Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Framework pro prezentaci NLP algoritmů

Plzeň 2016 Roman Zeleník

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 13. června 2016

Roman Zeleník

Abstract

This thesis focuses on web presentation algorithms for natural language processing, including collecting, storing and presenting statistics on the use of the website. It describes what natural language processing is, the field of the recovery and approach solution 3 selected tasks. It includes a design of framework for Javascript application for the transfer of data between the user and the computer server and design of the analysis tool for storing and presenting data on the use of the application. The suggested framework as well as an analysis tool were also implemented, including a simplified implementation of algorithms for solving 3 selected tasks in the field of natural language processing. To demonstrate the usage of the framework, and analysis tool were created website, including the application itself, which presents 3 selected tasks, allowing them to try out solutions and present the collected statistical data. At the end of this thesis you can find a summary of results including thumbnails parts created Web pages.

Abstrakt

Bakalářská práce se zaměřuje na webovou prezentaci algoritmů pro počítačové zpracování přirozeného jazyka včetně sbírání, ukládání a prezentaci statistických údajů o používání webové prezentace. V práci je popsáno co počítačové zpracování přirozeného jazyka je, oblasti jeho využití a přiblížení řešení 3 vybraných úloh. Práce obsahuje návrh frameworku pro vytvoření aplikace v jazyce JavaScript pro předávání dat mezi uživatelem a výpočetním serverem a návrh analyzačního nástroje pro ukládání a prezentování dat o používání aplikace. Navržený framework i analyzační nástroj byly dále implementovány, včetně zjednodušené implementace algoritmů řešících 3 vybrané úlohy z oblasti zpracování přirozeného jazyka. Pro demonstraci použití frameworku a analyzačního nástroje byly vytvořeny webové stránky, včetně samotné aplikace, které prezentují 3 vybrané úlohy, umožňují vyzkoušet jejich řešení a prezentují nasbíraná statistická data. Na konci bakalářské práce je shrnutí výsledků včetně náhledů částí vytvořených webových stránek.

Obsah

1 Úvod **3**

2 NLP **4**

2.1 Co je NLP 4

2.2 Čím se zabývá 4

2.2.1 Předzpracování a analýza textu 4

2.2.2 Zpracování psaného textu 7

2.2.3 Zpracování řeči 8

2.2.4 Další úlohy 9

2.3 Korpusy textů 10

2.4 Vybrané úlohy 10

2.4.1 Sumarizace 10

2.4.2 NER 11

2.4.3 Identifikace jazyka 11

3 Návštěvnost a zpětná vazba **13**

3.1 Google Analytics 13

3.1.1 Přehledy 13

3.1.2 Události 16

3.2 Vlastní „Analytics“ 17

4 Návrh **19**

4.1 NLP framework 19

4.1.1 Struktura 19

4.1.2 Rozšiřitelnost 20

4.1.3 Nastavení 20

4.2 Analytics 20

4.2.1 Interakce uživatele 20

4.2.2 Vizualizace dat 21

5 Implementace **22**

5.1 Použité technologie 23

5.1.1 Klient 23

5.1.2 Server 25
5.2 Adresářová struktura webu 25
5.2.1 Diagram 25
5.2.2 Obsah adresářů 25
5.3 Architektura NLP frameworku 26
5.3.1 Adresářová struktura 26
5.3.2 (M)VC architektura 27
5.3.3 Main 29
5.3.4 Komunikace a formát dat 29
5.3.5 Konfigurace, úpravy a rozšíření 31
5.4 Architektura Analytics 34
5.4.1 Adresářová struktura 34
5.4.2 Sbírání a ukládání dat 35
5.4.3 Prezentace dat 35
5.4.4 Příprava a interakce s daty 36
5.4.5 Main 36
5.4.6 Databáze 36

6 Dosažené výsledky **38**
6.1 Zkušební dema 38
6.2 Prezentace Analytics 43

7 Závěr **49**

Literatura **50**

A Instalace a sestavení **53**

B Uživatelská příručka **54**
B.1 NLP Demos (web) 54
B.1.1 Navigace 54
B.1.2 Ovládání dema 55
B.2 Analytics (přehled) 57

