PLIN013 Proseminář z počítačové lingvistiky

Miloš Jakubíček

Centrum zpracování přirozeného jazyka
Fakulty informatiky, Masarykova univerzita
jak@fi.muni.cz

December 12, 2012

Obsah

Cíl kursu

- získat základní orientaci v oboru ZPJ
- prohloubit poznatky z předmětů FI:IB000 a FF:PLIN004
- osvojit si elementární principy programování a práci v prostředí operačního systému Linux
- čas pro organizační záležitosti oboru a zpětnou vazbu od studentů

Požadavky k zápočtu

- přiměřená účast (max. 3 neomluvené absence)
- vyhotovení 2 průběžných úkolů a složení závěrečného testu

Motivace I – cíle ZPJ

- cílem počítačového zpracování přirozeného jazyka je zejména **objektivní** popis jazyka a jeho **formalizace** pro účely **praktických aplikací**
- pro úkoly v oblasti ZPJ je důležitá dobrá spolupráce mezi informatiky a lingvisty, z nichž obě skupiny by měly mít přiměřené vzdělání v druhém oboru: informatici základní povědomí o lingvistické teorii, soudobé kodifikaci a terminologii, lingvisté zejména schopnost formální (matematické) abstrakce a znalost principů algoritmizace/výpočtů.

Motivace II – mýty a pověry o informatice

- „informatika se zabývá především programováním“
- „informatik má umět dobře pracovat s aplikací X“
- „informatika je úzce svázána s posledními trendy v informačních technologiích“
- . . . dále viz předmět FI:IB000 Úvod do informatiky a další povinné a povinně-volitelné předměty oboru

ZPJ – přehled

- Jedna (avšak nikoli nutně jediná) možná základní klasifikace:
  - zpracování textu ($\rightarrow$ morfologie, syntax, sémantika, příp. logika)
  - zpracování řeči ($\rightarrow$ fonetika)
  - dialog ($\rightarrow$ diskurs, pragmatika)
- analýza (+ desambiguace) vs. syntéza
- formální vs. přirozené jazyky:
  - determinismus vs. víceznačnost
  - kontrolovaný vs. nekontrolovaný vznik a vývoj
  - malý počet mluvčích, z nichž většina má velmi odborné znalosti o jazyku vs. velký počet mluvčích, z nichž většina nemá odborné znalosti o jazyku žádné
  - ...a mnoho další vlastností přirozených jazyků, které formální jazyky zcela postrádají
- vše podrobněji v FI:IB030 Úvod do počítačové lingvistiky

ZPJ – stav pro češtinu

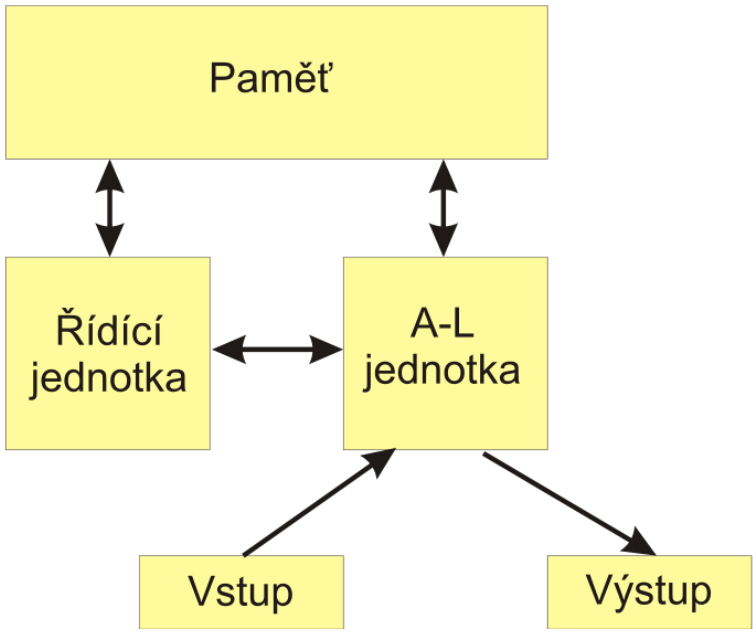
- v rámci ZPJ velmi intenzivně studovaný jazyk, v míře zpracování patří k celosvětové špičce, mnoho dostupných jazykových zdrojů
- pracoviště v Praze (ÚFAL MFF UK, ÚTKL FF UK, ÚČNK ÚJČ), Brně (CZPJ FI MU, Speech@FIT), Plzni (KKY ZČU), Liberci (LPZŘ FM TU)
- ukázky: wwwajka, wwwsynt

