Emoční klasifikace textu

Martin Tomáš

Obsah

1 Úvod

Cílem projektu bylo navrhnout systém, který na základě minulých znalostí rozhodne zda-li zadaný text je kladně nebo záporně emočně zabarvený.
Pro řešení tohoto problému byl navržen systém v jazyce Java využívající dva statistické modely (Bernouliho a Multinomiální). Dále byly implementovány metody pro správu a získání potřebných dat z internetu, které posloužily pro naučení a otestování aplikace. Pro vývoj byly využity knihovny H2 a Jsoup a aplikace Majka.

2 Instalace

Aplikace je navržena v jazyce Java a tudíž nevyžaduje žádnou instalaci ani kompilaci (aplikace je k dispozici i v spustitelném .jar souboru). Bohužel část programu využívá software Majka (licence Creative Commons Attribution-ShareAlike 3.0 Unported) pro získání slov v základním tvaru, který je platformně závislý. K aplikaci je přibalen program Majka kompilovaný pro 64 bitový operační systém Linux. Aby byla zajištěna správná funkčnost celé aplikace je nutné program spouštět na této platformně (případně zkompilovat program Majka na svém stroji a ten poskytnout aplikaci).
Systém dále využívá knihovny H2 (licence MPL 1.1) a Jsoup (MIT licence).
K aplikaci nejsou přibaleny datové soubory (data stažená z webu Heureka a Srovname) a naučené znalostní báze. Toto řešení bylo zvoleno z důvodu značné velikosti těchto souborů (dohromady přes 1GB). V případě spuštění aplikace bez těchto dat si sice aplikace příslušné soubory vytvoří, ale nebude schopna správně identifikovat emoční zabarvení textu (pokud aplikace nemá žádné informace vhodné pro určení zabarvení, vždy hádá, že vstupní text je negativně zabarvený). V takovémto případě je nutno pustit nejdříve parsery a následně proces učení, aby si aplikace vytvořila příslušnou znalostní bázi a mohla efektivně určovat zabarvení vstupního textu.

3 Běh aplikace

Aplikace po svém startu provede několik činností (viz Obrázek 1):

1. inicializuje databáze
2. otáže se jestli chce uživatel stahovat data z internetu ⇒ start parserů
3. otáže se jestli chce uživatel zahájit učení ⇒ start lokálního parseru
4. otáže se jestli chce uživatel zahájit testování ⇒ start lokálního parseru s referencí na testovací objekt (obsahuje loggovací soubor)
5. pokud uživatel nechce zahájit testování, je mu nabídnuta možnost zadat vstupní text do konzole, ten je následně vyhodnocen a je zobrazena informace o jeho emočním zabarvení
6. ukončí databáze

Veškerá komunikace s aplikací probíhá v české jazyce přes textový terminál.

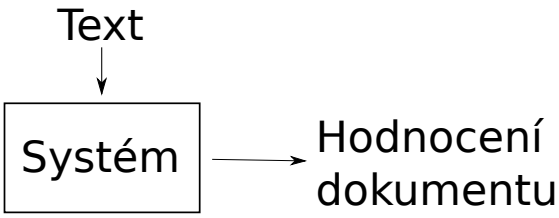
4 Cíle

Zamýšlené chování systému ukazuje Obrázek 2. Aplikace má na základě vstupu, zadaného uživatelem nebo v případě testování získaného automaticky z databáze, rozhodnout

```
Nactena db: heureka-local
Nactena db: srovname-local
Chcete zahajit stahovani dat z Internetu do lokalni db (n|Y)?n
PRESKAKUJI STAHOVANI DAT Z INTERNETU
Nactena db: memory
Chcete zahajit uceni z lokalnich dat heureky (n|Y)?n
PRESKAKUJI UCENI
Chcete zahajit testovani (n|Y)?n
PRESKAKUJI TESTOVANI
Zadejte vstupni text (konec textu ukoncete zadanim tecky na prazdem radku):
To je ale dneska krásně.
.
Text je podle Bernouliho modelu pozitivní
Text je podle Mulinomialniho modelu pozitivní
Byla ukoncena db: heureka-local
Byla ukoncena db: srovname-local
Byla ukoncena db: memory
```