1 Úvod

Katedra informatiky a výpočetní techniky na Západočeské univerzitě v Plzni se mimo jiné zabývá Umělou inteligencí a počítačovým zpracováním přirozeného jazyka. Výzkumníci se zabývají několika úlohami a jejich algoritmickým řešením. Algoritmy běží na univerzitních serverech a jejich autoři by rádi prezentovali veřejnosti jejich funkčnost na webu a umožnili návštěvníkům vyzkoušet si algoritmy nad vlastními daty.

Cílem této práce je co nejlépe splnit požadavek těchto programátorů na vytvoření grafického webového rozhraní, které umožní uživateli zadat potřebná vstupní data a spustit zpracování algoritmem. Je kladen důraz na snadné přidání prezentace dalších algoritmů. Dalším požadavkem byla možnost zpětné vazby na výsledky daný algoritmů a přehled o jejich návštěvnosti či využití návštěvníky.

Algoritmy jsou prezentovány jako samostatná dema. Demo je v podstatě „single-page“ aplikace, která na pozadí může komunikovat s libovolným serverem, na kterém algoritmus běží a který dokáže zpracovávat `HTTP` požadavky.

Aplikace navíc dokáže monitorovat a ukládat informace o tom, jaký text návštěvník algoritmu zadal, jaké nastavil vstupní parametry, čas strávený přípravou či prohlížením výstupu algoritmu a jak jej ohodnotil. Všechna tato data se ukládají do relační databáze. Pro zpracování a prezentaci těchto dat vývojářům algoritmů byla v rámci práce vytvořena webová prezentace, která umožňuje interaktivní procházení a filtrování návštěv konkrétních dem.

2 NLP

2.1 Co je NLP

Počítačové zpracování přirozeného jazyka (Natural language processing - NLP) je obor na pomezí lingvistiky a informatiky (umělé inteligence), popř. též akustiky a dalších. Zkoumá problémy analýzy či generování textů nebo mluveného slova, které vyžadují určitou (ne absolutní) míru porozumění přirozenému jazyku strojem. [15]

Moderní NLP algoritmy jsou založeny na strojovém učení, zejména statistickém strojovém učení. Paradigma strojového učení se liší od většiny dřívějších pokusů o zpracování jazyka. Dřívější implementace úloh zpracování jazyka obvykle vyžadovala ruční kódování velkých souborů pravidel. Paradigma strojového učení vyzývá místo toho k použití obecných algoritmů učení, často založených na statistické inferenci (statistické indukci), automaticky se učit tato pravidla na základě analýzy velkých korpusů typických příkladů z reálného světa. [10]

2.2 Čím se zabývá

2.2.1 Předzpracování a analýza textu

Morfologie

Morfologie, neboli tvarosloví, je disciplína lingvistiky, která studuje strukturu slov. Popisuje, jak se slova skládají ze základních jednotek (morfémů), zabývá se ohýbáním (skloňování, časování) a pravidelným odvozováním slov. Morfémy jsou nejmenší jednotky jazyka, které mohou nést význam (kořeny, kmeny, kmenotvorné přípony, předpony, přípony, koncovky). Morfologie studuje vztahy mezi jednotlivými částmi slov. Z morfologického hlediska se slova dělí na ohebná (skloňování a časování) a neohebná. [7][15]

Příklady několika úloh

- Identifikace přirozeného jazyka (Native Language Identification) - Určuje jazyk předloženého textu.

- Segmentace vět (Sentence breaking) -
  Známý také jako desambiguace hranice věty (sentence boundary disambiguation). Hledá hranice vět v textu. Hranice jsou často označeny tečkami nebo jinými interpunkčními znaménky. Ta však mohou mít i jiný význam (např. označují zkratku slova).