Co je počítač? I

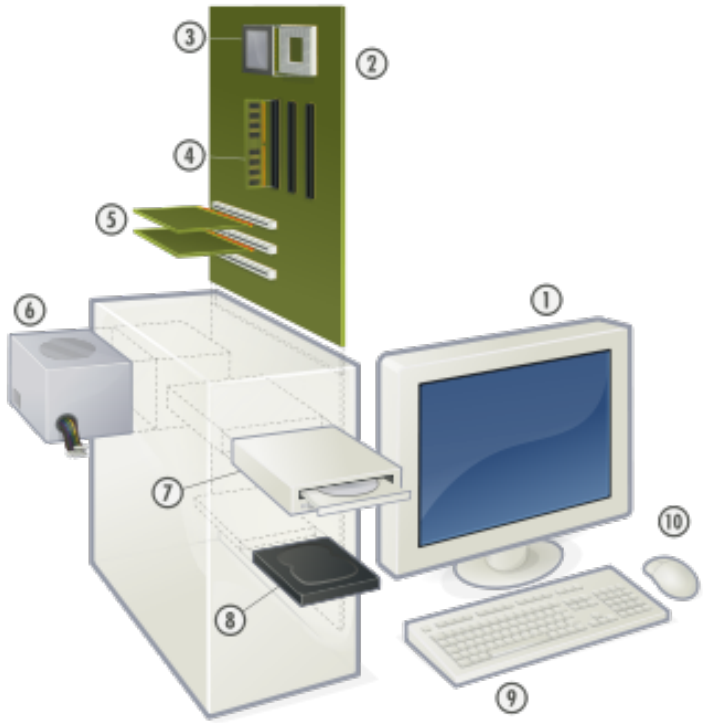
„Počítač je v informatice elektronické **zařízení**, které zpracovává **data** pomocí předem vytvořeného **programu**. Současný počítač se skládá z **hardware**, které představuje fyzické části počítače (procesor, klávesnice, monitor atd.) a ze **software** (operační systém a programy). Počítač je zpravidla ovládán uživatelem, který poskytuje počítači data ke zpracování prostřednictvím jeho **vstupních zařízení** a počítač výsledky prezentuje pomocí **výstupních zařízení**.“
(zdroj: Wikipedia)

Co je počítač? II

Von Neumannovo schéma:



Co je počítač? III



Soudobé počítače

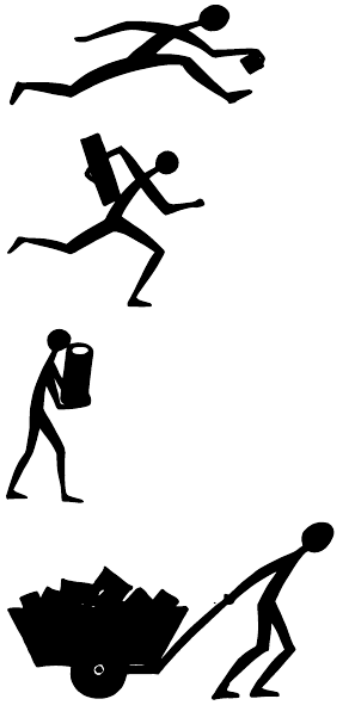
- digitální, binární informace
- základní jednotkou informace je 1 bit (b) (1/0 = pravda/nepravda = jde proud/nejde proud)
- v bitech (za jednotku času) je **zpravidla** udávaná přenosová rychlost
- velikost úložného prostoru **zpravidla** vyjadřována v bytech (B), 1 B = 8 b
- 1 TB = 1024 GB = 1024^2 MB = 1024^3 kB = 1024^4 B