Obrázek 1: Ukázka běhu aplikace

zda-li je text kladně nebo záporně zabarvený. Systému v tomto bodě nejsou poskytnuty žádné další informace kromě samotného textu. Rozhodování je tedy realizováno pouze s využitím znalostí báze vytvořené postupným učením na označkovaných příkladech (obsahují informaci o svém zabarvení).
Výstupem je čtveřice pravděpodobností, každá pro jeden klasifikační model a pro jeden emoční případ (2∗2). Tato čtveřice je systémem vhodně interpretována a uživateli je zobrazena pouze informace jak je text zabarven.



Obrázek 2: Vstupy/výstupy aplikace

5 Zvolený princip

I když se můžeme snažit text rozdělit podle mnoha charakteristik a do mnoha skupin (např. spam nebo nospam), mnou navržená aplikace využívá rozdělení pouze na dvě emoční skupiny (kladně a záporně zabarvený text). Toto rozdělení bylo zvoleno z důvodu jednoduchého získání dostatečného množství příkladu na kterém je možné aplikaci naučit toto rozdělení realizovat.
Z hlubšího pohledu je toto rozdělení realizováno pomocí **naivního Bayese**, který dovoluje zkoumat slova v jednotlivých větách bez dalších pravděpodobnostních vazeb na zbytek obsahu. V průběhu učení tedy ztrácíme další informace skryté ve větě (např. syntaxi) a omezujeme se pouze na výskyty jednotlivých slov v různě zabarveném textu. Tento jed-

A co že máš tak velké oči babičko?

	pozitivní	negativní
velké	2	1
máš	5	4
oči	6	3
babičko	7	2

četnost výskytu

Obrázek 3: Charakteristika slov ve větě

noduchý princip demonstruje Obrázek 3, z kterého je patrné, že aplikace si v znalostní databázi udržuje informace o četnostech slov v různě zabarveném textu. Příkladem tohoto

principu může být jednoduché zjištění, že např. slovo „smrdí“ bude s velkou pravděpodobností použito pouze v záporně zabarveném textu. Pokud v nějaké větě objevíme takovéto slovo, můžeme s velkou pravděpodobností rozhodnout, že text nebude pozitivní. Tyto informace lze tedy poměrně dobře využít pro samotné rozhodování nad neznámým textem.

5.1 Klasifikační modely

Pro klasifikaci textu byly zvoleny dva klasifikační modely. Oba dva využívají jiné pravděpodobnostní rozložení a jejich výstupy se mohou proto v některý případech značně lišit. Použité modely jsou:

- Multinomiální model
  - bere v úvahu vícenásobný výskyt slova v textu
  - vhodnější pro delší texty
  - samotný výpočet je realizován podle vzorců:

$$P(w|c) = (\text{četnost}+1)/(\text{počet slov}+\text{velikost slovníku}),$$
$$P(c|d) = P(c) * P(w_1|c)...P(w_n|c) \text{ a } d = \{w_1, ..., w_n\},$$

    kde d je vstupní text složen ze slov w_i a c je příslušná kategorie (kladný nebo záporný text)

- Bernoulliho model
  - nebere v úvahu vícenásobný výskyt slova v textu
  - vhodnější pro kratší texty
  - samotný výpočet je realizován podle vzorců:

$$P(w|c) = (\text{výskyt}+1)/(\text{počet dokumentů}+\text{počet případů}), \text{ kde výskyt} \in \{0, 1\},$$
$$P(c|d) = P(c) * P(w_1|c) * ... * P(w_n|c) * (1 - P(t|c)),$$
$$\text{kde } d = \{w_1, ..., w_n\} \text{ a } t = \text{Slovník} - d \text{ (všechna zbylá slova)}$$