- Segmentace slov (Word segmentation) -
  Rozdělí část souvislého textu na slova. V jazyce jako je např. angličtina je to velice triviální, protože slova jsou obvykle rozdělena mezerami. Nicméně některé písemné jazyky jako jsou čínština, japonština nebo thajština takto slova neoddělují a segmentace textu v těchto jazycích je významnou úlohou vyžadující znalost slovní zásoby a morfologie daného jazyka.

- Morfologická segmentace (Morphological segmentation) -
  Rozdělení slov do jednotlivých morfémů a identifikace třídy morfémů. Obtížnost této úlohy velmi závisí na složitosti morfologie jazyka, za který je považován. Angličtina má poměrně jednoduchou morfologii, zejména skloňování, a proto je často možné ignorovat zcela tuto úlohu a jednoduše modelovat všechny možné formy slova (např. „open, opens, opened, opening“) jako jednotlivá slova. V jazycích, jako turecký nebo Manipuri (velmi aglutinovaný indický jazyk), však takový přístup není možný, protože každé heslo ve slovníku má tisíce možných forem.

- Označení slovních druhů (Part-of-speech tagging) -
  Určuje slovní druh každého slova ve větě. Spousta slov, obzvlášť běžných, může patřit mezi více slovních druhů. V některých jazycích se tato víceznačnost vyskytuje více než v jiných. Jazyky, ve kterých se málo skloňuje (např. angličtina), jsou na tyto víceznačnosti obzvlášť náchylné.

- Určování kořene slova (Stemming) -
  Řeší se např. pomocí lemmatizace (vytvoření základního tvaru slova) nebo n-gramové analýzy. Používá se např. k seskupování slov s podobným základním významem.

- Korekce chyb (Text-proofing) -
  Využívá se pro opravy překlepů v textu atd.

[10][12]

Syntax

Syntax je lingvistická disciplína, která se zabývá vztahy mezi slovy ve větě, gramaticky správným tvořením vět a slovosledem. Do syntaxe nepatří popis významu, který nesou jednotlivá slova a skupiny slov. Základní jednotkou je věta. Čeština představuje možná úskalí v tom, že nemá striktní pravidla pro uspořádání členů ve větě, tj. má volný slovosled. Vstupem syntaktických analyzátorů a generátorů jsou věty přirozeného jazyka, výstupem je jich reprezentace nejčastěji v podobě větných struktur (strom-graf s označením větných vztahů). [7][15]

Příklad úlohy

- Větný rozbor (Parsing) -
  Vytváří derivační strom (gramatickou analýzu) dané věty. Gramatika přirozených jazyků je nejednoznačná a typické věty mají více možných derivačních stromů. Typické věty mohou mít ve skutečnosti až tisíce možných rozborů (většina z nich však přijde člověku úplně nesmyslná).

[10]

Sémantika

Sémantika se zabývá významem jazykových výrazů (slov a jejich spojení). Popisuje, jak se jejich významy kombinují, aby tvořili smysluplné věty. Zde se uvažuje o významu vět nezávisle na kontextu. Vstupem je větná stromová struktura s označením větných vztahů. [7][15]

Příklady několika úloh

- Desambiguace smyslu slova (Word sense disambiguation) -
  Desambiguace znamená zjednoznačnění. Mnoho slov má několik významů. Musíme tedy zvolit ten, který dává v kontextu největší smysl. Pro tento typ problému máme obvykle seznam slov a souvisejících slovních významů. Např. ze slovníků nebo online zdrojů jako je WordNet.

- „Kapitalizace“ slov (Truecasing) -
  Cílem je určit, zda slova v textu mají začínat velkým písmenem. Využívá se např. při automatické úpravě textu, kde jsou všechna písmena malá (nebo velká). Uplatňuje se i v jiných NLP úlohách jako jsou NER, strojový překlad nebo automatická extrakce obsahu.