Procesor

- frekvence
- instrukční sada
- počet jader
- velikost vestavěné paměti

Paměť

rychlost: CPU cache > RAM > HDD > páska, cena inverzně



Co je počítačový program? I

Posloupnost instrukcí (daným způsobem formalizovaná a tedy strojově zpracovatelná), která je počítačem vykonána:

```
f319 255c 3d1b 9b5c 1eb3 01aa 8bd4 2ea7
0003 0000 0070 0000 0002 0000 0007 0000
0180 2c1a 0900 4900 0548 0d08 0488 2000
0070 0000 0074 0000 0079 0000 8a10 234d
4dc8 2342 e3ba 7c92 4543 ecd5 41c8 9087
099c d632 2d46 dee4 71d8 1c58 8db9 0ef1
d3ea 0eef 043e 864b a9fc 79da ed1c 2a63
d42b d3b8 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
```

Co je počítačový program? II

Programy ale vytvářejí lidé, je tedy nutná i lidsky čitelná podoba zápisu ⇒ je potřeba překladač (kompilátor), příp. interpreter:

```
int main() {
    printf("Hello world!");
}
```

⟶

```
f319 255c 3d1b 9b5c 1eb3 01aa
0003 0000 0070 0000 0002 0000
0180 2c1a 0900 4900 0548 0d08
0070 0000 0074 0000 0079 0000
4dc8 2342 e3ba 7c92 4543 ecd5
099c d632 2d46 dee4 71d8 1c58
d3ea 0eef 043e 864b a9fc 79da
d42b d3b8 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000
```

Co je počítačový program? III

Programovací paradigmata

- imperativní (nejčastěji používané): program je vykonání posloupnosti příkazů
- funkcionální: program je redukcí výrazu
- logické: program je „dokazovač“ logických formulí

Servery CZPJ

- server: aurora.fi.muni.cz
- přihlašovací jméno a heslo shodné s přihlášením na FI
- operační systém Linux (viz FI:PV004 UNIX)

Práce v shellu I

= inteligentní „příkazový řádek“

- adresářová struktura vždy začíná od `/`, žádné `C:`!
- uživatelské adresáře v `/home/<login>`
- příkazy pro pohyb v adresářové struktuře a její výpis: `cd` – change directory, `ls` – list
- `cd ..`, `cd -`

Práce v shellu II

- základní příkazy pro manipulaci s textem: `echo`, `cat`, `less`, `wc`, `grep`, `sort`, `uniq`, `cut`, `exit`, `man`
- práce se standardním vstupem a výstupem (`>`, `<`, `|`), velikost písmen hraje roli!
- základní příkazy pro manipulaci se soubory: `mv`, `cp`, `rm`, `touch`
- jednoduchý textový editor: `nano`
- popis a další informace v češtině: http://www.abclinuxu.cz/ucebnice/prehled-prikazu

Hrátky v shellu

- zpracovávání tzv. vertikálního textu – prvních 10 tis. slov z korpusu DESAM
- `cp /home/xjakub/PLIN013/desam10k.vert muj.vert`
- `less muj.vert`
- `sort muj.vert | less`
- `sort muj.vert | uniq -c | less`
- `cut -f 1 muj.vert | sort | uniq -c | sort -rn | less`

nástroj grep I

- `g/re/p` v editoru `ed` = globally find regular expression `re` and print
- RE = regular expression = regulární výraz
- regulární, ne regulérní
- důležité souvislosti s matematickou lingvistikou a teoretickou informatikou (regulární výrazy vs. regulární gramatiky vs. konečné automaty – bude v PLIN004), zde se zaměříme na praktické použití v programu `grep`
- obecně: jde nám o konečný popis nekonečných (matematických) struktur (např. slov)

nástroj grep II

- . = lib. znak, [zur] = výčet prvků, [a-zA-Z0-9] = výčet prvků rozsahem
- * = lib. počet výskytů, včetně nultého (= znak není)
- \+ = alespoň jeden výskyt, \? = nejvýše 1 výskyt
- \{m\} = právě *n* výskytů, \{m,n\} = *m* až *n* výskytů
- ^ = začátek řetězce, $ = konec řetězce
- dále např.: http://www.cyberciti.biz/faq/grep-regular-expressions/
- http://stts.se/egrep_for_linguists/egrep_for_linguists.html

nástroj grep III

- `grep zkouška muj.vert`
- `grep k1 muj.vert`
- `grep "k2.*c3" muj.vert`
- `cut -f 2 muj.vert | grep "^a" | sort | uniq -c | sort -rn`

