ČEŠTINA MEZI ČLOVĚKEM A POČÍTAČEM

Vilém Novák, Svatopluk Zdebski

1. Úvod

Dovolte úvodem tuto citaci: "Už S. Čech zjistil prostřednictvím pana Broučka, že na Měsíci se mluví česky. Také v pohádce Hrnečku vař je právě čeština prostředkem k dorozumění s divotvornou nádobou. Proč by čeština neměla být jedním z prvních jazyků, se kterým budou pracovat samočinné počítače?". Těmito větami začíná kniha kolektivu českých autorů "Učíme stroje česky" [5].

V loňském roce jsme zde hovořili o parametrických programech a otázkách parametrizace. Snažíme se dělat programy efektivnější a efektivněji. Píšeme strukturovaně, normalizovaně, popisujeme je nebo generujeme na základě rohodovacích tabulek a děláme všechno možné, abychom měli méně práce a více zábavy. Je však také zábavou používání našich produktů pro ty, pro které jsou určeny, tedy pro uživatele? Na to ať si odpoví každý sám. A každý sám ať si dá do souvislosti úvodní citaci s našimi pochybnostmi.

O čem chceme hovořit? V souladu s názvem příspěvku o možnostech (a přednostech i těžkostech) použití češtiny (ovšem jako jednoho z reprezentantů přirozených jazyků) při styku uživatele s našimi produkty. A v rozporu s názvem článku o přejímání obecných atributů přirozeného lidského vyjadřování (tedy nejen v češtině) jako je vágnost, nejednoznačnost a komolení jak při porozumění uživateli, tak při tvorbě programů.

2. Přirozený jazyk jako prostředek dotazování

Naši situaci můžeme charakterizovat následujícím způsobem: Stávající informační systémy disponují určitou základnou informací. Známe více nebo méně efektivní algoritmy výběru informací z různě organizovaných datových základen. Otázky uspořádání a manipulace s daty si my, programátoři, musíme vyřešit sami (a vlastně pro sebe). Otázku věcné náplně datové základny nám vyřeší informační a systémoví inženýři. Jakkoliv jsou tyto otázky složité, jsou pro vlastního uživatele podružné. Předpokládejme (možná ještě dnes neoprávněně), že uživatel ví, co se

chce dozvědět. Jak však má formulovat svůj požadavek? Nutíme jej učit se naše "geniálně jednoduché a úplné" parametrické systémy a divíme se, že je uživatel odmítá. A opomíjíme základní skutečnost, že jediným úplným a pro uživatele nejjednodušším systémem je jemu daný mateřský přirozený jazyk.

Podle nás je nejednoznačný (ale náš systém interpretující dotaz v přirozeném jazyce nemusí být pasivní; aktivně si může vyžádat upřesnění). Podle nás je jeho analýza složitá (ale stejně složitá se zdála našim předchůdcům i analýza cobolských a algolských výroků). Podle nás je příliš bohatý (ale operujeme nad výsekem reality, což zužuje i množství potřebných výrazových prostředků). Je možno uvést řadu dalších argumentů proti a postupně je vyvracet.

Jsme programátoři a mělo by nás zajímat, jak na to: kde čerpat základy pro naši práci s přirozeným jazykem, jak interpretovat dotazy a kde hledat inspiraci, poučení a příklady použití.

Existuje dnes poměrně široká literatura o komunikaci v přirozeném jazyce. Proces analýzy přirozeného výroku je zajímavý z hlediska své algoritmizace a způsobu uložení našich znalostí o jazyce a jeho prvcích. Obecně probíhá analýza ve třech etapách, přičemž konečným cílem je překlad výroku v přirozeném jazyce do formalizovaného zápisu v určitém přesně definovaném kalkulu (zpravidla logickém), který budeme nazývat metavýrokem a který je náš programový aparát (metainterpretátor) schopen jednoznačně dekódovat. Protože sémantická náplň datové základny je známá, je schopen najít v ní požadované informace.

