Eliza v češtině

Lubomír Sedlář (359719)

20. května 2011

Abstrakt

Eliza je program, který dokáže vést s uživatelem jednoduchý rozhovor simulující chování psychoterapeuta. Původní verze fungující v angličtině vznikla v letech 1964–1966[1].

Cílem tohoto textu je shrnout změny potřebné pro úpravu programu tak, aby fungoval pro češtinu. Jako základ jsem použil implementaci v jazyce Python z knihovny NLTK[2].

1 Princip fungování

Základním prvkem programu je seznam obsahující dvojice (uživatelský vstup, reakce). Reakce, kterých může být více, jsou uloženy v seznamu a vybírá se mezi nimi náhodně. Vstupy jsou zadávané jako regulární výrazy, které umožňují zachytit část textu a použít ji v reakci.

Do odpovědí je možné přidat část uživatelského vstupu pomocí sekvencí `%n`, kde `n` odpovídá pozici požadovaného řetězce v regulárním výrazu.

Originální program potom zpracuje řetězec tak, aby tvořil gramaticky správnou větu, která má smysl, a vytiskne jej uživateli. Úpravy spočívají v převedení slov z první osoby do druhé a naopak z druhé osoby do první. V anglickém programu je tato funkce realizována pomocí jednoduchého slovníku.

2 Problémy v češtině

Pouhým přeložením reakcí z angličtiny do češtiny dostaneme program, který sice reaguje na uživatelský vstup, ale jeho reakce mají do správně gramaticky utvořených vět hodně daleko. Úprava do použitelného tvaru je ale výrazně složitější než v angličtině. Narazil jsem na tři hlavní problémy, které je možné programově elegantně řešit. Mezi obtížně řešitelné problémy patří třeba možnost

[1] `http://en.wikipedia.org/wiki/ELIZA`
[2] `http://www.nltk.org/`

rozlišovat tykání nebo vykání (program vždy vyká, ale slyší na obě možnosti) nebo úprava vět podle pohlaví uživatele (program mluví k muži).

2.1 Reflexe sloves

Reflexe pomocí slovníku je v češtině možná pouze pro zájmena. Přestože i slovesa svádí k použití jednoduché metody na základě koncovek, tento přístup nevede k cíli. Např. slova *chceš*, *chci* vyzývají k pravidlu koncovku *-eš* měnit na *-i*. To ale nefunguje třeba v případě *nebudeš*.

Zohledňovat kromě koncovek i část kořene by vedlo k velice komplikovanému programu. Jednodušší a čistší princip je použít morfologický analyzátor. Já jsem použil analyzátor Majka, který pro slovo určí lemma a značku nebo pro lemma a značku vrátí příslušný tvar.

Větu určenou k vypsání uživateli tedy stačí slovo po slovu zpracovat analyzátorem a pokud se jedná o sloveso v první nebo druhé osobě, upravit patřičným způsobem značku, společně s lemmatem předat analyzátoru a získané slovo vložit do věty.

2.2 Slovosled zvratných sloves

Pokud se v části vstupu, který bude použit v odpovědi, vyskytuje zvratné sloveso, výsledná věta má ve většině případů po jednoduchém spojení dost nepřirozený slovosled, jako např. *Potřebuji se dostat domů. Proč potřebujete se dostat domů?*. Stejná situace nastává s některými zájmeny.

Obecné řešení tohoto problému by asi bylo algoritmicky značně náročné. Protože ale odpovědi jsou předem známé, stačí sestavit seznam slov, která vedou k tomuto problému a ten k opravě slovosledu využít.

Programu tedy stačí hledat v odpovědi tato „problémová“ slova a pokud jsou následovaná zvratnou částicí nebo zájmenem (opět uloženo v seznamu), vzájemně slova prohodit. Je třeba myslet na to, že může být nezbytné v jedné větě prohodit více slov – *Když už tedy máte se dobře.* → *Když už se tedy máte dobře.*.

2.3 Shoda podmětu s přísudkem

Posledním vážnějším problémem, na který jsem narazil, je potřeba upravovat rod slovesa v předdefinované odpovědi tak, aby odpovídal podmětu dodanému z uživatelova vstupu. Jako příklad poslouží věta *Kdyby to bylo X*, kde *pes* jako X vyžaduje jiný tvar slovesa než třeba *kočka*.

Oprava tohoto jevu opět není univerzální, ale pracuje přímo s reakcemi z programu. Najde první podstatné jméno v části textu od uživatele, zeptá se analyzátoru na jeho rod a příslušně podle něj pozmění sloveso.

2.4 Nejednoznačná morfologická analýza

Pro naprostou většinu slov analyzátor Majka vrací více možností pro lemma a značku. Protože jednotlivé možnosti zřejmě nejsou nijak seřazené, výběr vždy první nebo naopak poslední možnosti vede k chybám na některých slovech. Proto značku a lemma z možností vybírám náhodně. To sice nezabrání chybám, ale odstraní to situace, kdy některé slovo nikdy není správně rozpoznáno.

3 Instalace

Pro používání programu stačí rozbalit archiv se zdrojovým kódem na libovolné místo a spustit příkazem `python eliza.py`. Jediným předpokladem správného fungování je přítomnost morfologického analyzátoru Majka v adresáři `/nlp/projekty/ajka` s podadresáři `bin/` se spustitelným souborem `majka` a podadresářem `lib/` s datovými soubory `majka.w-lt` a `majka.lt-w`.

V dodaném programu je v souboru `majka.py` ve funkci `run_majka` definovaná proměnná `path` obsahující cestu ke kořenovému adresáři analyzátoru. V případě potřeby je možné ji změnit.

Přestože program vychází ze zdrojových kódů knihovny NLTK, tato knihovna není při používání potřeba.

4 Příklad použití

Řádky začínající znakem `>` zadává uživatel, zbytek jsou reakce programu.

```
Dobrý den. Jak se dnes cítíte?
> Mám se výborně.
A když už se tedy máte výborně, co budete dělat dál?
> Chtěl bych něco sníst.
Proč byste chtěl něco sníst?
> Mám chuť na něco dobrého.
Opravdu máte chuť na něco dobrého?
> Ano, vždyť to říkám.
Dobře, ale můžete to trochu rozvést?
> Potřebuju něco sladkého.
Proč potřebujete něco sladkého?
> Nemůžu se bez toho soustředit.
Jak byste se cítil, kdybyste se mohl bez toho soustředit?
> konec
Děkuji, bude to 1450 korun. Přeji pěkný den!
```