl ten kuk běžel čtyři?
- Nemá paměť, klade dotazy, které jsou nesmyslné ve světle toho, co už bylo řečeno.

Po světě běhá spousta Eliz, které si můžete vyzkoušet

- (http://www-ai.ijs.si/eliza/eliza.html) … nefunkční
- http://www.manifestation.com/neurotoys/eliza.php3

- a tak dále: google.com: „eliza therapist“

Domácí „úkol“

- Pohádkové dítě:

  http://ufal.mff.cuni.cz/dite

- Přečtěte si ukázku, stojí za to ☺

Objektivita hodnocení

- Ideální jsou měřitelné metody
- Potřebujeme testovací data (pokud možno v reprezentativním složení a množství)
  - Tzv. zlatý standard (podrobnosti za chvíli)
- Při subjektivním hodnocení alespoň reprezentativní složení a počet hodnotitelů, kteří přiřazují číselné ($\Rightarrow$ zprůměrovatelné) známky.
  - Hodnocení typu „Jé, to je krásný!“ se průměrují těžko.
- Výběr kategorií ke známkování je důležitý

Testovací data

- Modelová úloha: značkování
  - Každému slovu přiřadit značku z předem známé množiny
- Zlatý standard *(gold standard)* představují texty, ve kterých někdo přiřadil ke každému slovu správnou značku ručně.
- Velikost řádově 10 % trénovacích dat
  - Pokud tedy jde o metodu, kde něco trénujeme
  - Obvyklé rozpětí několik set až tisíců vět
  - Ruční anotace je drahá, někdy si musíme aspoň malý vzorek anotovat sami

Vývojová a vyhodnocovací data

- Testovací data se nesmí překrývat s trénovacími!
  - Ani na trénovacích nebude úspěšnost 100 %
    - Odpadnou ale problémy s neznámými slovy a jevy
    - Chceme, aby program uměl zobecňovat, ne aby se naučil papouškovat trénovací data
- Kromě toho 2 poloviny testovacích dat:
  - Vývojová data používáme, dokud program není hotov. Smíme zkoumat chyby, které na nich dělá.
  - Vyhodnocovací (závěrečná) data použijeme na konci jednou, tyto výsledky pak publikujeme.

Klasifikační úlohy

- Klasifikační úloha je taková, kde každému slovu (popř. znaku nebo jinému segmentu) přiřazujeme právě jednu hodnotu z předem známé množiny.
  - Značkování, lematizace, word sense disambiguation, rozpoznávání jednoslovných pojmenovaných entit, háčkování.
- Vyhodnocení: prostá úspěšnost (či chybovost) na slovech.
  - Potřebujeme ručně označkovaná testovací data.
  - Úspěšnost *(accuracy)* je procento slov, jimž jsme přiřadili správnou značku.
  - Někdy se uvádí chybovost *(error rate)*, což je doplněk úspěšnosti do 100 %.

Příklad klasifikační úlohy: háčkování

- Testovací data získáme snadno: stačí několik set nebo tisíc vět (ve kterých nechybí diakritika)
- Odstranit z textu háčky a čárky (jednoduché)
- Nechat testovaný program, aby je vrátil
- Porovnat výstup programu s původním textem
- Zjistit procento správně oháčkovaných slov (alternativou by bylo počítat jednotlivé znaky, ale slova dávají větší smysl)

Úspěšnost háčkování

- Úspěšnost A = G / (G+B), kde
  - G je počet správně oháčkovaných slov
  - B je počet špatně oháčkovaných slov
- Otázky:
  - Počítat i slova, kde nebylo co rozhodovat?
    - Slova, která nepřipouštějí žádnou diakritiku (v češtině vzácné, třeba *blb*)
      - Značkování: slova, která připouštějí jen 1 značku
    - Slova, která by připouštěla diakritiku, ale žádné slovo, které by tím vzniklo, nemáme ve slovníku (problematické, co když máme nekompletní slovník?)