Morfologická analýza provádí analýzu jednotlivých výskytů slovních forem ve větě (dotazu), tj. identifikuje slovní základ, zkoumá přípustnost konkrétního tvaru a přiřazuje každé slovní formě morfologické charakteristiky (slovní druh, čas, číslo apod.). Fáze syntaktické analýzy zkoumá vztahy mezi jednotlivými slovními formami. Přípustné vztahy jsou popisovány podobně jako u běžných umělých jazyků pomocí jazyka produkcí nebo syntaktických stromů. Specifické je to, že v první fázi se hledají všechny možné způsoby rozkladu věty na tzv. jmenné skupiny (predikáty, kompletivy, příslovečná určení, kvalifikátory, deklarativy apod.). Pro každý způsob rozkladu se zkoušejí všechny uplatnitelné produkce, které vedou k zjednodušení. Výsledkem je buď úplný metavýrok pracující pouze s funktory,

kvantifikátory, predikáty, proměnnými a konstantami, nebo neúplný metavýrok obsahující neinterpretovatelné slovní výrazy. Ve druhém případě zkoušíme jiný rozklad na jmenné skupiny, a to tak dlouho, dokud se nedospěje k úplnému metavýroku nebo "nejasnosti". V poslední fázi - sémantické analýze - se konstruuje sémantický graf výroku a provádí se jeho strukturalizace. Vlastní realizace dotazu je pak interpretací metavýroku a sémantického grafu výkonnou částí systému.

Nebudeme uvádět příklady reálně fungujících zahraničních systémů. I u nás již lze rozmlouvat s počítačem v češtině. V Praze v ÚVTEI byl pracovníky MFF UK Praha implementován systém KODAS, který je schopen zodpovídat víceméně složité dotazy nad datovou základnou týkající se pracovníků MFF. Základem je obecná metoda převodu dotazu v přirozeném jazyce do formálního jazyka blížícího se predikátovému kalkulu. Vlastní interpretace je závislá na konkrétní databázi, nad kterou se operuje, ale to již není problém nepřekonatelný. Pevně daná struktura slovníku (i jeho produkční části) a procedurní způsob uložení by i v našich omezenějších podmínkách měl umožnit reálné použití tohoto systému. Pokud půjde vše dobře, chtěli bychom se pokusit o implementaci tohoto systému v jiných podmínkách, než ve kterých vznikal. A tak tedy příště budeme o tomto problému schopni napsat více konkrétních zkušeností.

3. Přibližné prohledávání seznamů klíčových slov

Využití přirozeného jazyka při komunikaci s počítačem je velmi perspektivní a podle našeho názoru povede k výraznému zvýšení uživatelského komfortu a tím k častějšímu vyhledávání služeb počítače uživateli. Srovnáme-li však tuto komunikaci s komunikací mezi lidmi, zjistíme jeden výrazný rozdíl: počítač není schopen tolerovat žádné chyby nebo je velmi omezeně. Napíšeme-li např. PODNAK místo PODNIK, pak člověk je schopen tuto chybu tolerovat, zatímco počítač nám oznámí, že takové slovo nezná. Jedním z prostředků, pomocí něhož lze také u počítače dosáhnout velmi uspokojivé tolerance takovýchto chyb, je tzv. teorie fuzzy množin (čti fazi). Jejím cílem je matematicky modelovat význam vágních pojmů. Tato teorie se v současné době ve světě bouřlivě rozvíjí a má řadu zajímavých aplikací.

Obecně je fuzzy množina A funkce

$$A: U \longrightarrow \langle 0, 1 \rangle ,$$

kde U je množina nazývaná univerzum a číslo $A_x \in \langle 0, 1 \rangle$ se nazývá stupeň příslušnosti prvku $x \in U$ do fuzzy množiny A. Na rozdíl od klasické množiny je zde jednoznačné rozhodnutí o patření či nepatření prvku $x \in U$ do dané množiny nahrazeno číslem z intervalu $\langle 0, 1 \rangle$ vyjadřujícím pouze míru jeho náležení do ní. Je-li A fuzzy množina v univerzu U, pak píšeme $A \subseteq U$.