6 Zdroj dat

Aby aplikace mohla dostatečně přesně charakterizovat text, bylo nutné zvolit dostatečně rozsáhlý, ale snadno získatelný text. Pro tento účel byly zvoleny dva weby, **Heureka a Srovname**. Tyto weby obsahují dostatečné množství snadno emočně charakterizovatelného materiálu v komentářích a recenzích pro jednotlivé výrobky a nebo obchody. Aplikace tedy obsahuje automatizované rutiny, které dovolují stahovat a ukládat tento obsah pro budoucí snadné zpracování při samotném učení nebo testování.

7 Implementace

Celý systém je rozdělen na několik více či méně nezávislých částí. Těmi jsou:

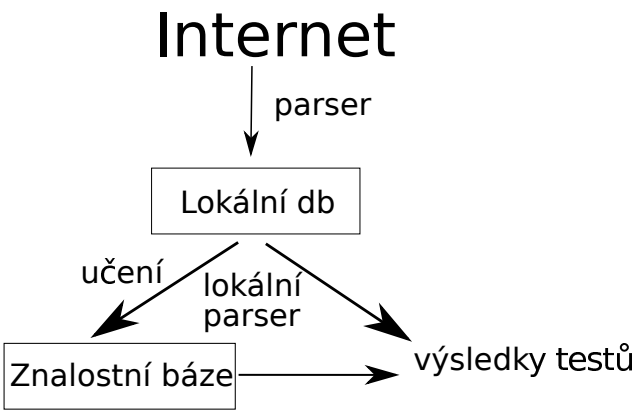
1. Parser – obsahuje celkem tři parsery, každý pro jeden web a jeden pro lokální data použit pro testování a učení
2. Datová vrstva – obsahuje třídy pro správu lokálních dat a znalostní báze
3. Klasifikační modul – je složen z algoritmů pro výpočet jednotlivých pravděpodobností charakterizující text a tříd pro interpretaci výsledků
4. Testování – jedná se o nezávislou část aplikace, jejíchž účelem je vyvolat testování a jeho loggování

7.1 Parser

Tento modul slouží k získání všech potřebných dat z internetu a ke konstrukci znalostní báze. Učení je ze svého pohledu velmi podobné samotnému procházení a získávání dat z internetu, pouze se zde jinak zachází se získaným textem. Z tohoto důvodu je učení a testování realizováno stejně jako samotné parsování, pouze jsou příslušné parsery obaleny novým objektem, který slouží k přesměrování získaných dat (návrhový vzor Dekorátor).
Parsery stahující data z internetu získávají pevně zadanou počáteční adresu, další adresy dále získávají postupným procházením. Bylo nutné zajistit, aby parsery nejen neopustily procházené weby, ale aby neprocházely příliš velké množství nadbytečných stránek. Z toho důvodu každý parser obsahuje kolem 20 regulárních výrazů, které filtrují nevhodné url odkazy.
Proces stahování dat z internetu jsou úlohy velmi dobře paralelizovatelné (pouze musíme zajistit, aby jednu stránku neprocházelo více vláken) a proto jsou parsery vybaveny metodami pro bezpečné spuštění několika vláken najednou.

7.2 Datová vrstva

Vrstva pro správu a ukládání dat byla rozdělena na dvě samostatné podvrstvy, jejichž spolupráci blíže popisuje Obrázek 4. Lokální datová vrstva obsahuje data přímo stažená z



Obrázek 4: Datová vrstva aplikace

internetu. Tato vrstva je zcela nezávislá na použitém modelu a při jakékoliv hlubší změně v samotném programu lze na jejím základě zrekonstruovat znalostní bázi aplikace. Implementačně tato databáze obsahuje dvě tabulky – url odkazy a klasifikované texty.
Druhou datovou vrstvou je znalostní báze dat. Tato vrstva už je přímo závislá na použitém modelu a ukládá informace o četnostech v jednotlivě emočně zabarvených dokumentech. Každý z modelů počítá výsledné četnosti rozdílně a proto tabulka četností obsahuje čtyři sloupce. Dále pak tato databáze obsahuje počet kladně a záporně ohodnocených dokumentů, které byly použity k jejímu vytvoření.