Případy, kde není co rozhodovat

- Program na nich boduje zadarmo.
- Pro hodnocení kvality programu nezajímavé, šlo by vyloučit.
  - Jenže pak bychom měli také rozlišovat případy, kdy jsme vybírali ze dvou variant, od případů, kdy jsme vybírali z dvaceti…
- Pro hodnocení kvality výstupu důležité, ponechat!
  - Ten, kdo bude výstup programu používat, chce znát kvalitu výstupu, ne to, jak obtížně jí program dosáhl.

Úspěšnost hledání hranic vět

- Vstup: text s vyznačenými hranicemi slov.
- Výstup: obohacený o hranice vět.
- 1. možnost: klasifikační úloha
  - Hranice věty je atributem slova: začíná přede mnou věta? (ano / ne)
  - Spočítat procento slov se správně určenou hodnotou tohoto atributu.

Hranice vět

- Problém: výsledná čísla jsou zavádějící.
- Před drtivou většinou slov věta **nezačíná.**
- Průměrná česká věta obsahuje 17 slov. Nepoznáme-li ani jednu větu, dosáhneme úspěšnosti kolem 16/17 = 94 %!
- Tedy jinak: co takhle procento rozpoznaných vět?

Hranice vět

- Problém: nejde jen o zapomenuté hranice vět, často je také přidáme tam, kde nejsou:
  - <s>Narodil se v pondělí po sv. <s>Janu. <s>…
- Úspěšnost = počet nalezených / počet hledaných?
  - Takže třeba 120 %? To nedává smysl.
- Najdeme-li správný počet vět, nemusí to být ty správné věty.
- => místo jedné veličiny budeme sledovat dvě: přesnost a úplnost

Přesnost a úplnost

- Přesnost (precision) *P*: kolik z toho, co jsem našel, jsem měl najít?
- Úplnost (recall) *R*: kolik z toho, co jsem měl najít, jsem našel?

Přesnost a úplnost hranic vět

- 2 soubory: tvrzení (které testuju) a vzor (s nímž tvrzení porovnávám).
- 4 možná hodnocení každého slova:
  - v1t1 = věta před ním začíná ve vzoru i v tvrzení
  - v0t0 = ani ve vzoru, ani v tvrzení *(true negatives)*
  - v0t1 = ve vzoru ne, v tvrzení ano *(false positives)*
  - v1t0 = ve vzoru ano, ne tak v tvrzení *(false negatives)*
- Označme si v1t1 počet slov hodnocených „v1t1“ atd.

Přesnost a úplnost hranic vět

- Úspěšnost (Accuracy)

$$A = (v1t1 + v0t0) / \Sigma\ vXtY$$

- Přesnost (Precision)

$$P = v1t1 / (v1t1 + v0t1)$$

- Úplnost (Recall)

$$R = v1t1 / (v1t1 + v1t0)$$

Přesnost a úplnost hranic vět

- Příklad: 100 vět o celkem 1700 slovech
- Neoznačíme žádnou větu
  - v0t0 = 1600, v0t1 = 0, v1t0 = 100, v1t1 = 0
  - A = 1600 / 1700 = 94 %
  - P = 0 / 0 = ?
  - R = 0 / 100 = 0 %
- Poznáme všechny věty správně
  - v0t0 = 1600, v0t1 = 0, v1t0 = 0, v1t1 = 100
  - A = 1700 / 1700 = 100 %
  - P = 1600 / 1600 = 100 %
  - R = 100 / 100 = 100 %
- Označíme větu před každým slovem
  - v0t0 = 0, v0t1 = 1600, v1t0 = 0, v1t1 = 100
  - A = 100 / 1700 = 6 %
  - P = 100 / 1700 = 6 %
  - R = 100 / 100 = 100 %

Míra F (F-measure, F-score)

- Potíž: dvouhodnotová hodnocení se těžko porovnávají.
- Je lepší program, který dosáhne P = R = 50 %, nebo program, který na stejných datech dosáhne P = 90 % a R = 10 %?
- Odpověď: ten první. Obecně nelze zanedbat ani P, ani R! (V konkrétních aplikacích to může být jinak.)
- F-measure: tzv. harmonický průměr P a R. U blízkých hodnot zhruba průměr, u vzdálených se kloní k té nižší.
- Zde: první program F = 50 %, druhý F = 18 %.