Vraťme se nyní k problému tolerance chyb. Nechť Σ je abeceda, Σ^* množina všech slov nad touto abecedou a $U \subseteq \Sigma^*$ je zadané množina slov (seznam slov, mezi nimiž chceme najít to správné). Zadáme-li slovo $v \in \Sigma^*$, pak existuje fuzzy množina $A^{(v)} \subseteq U$ všech slov podobných slovu v. Stupeň příslušnosti $A^{(v)}u \in \langle 0, 1 \rangle$, kde $u \in U$ lze interpretovat jako stupeň podobnosti slova u slovu v. Je několik způsobů, jak stupeň příslušnosti $A^{(v)}u$ určit.

Maximální společný řetězec (MSŘ) je souvislý řetězec písmen délky > 2 společný oběma slovům u, v. Označme c(u, v) délku sjednocení všech MSŘ. Pak stupeň příslušnosti $A^{(v)}u$ pro dané slovo $u \in U$ lze nejjednodušeji definovat takto:

$$A^{(v)}u = \frac{c(u, v)}{|u|}$$

kde |u| označuje délku slova u. Blíže o této metodě - viz [7].

Autoři příspěvku se zabývali realizací uvedené metody v podmínkách budování informačních systémů. Výsledkem je víceméně "obecný" podpůrný prostředek umožňující nejrůznější aplikace zaměřené jak směrem k terminálovým uživatelům informačního systému, tak k jeho projektantům.

Uživatelské aplikace uvedené metody vidíme ve dvou směrech:

- upřednostnění otevřeného označování objektů výběru v , interaktivních dotazových systémech s pevnou strukturou dotazu. Tolerují se jak zjevné chyby při zadání, např. HRUBA TEZBA ≃ HRBA TEZBA ≃ HRUBA TZEBA, tak chyby vyplývající z nejednoznačnosti označování, např. DUL 9. KVETEN ≃ PODNIK 9. KVETEN ≃ 9. KVETEN .
- realizace rychlých a účinných prohledávacích funkcí deskripčních schémat přiřazujících otevřenému názvu (podniku, závodu, ukazatele, výrobku) jeho kódové označení. V dosavadních aplikacích, kdy základním výrazovým prostředkem uživatele byl kód, šlo o prohledávání kód→text. Uvedená metoda zrychlí prohledávání text→kód, takže uživatel disponující obecnou znalostí vnějšího označení

objektů popisovaných informačním systémem se snadno seznámí s vnitřním (strojovým) označením objektu. Metoda opět toleruje chyby zadání a nejednoznačnosti.

Projekční aplikace lze spatřovat v budování programovacích prostředků umožňujících toleranci chyb zadání identifikátorů, datových struktur, prvků a klíčových slov a usnadňující řešení problému synonymie (IF, KDYZ, JESTLIZE, KDYBY). Programovací jazyk si vytvoří deskripční předlohu jednak z pevných programových slov (každému slovu je přiřazen vnitřní jednoznačný symbol) a z označení identifikátorů z deklarační části. Při zpracování procedurální části programu se každá lexikální jednotka porovnává s předlohou a vybírá se nejpodobnější ekvivalent. Cílem nejbližší doby je přebudovat na tomto principu generátor konverzních programů používaný v rámci našich informačních systémů. Jde o prostředek se syntaxí příbuznou COBOLu, používaný správou báze dat (tedy neprogramátory).

Vlastní modul má tyto parametry:

- funkce FRST, NEXT podle toho, zda požadujeme nejpodobnější nebo další slovo,
- deskripční předloha (seznam zadaných slov se spřaženými atributy,
- testované slovo,
- pole pro odpověď.

Modul byl realizován a ověřen na počítači IBM/370 a připravujeme realizaci na SM-4.