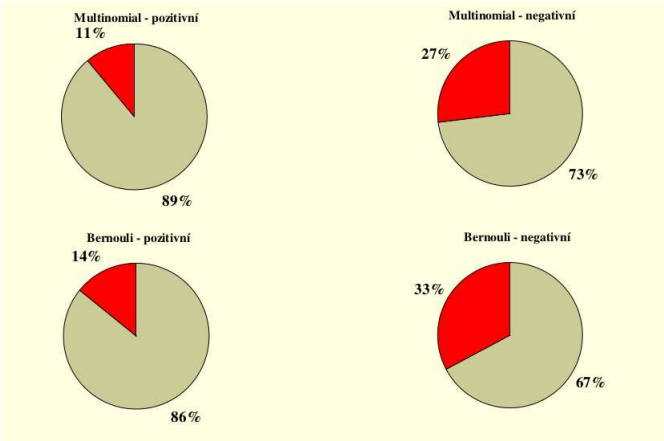
7.2.1 Zpracování textu

Každý text prochází před uložením do databáze určitým předzpracováním. Prvotní text, který je ukládán do lokální databáze je uložen v co nejsurovější podobě, pouze je zajištěno, aby nemohl svým obsahem ohrozit zbytek databáze (SQL injection, atd.).
Texty použity pro vytvoření znalostní báze procházejí dalším zpracování. Prvotně je celá věta rozdělena na jednotlivá slova a symboly (např. vítr! je rozdělen na dvě slova a to vítr a !). Po rozdělení následuje odstranění všech nadbytečných mezer. Takto vzniklé slovo prochází programem Májka, z kterého je získáno slovo v základním tvaru. Pokud takové slovo

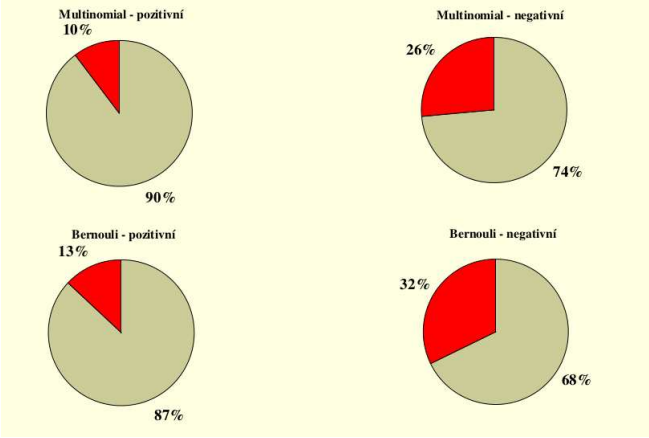
nebylo nalezeno, je použito slovo původní. Ke každému konkrétnímu slovu je zjištěna informace kolikrát je v textu obsaženo. Tento seznam prochází na výstupu ještě dodatečným filtrem, který odstraňuje STOP slova (ty nebudou v znalostní databázi použita).

7.3 Klasifikační modul

Tento modul obsahuje nejdůležitější část celé aplikace a slouží k zjištění jaký má text emocionální zabarvení pro ten či onen model. Aby bylo zajištěno co nejvyšší zapouzdření oproti zbytku aplikace, je modulu poskytován pouze textový řetězec. Výsledek výpočtu je vracen v zapouzdřeném enumerátoru, který v případě potřeby transformuje výsledek do lehce srozumitelné podoby. Výpočty jsou prováděny nad logaritmickými hodnotami, aby nedošlo k akumulaci chyby při velkém počtu násobení malých hodnot.
Klasifikační algoritmy využívají hojně metody poskytované znalostní bází pro získání potřebných dat pro samotný výpočet. Chyby při výpočtu jsou zachytávány a zobrazovány na uživatelův terminál.