F-measure

$$F = \frac{2PR}{P+R}$$

F hranic vět

- Označíme jedinou větu a ještě špatně
  - A = 1599 / 1700 = 94 %
  - P = 0 / 1 = 0 %
  - R = 0 / 100 = 0 %
  - F = (2 × 0 × 0) / (0 + 0) = ? (0) %
- Označíme jedinou větu, ale správně
  - A = 1601 / 1700 = 94 %
  - P = 1 / 1 = 100 %
  - R = 1 / 100 = 1 %
  - F = (2 × 1 × 0,01) / (1 + 0,01) = 2 %
- Poznáme všechny věty správně
  - A = 1700 / 1700 = 100 %
  - P = 1600 / 1600 = 100 %
  - R = 100 / 100 = 100 %
  - F = (2 × 1 × 1) / (1 + 1) = 100 %
- Označíme větu před každým slovem
  - A = 100 / 1700 = 6 %
  - P = 100 / 1700 = 6 %
  - R = 100 / 100 = 100 %
  - F = (2 × 0,06 × 1) / (0,06 + 1) = 11 %
- Příklad: 100 vět o celkem 1700 slovech

Úspěšnost vs. přesnost a úplnost

- Úspěšnost se hodí pro klasifikační úlohy
  - Každé slovo (písmeno, věta, jiný prvek) má dostat právě jednu nálepku
  - Víme předem, kolik nálepek máme dodat
  - Ví to i hodnocený program a může to dodržet (když odpověď nezná, odpoví náhodně)
- Přesnost, úplnost a F
  - Hodí se tam, kde program předem neví, kolik odpovědí (nalezených jevů) se od něj očekává
  - Případně i u klasifikačních úloh, pokud program umí poznat, kdy si svou odpovědí není jistý, a proto
    - Nechce odpovědět vůbec
    - Nebo chce vrátit množinu možných odpovědí (odfiltruje jen ty, u kterých si je jistý, že jsou špatné)

Závislostní syntaktická analýza

- Každé slovo má právě jednoho rodiče.
  - Strom lze zapsat jako posloupnost odkazů na rodiče.
    - Věta: „Pavel půjčil Petrovi knihu.“
    - Strom: # (půjčil (Pavel, Petrovi, knihu), .)
    - Zápis: 2, 0, 2, 2, 0
- Úspěšnost přiřazování řídících uzlů jednotlivým slovům.
  - Pro každé slovo se podívat, zda index rodiče, který jsme mu přiřadili, odpovídá vzorové anotaci.
  - Vydělit počtem slov v testovacích datech.
- Někdy přísnější metoda: sedět musí celá věta.
  - Po závislostech vychází 70 až 90 %.
  - Po větách je těžké se dostat přes 30 %.

Složková syntaktická analýza

- Počet složek ve větě není předem znám.
  - Příklad:
    - Věta: „Pavel půjčil Petrovi knihu.“
    - Strom 1: (S (NP Pavel) (VP půjčil (NP Petrovi) (NP knihu)) .)
    - Strom 2: (S (VP (NP (NN Pavel)) (VB půjčil) (NP (NN Petrovi)) (NP (NN knihu))) (. .))
- Přesnost, úplnost a F-skóre:
  - P: kolik složek, které jsme našli, jsme měli najít?
  - R: kolik složek, které jsme měli najít, jsme našli?
  - „Labeled precision/recall“ – složku identifikuje rozsah a neterminální symbol. „Unlabeled“ – bez symbolu.
  - Crossing brackets
  - Complete match (závorkování celé věty musí být dobře)
  - Tagging accuracy
- Standardem je program **`evalb`** (Satoshi Sekine, Michael Collins)
- Často se *nezapočítává* interpunkce (ale jak ji poznat?)