4. Vyhledávání informací ve volném textu

Protože přirozený jazyk (čeština) je nejužívanějším prostředkem pro zaznamenávání informací všeho druhu, byl na MFF UK vyvinut systém automatizovaného vyhledávání informací z úplného textu AZIMUT. Tento systém pracuje s úplnou morfologickou analýzou češtiny. Jeho podstata je následující:

Na vstupu je volně psaný text, který je rozčleněn běžným způsobem na hlavy, části, články, odstavce. Nejprve se z něho na základě tzv. negativního slovníku vyloučí všechna nadbytečná slova, např. "je, jsou, který, on apod.", která jsou velmi frekventovaná a nepřinášejí žádnou informaci. V další fázi se vytvoří tzv. konkordanční slovník, tj. každé slovo, které nebylo v první fázi vyloučeno, se uloží společně s adresami všech

jeho výskytů v rámci výše zmíněného členění. Dotazy na výskyt určité informace lze formulovat volně. V dotazu jsou vyskloňována slova, která se mohou vyskytovat v různých tvarech, např. "hrubá výroba", "hrubou výrobou", "hrubé výrobě" apod. a prohledává se konkordanční slovník. Na výstupu se vypisují adresy všech výskytů informací, na něž byl položen daný dotaz.

Možnosti využití tohoto systému jsou poměrně široké. Vyskytují se např. poměrně rozsáhlé seznamy víceslovních názvů určitých informací (např. názvů hospodářských ukazatelů). Jejich vizuální prohledávání při počtu několik tisíc je již prakticky beznadějné.

Jiné využití je např. v tom, že uživatel mající k dispozici displej by si mohl vést svůj vlastní záznamník obsahující nejrůznější informace např. o výsledcích porad, diskusí, důležitá rozhodnutí aj. Tyto informace by si mohl psát volným textem a k jejich prohledávání použít popsaný systém. Čtenáře jistě napadnou další možnosti jeho využití.

Nespornou výhodou tohoto systému je úplná automatizace zpracování, tj. není nutné předzpracování zkušeným odborníkem (např. volba klíčových slov apod.). Chtěli-li bychom umožnit uživateli dokonce určité omyly a ještě větší volnost zápisu, bylo by možné uvažovat o rozšíření tohoto systému o prostředek přibližného prohledávání řetězců popsaný v článku 3.

5. Programovací jazyky s lingvistickým rozšířením

Poslední oblastí, o níž se v tomto příspěvku zmíníme, jsou programovací jazyky. Dosavadní programovací jazyky nepřipouštějí žádnou míru neexaktnosti, která je vlastní přirozeným jazykům. Anglická slova, která se v nich vyskytují, mají zcela přesně a jednou provždy definovaný význam. Teorie fuzzy množin zmíněná ve třetím článku však nabízí možnost modelovat sémantiku slov a celých výrazů v přirozeném jazyce včetně určitého podchycení jejich vágnosti. Na jejím základě se začínají objevovat projekty tzv. programovacích jazyků s lingvistickým rozšířením, pomocí nichž lze snáze na počítači implementovat tzv. fuzzy algoritmy. S takovými algoritmy se setkáváme na každém kroku, např. při řízení auta, vaření jídla, chůzi po ulici aj. Jejich hlavním rysem je porušení požadavku determinovanosti, tj. jejich výsledkem je přibližné řešení specifikovaného problému. Vyskytují se v nich příkazy např. "několikrát přičti pět",

"mírně zvyš teplotu", "jestliže jsou obrátky stroje nízké, pak hodně pootoč regulačním knoflíkem" apod.

Existující verze programovacích jazyků s lingvistickým rozšířením jsou zpravidla vytvářeny na základě existujících jazyků. V Politechnickém institutu v Rize v SSSR byl vyvinut jazyk FAGOL (Fuzzy AlGOrithmic Language), který je lingvistickým rozšířením FORTRANu [3]. Ve Francii je podobným rozšířením jazyka PL/1 jazyk L.P.L. (linguistic Oriented Programming Language) [1, 2]. Další podobné jazyky jsou FUZAL, FLOU a umělý jazyk pracující s fuzzy množinami FUZZY.