- Rozpoznávání názvů entit (Named entity recognition - NER) -
  Určuje, které položky v textu jsou vlastní jména (např. lidí, míst) a jejich typ (např. osoba, umístění, organizace). Velká počáteční písmena mohou pomoci v rozpoznávání názvů entit v jazycích jako je angličtina. Nemohou však pomoci při určování typu entity a v každém případě je to často nepřesné nebo nedostatečné. Kromě toho mnoho jiných jazyků v ne-západních skriptech (např. čínština nebo arabština) nepoužívá velká počáteční písmena. Jazyky, které je používají, je nemusí důsledně používat k odlišení jmen. Například v němčině začínají všechna podstatná jména velkým písmenem bez ohledu na to, zda se vztahují k názvům. Ve francouzštině a španělštině nezačínají velkým písmenem jména, která slouží jako přídavná jména.

[10][14]

2.2.2 Zpracování psaného textu

Příklady několika úloh

- Automatická sumarizace (Automatic summarization) -
  Produkuje čitelný souhrn části textu. Často se používá k poskytnutí shrnutí textu známého typu, jako jsou třeba novinové články o financích.

- Strojový překlad (Machine translation) -
  Automatický překlad textu z jednoho přirozeného jazyka do druhého. Toto je jeden z nejobtížnějších problémů a patří do třídy úloh hovorově nazývaných „AI-complete“ (UI-kompletní), tj. vyžadující veškeré různé typy znalostí, které lidé mají, k řádnému vyřešení (gramatika, sémantika, fakta o reálném světě, atd.).

- Porozumění přirozenému jazyku (Natural language understanding) -
  Převádí části textu do formálnější reprezentace jako jsou logické struktury prvého řádu, se kterými může počítačový program snadněji manipulovat. Porozumění přirozenému jazyku zahrnuje identifikaci zamýšlené sémantiky z několika možných sémantik, které mohou být odvozeny z výrazu přirozeného jazyka, který má obvykle podobu organizovaných zápisů konceptů přirozeného jazyka. Zavedení a vytvoření jazykového metamodelu a ontologie jsou efektivní, avšak emperická řešení. Explicitní formalizace sémantiky přirozených jazyků bez zmatků s implicitními předpoklady, jako jsou CWA (closed-world assumption) versus OWA (open-world assumption) nebo

subjektivní ANO/NE versus objektivní PRAVDA/NEPRAVDA se očekává pro sestrojení formalizace sémantik.

- Analýza sentimentu (Sentiment analysis) -
  Obvykle extrahuje subjektivní informace ze sady dokumentů často za použití online hodnocení k určení „polarity“ konkrétních objektů. To je užitečné zejména pro identifikaci trendů veřejného mínění na sociálních médiích za účelem marketingu.

- Segmentace a rozpoznávání témat (Topic segmentation and recognition) -
  Rozdělí text na segmenty z nichž každý je věnován nějakému tématu a následně to téma určí.

- Rozlišení koreference (Coreference resolution) -
  Rozhodne která slova v dané části textu odkazují („zmiňují“) na stejné objekty („entity“). Příkladem této úlohy je rozlišení anafory, které se konkrétně zabývá odpovídajícími zájmeny a podstatnými jmény nebo jmény, na které odkazují. Obecnější úloha rozlišení koreference také zahrnuje identifikaci takzvaných „bridging relationships“ zahrnující odkazovací výrazy.

- Zjednodušení textu (Text simplification) -
  V NLP se používá k úpravě, zlepšení, klasifikaci nebo při jiném zpracování existujícího korpusu textu tak, aby gramatika a struktura textu byla zjednodušená. Základní význam a obsažené informace zůstávají stejné. Jedná se o důležitou oblast výzkumu, protože pomáhá při automatizovaném zpracování přirozeného jazyka.

- Automatizované bodování eseje (Automated essay scoring)

[10][13]

2.2.3 Zpracování řeči

Příklady několika úloh

- Segmentace řeči (Speech segmentation) -
  Rozdělí záznam mluvené řeči po slovech. Jedná se o podúlohu rozpoznávání řeči.