Rozpoznávání řeči

- Podobné, jako hranice slov a vět: mohou být popletená slova, přidaná slova, ubraná slova.
- WER („word error rate“) – zvykem je udávat procento chyb, ne správných slov, tj. čím vyšší číslo, tím hůře.
- Počet přidaných slov, chybějících slov, popletených slov.
- Problém se synchronizací
  - Jak poznám, jestli vidím zkomolené slovo w_k, nebo jestli je tohle přidaný šum a slovo w_k uvidím za chvíli správné?

Strojový překlad

- Může existovat několik stejně správných překladů.
- Ruční vyhodnocování
  - Několik hodnotitelů – lidí, známky v různých kategoriích, na konci zjistit průměrnou známku.
  - Zkoumá se zvlášť zachování významu a gramatická správnost.
  - Pracné, drahé, zdlouhavé. Vhodné pro závěrečnou prezentaci systému, nevhodné pro vývoj.
- Automatické vyhodnocování
  - BLEU skóre, NIST, (H)TER, METEOR…
- Poloautomatické vyhodnocování
  - Člověk opraví výstup systému, aby byl gramaticky a významově v pořádku. Nejmenší nezbytný počet změn!
  - Automaticky (objektivně) porovnat výstup systému s výstupem opraveným člověkem.
- O metodách hodnocení překladu už je možná víc publikací než o překladu samotném!

BLEU score

- Čte se (kupodivu) [blú]. Vynález IBM (Kishore Papineni et al. 2001).
- Všichni na něj nadávají a všichni ho používají.
- Záleží na počtu referenčních překladů (typicky 4, pokud jsou k dispozici).
- Zjišťuje se, kolik slov, dvojic, trojic a čtveřic slov z výstupu systému se vyskytuje v referenčních překladech. Vážená kombinace, výsledkem je číslo mezi 0 a 1, resp. mezi 0 a 100.
- Absolutní hodnota neříká nic! Lze pouze porovnávat dva systémy nad stejnými testovacími daty.

Porovnávání kvality dvou programů

- Dva nástroje, které řeší stejnou úlohu. Který je lepší?
  - Stará a nová verze téhož nástroje. Je ta nová lepší?
- Důležité: porovnávané programy testovat na totožných datech!
  - Jinak je porovnání k ničemu, leda by rozdíl činil desítky procent.
- Je rozdíl v úspěšnosti dvou programů *statisticky významný*? Různé statistické testy, např.
  - McNemarův test pomocí χ^2
  - Wilcoxonův test aj.

Statistická významnost

- Příklad
  - Starý tagger dosahoval úspěšnosti 91,37 %, nová verze dosahuje 91,41 %.
  - Je to skutečné zlepšení?
  - Nebo je to jen náhoda a na jiných datech bude lepší ten starý tagger?
- Test statistické významnosti
  - Má nám říct, zda zlepšení byla jen náhoda
  - Přesněji: zkontrolovat, že pravděpodobnost, že to byla jen náhoda, je menší než nějaké malé číslo, typicky 0,05 nebo 0,01
  - Potom říkáme, že zlepšení je „statisticky významné na úrovni 0,05“

Primitivní test konzistence

- Rozdělíme testovací data na *N* dílů, třeba 10.
- Oba klasifikátory vyhodnotíme na každém dílu zvlášť.
- Pokud je klasifikátor 2 lepší než klasifikátor 1 na všech dílech, pak asi zlepšení není náhodné.