Kódování a znakové sady I

- počítače „rozumí“ pouze číslům, pracují v binární soustavě
- jak ukládat znaky/text?
- co je soubor typu „čistý text“ (plain text)?
- jak se ukládá mezera/tabulátor/nový řádek?
- znaková sada = zobrazení (funkce) ze znaků do čísel
  `(např.: 'a' = 1, b = '2', c = '3', ...)`
- kódování = způsob uložení čísla
  `(např.: 1 = 0001, 2 = 0010, 3 = 0011, ...)`

Kódování a znakové sady II

- existují různé znakové sady a různá (vzájemně nekompatibilní) kódování
- společným základem je většinou 7bitové kódování ASCII
- pro češtinu dříve zejména znakové sady ISO LATIN2 (ISO-8859-2) a Windows-1250 (rozdíly v š, ť a ž)
- dnes snaha rozšiřovat znakovou sadu Unicode, nejčastěji v kódování UTF-8

Domácí úkol

- napište posloupnost příkazů, pomocí které získáte pět nejčastějších morfologických značek pro podstatná jména obsažená v korpusu DESAM taková, že začínají na první písmeno vašeho příjmení (bez ohledu na velikost znaku – pro Novák hledejte jak ta, která začínají na 'n', tak i na 'N')
- soubor s vertikálním textem se nachází na `aurora:/home/xjakub/PLIN013/desam.vert`
- způsob a forma odevzdání: v jednom textovém souboru uveďte na prvním řádku váš příkaz a na následujících pěti řádcích nalezené morfologické značky; takový soubor vložte do odevzdávárny v ISu
- hodnocení: 0–20 bodů (celkově 20/20/60 – 1. úkol, 2. úkol, test)
- **termín: 21. 11. 2012 včetně**

Programovací jazyk Python

- interpretovaný jazyk vyšší úrovně, multiplatformní
- automatická správa operační paměti
- důraz kladen na jednoduchost, čitelnost, povinné odsazování kódu
- http://docs.python.org

Programovací jazyk Python II

- spuštění interpreteru: `python`
- zkuste použití „jednoduché kalkulačky“: přiřazení (`proměnná = hodnota`), základní aritmetické operace
- výpis hodnoty proměnné: buď `print` nebo i přímo název proměnné

```
>python
Python 2.5.1 (r251:54863, Jul 31 2008, 22:53:39)
[GCC 4.1.2 (Ubuntu 4.1.2-0ubuntu4)] on linux2
Type "help", "copyright", "credits" or "license" for more i
>>> a=0
>>> a+2
2
>>> c=2*a-2
>>> c
```

Python – datové typy

- dynamické typování
- `int, float`: celá čísla – 15, desetinná čísla – 15.575
- `str, unicode`: řetězec (v daném kódování) – "ahoj", Unicode řetězec – u"ahoj"
- `list, tuple, set, dict`:
  seznam – [0, 1, "ahoj"],
  n-tice – (0, 1, "ahoj"),
  množina – set([0, 2, 2]) = set([0, 2]),
  slovník – {"jméno": "Miloš", "příjmení": "Jakubíček"}
- `True, False`: logická pravda a nepravda
- `None`: prázdný datový typ („nic“)
- objekty (ve smyslu OOP – objektově-orientovaného programování)

Python – operátory

- operátor = symbol dané operace (funkce)
- operand = argument operátoru
- `(2 + 5) * 7`
- operace na libovolných datových typech
- `"Dobrý" + " " + "den" = "Dobrý den"`
- `"den" * 2 = "denden"`

Python – základní konstrukty

- podmínečný příkaz:

```
if a == 1:
    print "Ahoj"
else:
    print "Nazdar"
```