Realizace těchto jazyků je ovšem složitější, neboť některé manipulace s fuzzy množinami jsou dosti složité. Např. pro provádění fuzzy algoritmů je typický určitý paralelismus, kdy se "do určité míry" provádějí všechny větve programu současně. Jazyky s lingvistickým rozšířením musí mít v sobě navíc zabudováno počítání s lingvistickými výrazy, aritmetiku tzv. fuzzy čísel, tj. speciálních fuzzy množin v reálné ose, provádění výkonných fuzzy příkazů a některé specifické příkazy, které v běžných programovacích jazycích nejsou nutné nebo možné.

Vlastní fuzzy množiny jsou zpravidla definovány v deklaracích, např. (jazyk L.P.L.):

```
DLP VYSOKY BY (10)0,1,2,3,4,5,7,8,9,10,10;
```

kde DLP je výrok definující lingvistický predikát VYSOKY, jehož význam je fuzzy množina se stupni příslušnosti 0, 1, ..., 10 (místo čísel v intervalu ⟨0, 1⟩). Univerzum je deklarováno prostřednictvím tzv. základní proměnné, např.

```
B_V VYSKA SCALE(50);
```

Podmíněné příkazy lze psát např. takto:

```
IF X=VYSOKY THEN příkaz 1; ELSE příkaz 2;
```

Speciálním příkazem, který v klasickém PL/1 není nutný, je např. příkaz PARALLEL definující současné provádění několika podmíněných příkazů. Bez tohoto příkazu by záleželo na pořadí zápisu podmíněných příkazů, což je nežádoucí.

Rozvoj programovacích jazyků s lingvistickým rozšířením je teprve na svém začátku, a proto není dostatek zkušeností s jejich používáním. Zdá se však, že by mohly být účinně aplikovány např. v umělé inteligenci při řešení problémů rozpoznávání situace, učení se, porozumění přirozenému jazyku, sofis-

tikované manipulaci s daty v databázi apod. Lze je chápat jako začátek cesty, na jejímž konci bude možné programovat roboty přímo v přirozeném jazyce (např. česky) bez jakékoliv formalizace.

6. Závěr

V závěru se obvykle konstatuje to, co bylo řečeno. My jsme však něco zamlčeli. Náš příspěvek je totiž provokace. Máme dojem, že my (tj. programátoři) začínáme být výlučnou skupinou uzavřenou do sebe a stále více se uzavírající. Kdosi loni v kuloárech prohlásil, že by byla na místě selekce účastníků a tedy žádný analytik mezi nás! Dopustit přítomnost uživatelů by byl asi ještě daleko větší přečin. Tak je trochu suplujeme (ale hovoříme i za svou profesi).

A ještě jedna kacířská myšlenka: Programovat se učíme podle pana Jacksona, Dijkstry, Dahla, Wirtha a dalších. Prostředky k tomu nám dodávají všemocné firmy IBM, ICL, DEC atd. Ale tu češtinu, tu si budeme muset vyřešit sami. A to nejenom proto, že mluvíme jinak, než jinde ve světě, ale asi také jinak myslíme.

LITERATURA

[1] Adamo, J.M.: L.P.L. A Fuzzy Programming Language: 1. Syntactic Aspects. Fuzzy Sets and Systems 3, 1980, s. 151-179.

[2] Adamo, J.M.: L.P.L. A Fuzzy Programming Language: 2. Semantic Aspects. Fuzzy Sets and Systems 3, 1980, s. 261-289.

[3] Borisov, A.N., Alekseev, A.V., Krumberg, O.A., Merkučeva, F.V., Popov, V.A.: Modeli prinjatija rešenij na osnově lingvističeskoj peremennoj. Riga, Zinatne 1982.

[4] Dubois, D., Prade, H.: Fuzzy Sets and Systems. Theory and Applications. Academic Press, New York 1980.

[5] Hajičová, E., Pitha, P., Sgall, P.: Učíme stroje česky. Panorama Praha 1982.

[6] Popov, E.V.: Obščenije c EVM na jestěstvennom jazyke. Mir, Moskva 1983.

[7] Schek, H.J.: Tolerating Fuzziness in Keywords by Similarity Searchers. Kybernetes, 1977, s. 175-184.