Wilcoxonův test pořadí se znaménkem

- Vstupem jsou sady spárovaných hodnot X_a a X_b (úspěšnost systémů *a* a *b* na vzorcích dat)
- Pro každou dvojici se zjistí absolutní hodnota rozdílu $|X_a–X_b|$.
- Nepřihlíží se k případům, kdy $|X_a–X_b| = 0$.
- Zbývající absolutní rozdíly se seřadí od nejmenšího k největšímu. Případné shodné rozdíly se dělí o pořadí.
- K *pořadí* každého rozdílu se přiřadí znaménko +, jestliže $X_a–X_b > 0$, jinak znaménko –.
- Takto oznaménkovaná pořadí se posčítají a výsledkem je hodnota *W*. Počet oznaménkovaných pořadí značíme *n*.
- Jestliže je $n \geq 10$, vzorkovací rozdělení *W* je rozumně blízkou aproximací normálního rozdělení. V takovém případě pokračujeme ve výpočtu podle vzorce. Pro menší vzorky ($n = 5$ až 9) se s vypočítaným *W* musí jít do tabulky kritických hodnot $\pm W$.
- Pro větší vzorky spočítáme směrodatnou odchylku $\sigma = \mathrm{sqrt}(n\times(n+1)\times(2n+1)/6)$.
- Spočítáme tzv. *poměr Z*, $Z = (W–0{,}5)/\sigma$.
- Z tabulky kritických hodnot *Z* zjistíme pravděpodobnost, že je změna úspěšnosti pouhým dílem náhody. Např. je-li *Z* větší než 1,96, pak tato pravděpodobnost není větší než 0,05.

McNemarův test

- V počítačové lingvistice oblíbený
  - Ale má široké statistické uplatnění, např. při zkoumání účinnosti nového léku
- Vytvoříme si *kontingenční tabulku* 2×2
  - Sloupce reprezentují stav „před podáním léku“
  - Řádky reprezentují stav „po podání léku“
  - Máme
    - 2 řádky (+ a –, „zdravý“ a „nemocný“)
    - 2 sloupce (taky + a –)
  - 4 buňky
    - *a* (+ před, + po), *b* (– před, + po), *c* (+ před, – po) a *d* (– před, – po)
  - Do buněk vyplníme počet pacientů odpovídajících popisu
  - Analogie: počet slov, kterým starý tagger dal špatnou značku a nový tagger správnou

Kontingenční tabulka

	Dříve +	Dříve –	Součet
Nyní +	$a = 101$	$b = 59$	160
Nyní –	$c = 121$	$d = 33$	154
Součet	222	92	314

McNemarův test

- Tabulka je homogenní, jestliže $b = c$
  - Po podání léku se ulevilo stejnému počtu pacientů, jakému se přitížilo, takže lék na tuto nemoc nemá vliv.
- Jak velký rozdíl mezi b a c už je statisticky významný?
- Jestliže b a c nejsou příliš malá ($b+c$ aspoň 20), můžeme naše rozdělení pravděpodobností aproximovat rozdělením χ^2 *[chí kvadrát, kaj skvér]* a významnost otestovat McNemarovým testem.
- $\chi^2 = (b-c)^2/(b+c)$
  - V našem případě $\chi^2 = (59-121)^2/(59+121) = 3844/180 = 21{,}4$
- Pro ne moc velká b a c ($b+c \leq 30$) se ještě provádí tzv. korekce nespojitosti: $\chi^2 = (|b-c|-1)^2/(b+c)$
  - V našem případě $\chi^2 = (|59-121|-1)^2/(59+121) = 3721/180 = 20{,}7$
  - Ale není to nutné, máme dost velká b a c
- Výsledek porovnáme s tabulkou prahových hodnot rozdělení χ^2. Např. jestliže nám vyšlo $\chi^2 \geq 3{,}84$, je rozdíl mezi měřením „dříve“ a „nyní“ statisticky významný na hladině 0,05.