

# 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 **freq<sub>sl</sub>**. Pro větu *Alenka tváří tvář Alence vyslovila: "Jmenuji se Alenka"* se jedná o zobrazení:

<i>alenka</i>	→	2
<i>tváří</i>	→	1
<i>tvář</i>	→	1
<i>alence</i>	→	1
<i>vyslovila</i>	→	1
<i>jmenuji</i>	→	1
<i>se</i>	→	1

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

Pro srovnání dvou jazyků, například výše uvedené věty ( $S_I$ ) 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:

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

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  $S_I(m) = S_{cs}(m) = 1$  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 použit proto, aby se odfiltrovaly patologické případy. Jeden je demonstrován na dalším případě:

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

<i>to</i>	→	31
..	→	..
..	→	..

$S_{cs}$ :

<i>to</i>	→	5
<i>není</i>	→	4
<i>pravda</i>	→	2
..	→	..
..	→	..

$S_2$  (věta "*To není pravda!*"):

<i>to</i>	→	1
<i>není</i>	→	1
<i>pravda</i>	→	1

Byť má slovník angličtiny (zpracovaný na základě *Alice in Wonderland*) s větou společně 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 vyskytnout 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 i porovnání četností dvojic a trojic, stejným způsobem. Mnohem nižší váhu má četnost samotných znaků, která se tedy použije pouze v případě, že průnik slov, dvojic a trojic je příliš nízká.

### 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 slovnících. 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 dříve použitého a uloženého jazyka. Obsahuje validaci vstupu.

- *save\_lang\_database*

Uloží aktuální databázi programu do souboru za pomoci 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  $\sim 2.3\text{MB}$ , zatímco zpracovaná databáze  $\sim 0.8\text{MB}$ . 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.