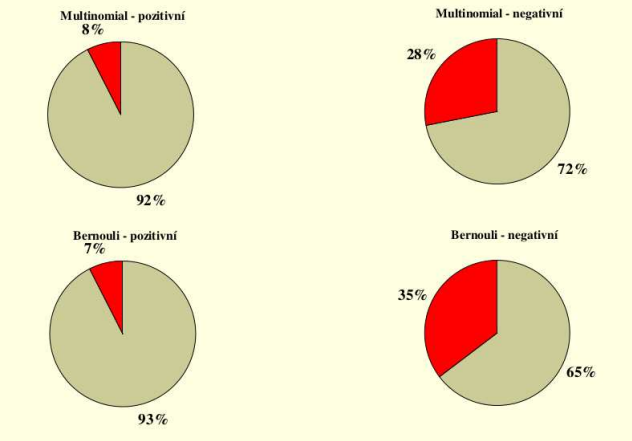
Obrázek 5: Použity data z webu Srovname a program Majka



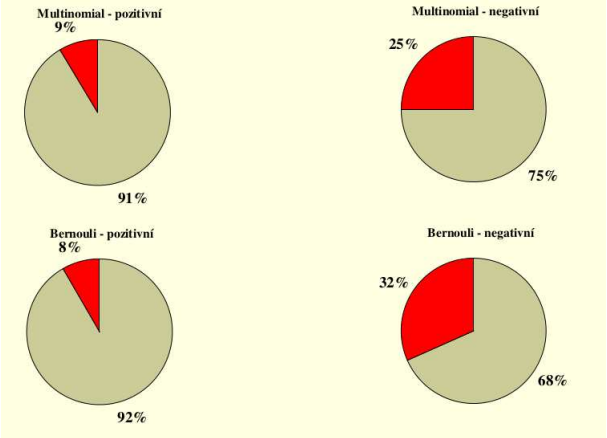
Obrázek 6: Použity data z webu Srovname bez programu Majka

7.4 Testování

Implementace testování byla do aplikace integrována s co největším důrazem na minimální ovlivnění zbylé části systému. Pro samotné testování byl využit lokální parser (stejný jako pro učení). V tomto případě mu je ovšem poskytnuta reference na testovací objekt,



Obrázek 7: Použity data z webu Heureka a program Majka



Obrázek 8: Použity data z webu Heureka bez programu Majka

který poskytuje dodatečný prostor pro ukládání výsledků testování a loggování celého procesu. Podle této reference pozná lokální parser, že text z lokální databáze nemá sloužit k učení, ale k testování a přesměruje získané data do klasifikačního modulu. Výsledky výpočtu jsou potom poskytnuty testovacímu objektu pro další zpracování (interpretace a uložení výsledků).

8 Výsledky testování

Pro testování byla zvoleno toto rozdělení dat:

- web Heureka (celkem: 2 017 017) – 70% použito pro učení, 30% pro testování
  - pozitivních příkladů bylo: 1 357 535 (67%)
  - negativních příkladů bylo: 659 472 (33%)
- web Srovname (celkem: 51 622) – všechny data použita pro otestování aplikace
  - pozitivních příkladů bylo: 31 925 (64%)
  - negativních příkladů bylo: 19 697 (36%)

Aplikace bylo otestována s použitím i bez použití programu Majka. Z časových důvodů nakonec nebylo testování realizováno na všech 30% datech. Aplikace bylo otestována na

Model	**Prec**	**Rec**	**Acc**	F_1**-measure**
Multinomiální (+majka)	92.37%	87.00%	85.65%	89.61%
Bernoulliho (+majka)	92.56%	84.19%	83.38%	88.18%
Multinomiální	91.28%	86.56%	85.38%	88.86%
Bernoulliho	91.72%	83.65%	83.27%	87.50%

