Technická dokumentace LANSHER

Vilém Zouhar

2017

1 Úvod a použité nástroje

LANSHER slouží k rozlišení jazyka na základě velkého množství trénovacích dat. Je naprogramován v jazyce Python3. Tato verze je programem striktně vyžadována, aby nedošlo k neočekávanému chování. Využívá se běžně dostupných knihoven, konkrétně **json, argparse, re** (regex) a **pickle** (serializace).

2 Fungování programu

Program je založený na tvorbě *slovníku* ze vstupních definic jazyka. Ze vstupu na rozeznání se vytvoří také *slovník* jazyka. Není tedy rozdíl v porovnávání dvou jazyků a rozeznávání vstupního textu (porovnání se všemi jazyky v databázi).

Termínem *slovník jazyka* je myšleno přiřazení slova **sl** k počtu výskytu $\mathbf{freq_{sl}}$. Pro větu *Alenka tváří tvář Alence vyslovila: "Jmenuji se Alenka"* se jedná o zobrazení:

alenka	$\rightarrow$	*2*
tváří	$\rightarrow$	*1*
tvář	$\rightarrow$	*1*
alence	$\rightarrow$	*1*
vyslovila	$\rightarrow$	*1*
jmenuji	$\rightarrow$	*1*
se	$\rightarrow$	*1*

Informace o interpunkčních znaménkách a velikosti písmen je pro jednoduchost srovnávání zahozena (vhodným regulérním výrazem). V programu je objekt *slovníku* implementován za pomocí *dictionary*, což má operace přidání prvku, získání prvku průměrně v konstantním čase.

Pro srovnání dvou jazyků, například výše uvedené věty (S_l) a českého překladu *Alenka v říši divů* (S_{cs}), první získáme množinu všech vzorů vyskytující se v obou *slovnících* (M) a první část jejich podobnosti určíme vztahem:

$$|S_1 S_{cs}|' = \sum_{i=1}^{|M|} \; log(S_1(M_i) \cdot S_{cs}(M_i) + 1)$$

Je to tedy logaritmus vzájemného násobku četnosti dvou klíčů, pro všechny klíče vyskytující se v obou množinách. Člen $+1$ byl zvolen kvůli tomu, že v případě $S_1(m) = S_{cs}(m) = 1$ je funkční hodnota logaritmu 0, tedy chceme pozitivně ocenit i jeden společný výskyt.

Četnosti se násobí, aby se získala větší podobnost při větší míře výskytu. Logaritmus je použití proto, aby se odfiltrovaly patologické případy. Jeden je demonstrován na dalším případě:

S_{en} (slovo *to* je v angličtině zastoupeno hojně):

to	→	*31*
..	→	..
..	→	..

S_{cs}:

to	→	*5*
není	→	*4*
pravda	→	*2*
..	→	..
..	→	..

S_2 (věta *"To není pravda!"*):

to	→	*1*
není	→	*1*
pravda	→	*1*

Byť má slovník angličtiny (zpracovaný na základě *Alice in Wonderland*) s větou společné jen jedno slovo, má velkou četnost. Český slovník má definované všechny tři slova ve větě, ale ta mají menší četnost. V případě bez logaritmů by se tedy věta nekorektně rozpoznala jako anglická. Využíváme toho, že logaritmus je funkce rostoucí, avšak s klesající derivací ($\frac{1}{x}$).

Další krajní případ může nastat tehdy, když databázi přetrénujeme. Tedy samotný jazyk bude mít definovaná slova, která bychom do něj běžně nezařadili. V trénovacím textu pro češtinu se může vyskytnou věta *"V němčině 'guten Tag' znamená 'dobrý den'"*. Výslední slovník by mimo jiné obsahoval i slova pro *guten, tag*. Součet podobností tedy vydělíme počtem definovaných slov v jazyce, čímž budeme slovník penalizovat za svoji rozsáhlost.

Podobnost dvou slovníků, $|S_1 S_2|$, definujeme jako:

$$|S_1 S_2| = \frac{\sum_{i=1}^{|M|} log(S_1(M_i) \cdot S_{cs}(M_i) + 1)}{log(|S_1||S_2| + 1)}$$

Kromě podobnosti slovníků se k podobnosti jazyků používá (byť v řádově menší míře) i porovnání četnosti znaků, stejným způsobem.

3 Struktura programu

Celý program je napsaný v souboru *lansher.py*. K jeho spuštění je zapotřebí Python3. Před vlastním programem je definována řada funkcí, přičemž všechny fungují průměrně v *O(n)* čase.

- *clean_data*

  Vyčistí vstupní řetězec od interpunkčních znamének a převede ho do lowercase.

- *element_frequency*

  Vrací objekt typu *dictionary*, kde funkční hodnota každého klíče je jeho četnost ve vstupním poli.

- *join_frequencies*

  Slouží k sloučení dvou slovníků jednoho jazyka

- *create_lang_object*

  Vytvoří, popřípadě sloučí, slovní a znakové slovníky jazyka. Také vypočítá počet slov ve slovsnících. Odkazuje se na *join_frequencies*.

- *add_to_database*

  Řídící funkce odkazující se na *create_lang_object*. Přidává nově vytvořený jazyk do databáze programu.

- *add_lang_file*

  Vytvoří objekt jazyka na základě vstupního souboru (JSON). Obsahuje validaci vstupu.

- *add_lang_database*

  Vytvoří objekt jazyka na základě vstupního souboru již dřívě použitého a uloženého jazyka. Obsahuje validaci vstupu.

- *save_lang_database*

  Uloží aktuální databázi programu do souboru za pomocí serializace *pickle*.

- *distance_langs*

  Vypočítá vzdálenost dvou vstupních jazyků, je možné parametrizovat *WORD_CHAR_RATIO* tedy poměr váhy četností slov a znaků.

- *compare_against_database*

  Porovná vstupní jazyk se zbytkem databáze programu a zobrazí výsledky v procentech.

- *podmínka na konci programu*

  Obsahuje zpracování přepínačů a řídí celý program.

Program je ošetřen oproti neplatným vstupům, je možné používat různé přepínače a zejména nápovědu **-h**.

4 Reálná funkčnost

S programem je dodávaná zpracovaná databáze 10 jazyků z různých překladů knihy *Alenka v říši divů*. Změřme tedy dobu zpracovávání databáze a její následné ukládání:

```
time ./lansher.py -ls alice_full.json  -sd alice_full.ld
```

Výstup:

```
real    0m0.478s
user    0m0.448s
sys     0m0.024s
```

Tedy půl vteřiny. Vstupní JSON má ∼2.3MB, zatímco zpracovaná databáze ∼0.8MB Pokud chceme použít ji, je na tom čas mnohem lépe:

```
time ./lansher.py -ld alice_full.ld -i "Byl pozdní večer - první máj.."
```

Výstup:

```
cs (59.713%)
es (6.132%)
fr (5.333%)
pl (5.242%)
no (5.124%)
de (4.966%)
en (4.824%)
it (4.803%)
ja (3.715%)
ru (0.149%)

real    0m0.066s
user    0m0.060s
sys     0m0.008s
```

Pokud se nejedná o příliš krátkou větu, která není reprezentativní pro daný jazyk, dokáže program ve většině případů jazyk rozeznat správně.