- cyklus `while`:

```
while a == 1:
    print "Ahoj"
    a = 2
```

- cyklus `for`:

```
for a in ["ahoj", "nazdar", "čau"]:
    print a
```

Python – OOP v kostce I

- OOP = objektově orientované programování
- objekt = datový typ reprezentující nějakou reálnou entitu ve světě
- každý objekt má **atributy** (vlastnosti objektu) a **metody** (funkce, které s objektem manipulují)
- příklad objektu: mikrovlnná trouba
  - atributy: barva, rozměry, příkon, výkon, výrobce, ...
  - metody: otevřiDvířka(), zavřiDvířka(), ohřej(minuty=2)

Python – list

```
seznam = [0, 1, 2, "Ahoj", 7.8]
```

- výpis *n*-tého prvku: `print seznam[n]`
- nejdůležitější metody:
  - přidání nové hodnoty `x`: `seznam.append(x)`
  - rozšíření o jiný seznam `s`: `seznam.extend(s)`
  - odebrání hodnoty `x`: `seznam.remove(x)`
  - zjištění počtu prvků s hodnotou `x`: `seznam.count(x)`
  - dále http://docs.python.org/tutorial/datastructures.html#more-on-lists

Python – str

`s = "Tohle je řetězec"`

- výpis *n*-tého znaku: `print s[n]`
- → řetězec si lze představit jako seznam znaků
- nejdůležitější metody:
  - `find(podretezec[, zacatek[, konec]])` – test, zda řetězec obsahuje jiný podřetězec
  - `startswith, endswith` – test, zda řetězec začíná nebo končí na jiný řetězec
  - `split([oddelovac[, maximum]])` – rozdělí řetězec na seznam podřetězců podle zadaného oddělovače
  - `join(seznam)` – spojí prvky seznamu pomocí řetězce
  - dále http://docs.python.org/library/stdtypes.html#string-methods

Python – čtení ze standardního vstupu I

```
import sys
for line in sys.stdin:
    print line
```

- Který unixový příkaz náš program z části napodobuje?
- Napište program, který bude načítat ze standardního vstupu celá čísla a pro každé z nich vytiskne jeho dvojnásobek.
- Napište program, který bude načítat ze standardního vstupu celá čísla a ukládat je do seznamu. Po ukončení vstupu (přes Ctrl+D) program vypíše, kolikrát se každá hodnota v seznamu vyskytuje (využijte datový typ `set` – množina).

2. domácí úkol I

- Ve studijních materiálech předmětu (složka Učební materiály) naleznete skript `suffix.py`, který jsme vytvořili během semináře.
- Znovu si projděte program `suffix.py` vytvořený během přednášky: úkolem programu bylo vypsat pro frekvenci četnosti 1–3 znakových přípon.

2. domácí úkol II

Upravte `suffix.py` tak, aby:

- vypisoval odděleně statistiku pro jedno-, dvou- a tříznakové přípony
- ekvivalentně zpracoval i statistiku pro jedno-, dvou- a tříznakové předpony vstupních slov
- nezahrnoval bílé místo (mezeru, tabulátor, nový řádek, ...) jako součást předpon/přípon
- ignoroval struktury v korpusu (tj. řádky začínající na „<“)
- nevypisoval všechny položky, ale vždy pouze 10 nejčastějších, přesně tak jako ukázkový výstup (viz dále)

2. domácí úkol III

Jak si zkontrolovat, že můj program dělá to, co má?

- Ve studijních materiálech máte také soubor `desam.out` představující korektní výstup Vašeho programu – můžete tedy porovnat svůj a správný výstup: uložte si výstup svého programu do souboru (`python mujskript.py < in > out`), porovnání můžete provést např. pomocí příkazu diff, příp. jeho nadstaveb Kompare/K3Diff (GUI aplikace).
- Jako vstup Vašemu programu použijte celý korpus DESAM (nejen jeho desetitisícovou podčást, kterou jsme často používali v hodinách), který je dostupný v aurora:/home/xjakub/PLIN013/desam.vert
- Termín: upravený skript (soubor .py) odevzdevejte opět do odevzdávárny v ISu do 4. 1. 2013 včetně.

2. domácí úkol IV

- Diskuse k úkolu: nejlépe na předmětovém fóru v ISu