Tabulka 1: Web Heureka

Model	**Prec**	**Rec**	**Acc**	F_1**-measure**
Multinomiální (+majka)	89.02%	84.31%	82.96%	86.60%
Bernoulliho (+majka)	85.97%	80.93%	78.79%	83.37%
Multinomiální	89.60%	84.67%	83.53%	87.06%
Bernoulliho	86.88%	81.33%	79.56%	84.02%

Tabulka 2: Web Srovname

577 636 příkladech s použitím programu Majka a na 160 598 bez použití tohoto programu. Výsledky testování pro web Srovname jsou zobrazeny na následujících obrázcích (obr. 5 a 6). Nejlepších výsledků dosáhl Multinomiální model bez použití programu Majka a to 83,53% při hádání emočního zabarvení testovaného textu.
Výsledky testování pro web Heureka jsou zobrazeny na Obrázcích 7 a 8. Nejlepších výsledků v tomto případě dosáhl Multinomiální model s použitím programu Majka a to 85,65% při hádání emočního zabarvení testovaného textu.
Ze všech výsledků testování je dále patrné, že Bernoulliho model na testovaných datech vracel horší výsledky než Multinomiální (a to hlavně pro negativně zabarvený text). Dále pak oba dva modely měly znatelně horší úspěšnost nad negativně zabarveným textem. Tato situace mohla být zčásti způsobena větším množstvím pozitivních příkladů ať pro učení, tak pro testování a určitou schizofrenií negativních komentářů (mnoho uživatelů do negativní recenze vkládalo nejen negativní hodnocení, ale pokud byly s produktem (obchodem) spokojeny i pochvalné hodnocení, např. „Nic mne nenapadá, s produktem jsem byl naprosto spokojen.“).

8.1 Zhodnocení testování podrobněji

Pro objektivní zhodnocení výsledků testování a tedy schopností aplikace předpovídat správné zabarvení textu byly zvoleny metriky Acc, Prec, Rec a F-measure, kde $Acc = \frac{tp+tn}{tp+fp+fn+tn}$, $Prec = \frac{tp}{tp+fp}$, $Rec = \frac{tp}{tp+fn}$ a F_β-measure $= \frac{(\beta^2+1)*Prec*Rec}{\beta^2*Prec+Rec}$. Proměnné tp, fp, tn a fn označují pozitivní příklad uhodnutý aplikací, pozitivní příklad neuhodnutý aplikací, negativní příklad uhodnutý aplikací a negativní text neuhodnutý aplikací. Vypočtené metriky ukazují přehledně tabulky 1 a 2.

9 Závěr

Projekt si kladl za cíl vytvoření softwarového řešení pro předpověď emočního zabarvení textu. Tento cíl se podařilo realizovat. Vytvořená aplikace je schopna automaticky získat data z prostředí Internetu, vytvořit si na jejich základě znalostí databázi a dále nezávisle na dalších vstupních datech klasifikovat zadaný text.
Pro aplikaci byly vytvořeny testovací rutiny a program byl otestován na značném množství příkladů (více jak 0.5 mil.). Z výsledků testování je patrné, že použité modely jsou schopny klasifikovat vstupní data s více jak 80% úspěšností.

Použitá literatura

[1] DVOŘÁK, Miloš, et al. Objekty – Vzory [online]. 27.05.2005, 03.02.2008 [cit. 2011-04-12]. Dostupný z WWW: <http://objekty.vse.cz/>.

[2] Pavel Šmerk. Fast Morphological Analysis of Czech. In Petr Sojka and Aleš Horák. Proceedings of Third Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2009. Brno: Masaryk University, 2007. p. 13–16. ISBN 978-80-210-5048-8.

[3] Christopher D. Manning, Prabhakar Raghavan a Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008. Dostupný z www <http://www-nlp.stanford.edu/IR-book/>.

[4] Jurafsky Dan, Manning Christopher, Natural Language Processing [online]. 15.05.2014 [cit. 2014-05-15]. Dostupný z www: <https://class.coursera.org/nlp/lecture/preview>.