- Rozpoznávání řeči (Speech recognition) -
  Převede záznam mluvené řeči do textové podoby. Jedná se o jeden z velice

obtížných problémů hovorově nazývaných „AI-complete“ (viz. výše). V přirozené řeči nejsou téměř žádné pauzy mezi následujícími slovy, a proto je segmentace řeči nezbytnou dílčí úlohou při rozpoznávání řeči (viz. níže). Ve většině mluvených jazyků splývají zvuky představující písmena jdoucí za sebou. Tento proces se nazývá koartikulace a převedení analogové signálu na jednotlivá písmena může být velice náročné.

- Syntéza řeči (Text-to-speech) -
  Jedná se o umělé vytváření řeči z textu napsaného v běžném jazyce. Programy vytvářející řeč se nazývají syntezátory. Řeč může být vytvořena spojením nahraných částí řeči uložených v databázi. Databáze může obsahovat např. fóny, difóny nebo celá slova. Alternativní metodou je tvoření řeči simulací charakteristik lidské řeči. Tím se vytvoří zcela syntetická (umělá) řeč.

[10][11]

2.2.4 Další úlohy

- Analýza rozmluvy (Discourse analysis) -
  Tato rubrika obsahuje řadu souvisejících úloh. Jednou z úloh je identifikace struktury rozmluvy připojeného textu, tj. podstata vztahů rozmluvy mezi větami (např. zpracování, vysvětlení, kontrast). Další možnou úlohou je rozpoznání a klasifikace aktů řeči v části textu (např. otázka ano-ne, obsah otázky, prohlášení, tvrzení, atd.).

- Generování přirozeného jazyka (Natural language generation) -
  Převod informací z počítačové databáze do lidského jazyka.

- Rozpoznávání optických znaků (Optical character recognition - OCR) -
  Rozpozná text na obrázku.

- Odpovídání na otázky (Question answering) -
  Určuje odpověď na základě uživatelovi otázky. Typické otázky mají specificky správné odpovědi (např. „Jaké je hlavní město České republiky?“). Někdy se berou v potaz i otevřené („open-ended“) otázky (např. „Jaký je smysl života?“). Nedávné práce se zaměřili na ještě složitější otázky.

- Extrakce souvislosti (Relationship extraction) -
  V dané části textu určuje vztahy mezi názvy entit (např. kdo si vzal koho).

- Vyhledávání v přirozeném jazyce (Natural language search)
- Rozšíření dotazu (Query expansion)

[10]

2.3 Korpusy textů

Jazykový korpus (z lat. corpus „tělo, těleso“) je rozsáhlý soubor autentických textů (psaných nebo mluvených) převedený do elektronické podoby v jednotném formátu tak, aby v něm bylo možné jednoduše vyhledávat jazykové jevy, zejména slova a slovní spojení (kolokace). Hlavní předností korpusu je vedle užití přirozeného jazykového materiálu i schopnost vypovídat o frekvenci (četnosti) jevů a jejich typickém úzu, což je informace jen pomocí badatelovy intuice nezjistitelná. Jelikož do korpusu vstupují texty jako celek, poskytuje na rozdíl např. od lístkového katalogu nevýběrové informace o všech typech jazykových jevů (navíc v rozsahu, který byl dříve nemyslitelný). [2]

Korpusy textů vznikají v rámci počítačové lingvistiky. Slouží lingvistům k pokusům o detailnější poznání jazyka. Korpusy jsou různého druhu, obsahují velké množství textů psaného i mluveného jazyka. Z hlediska obsahu by měly být co nejbohatší. Jednojazyčný korpus je souborem textů či promluv v jednom jazyce. Pro efektivní využití informace pro počítačové zpracování, je třeba shromážděným datům (korpusům) přiřadit značky, neboli anotace. To znamená přiřadit jednotlivým prvkům věty hodnoty kategorií (např. gramatických nebo lexikálně sémantických). [15]

Anotování jazykového korpusu sebou nese důležité výsledky v lingvistice a umožňuje vypracovat procedury, které by se mohly na základě dat, která vytvořili lingvisté, „naučit“ analyzovat běžný text, včetně neznámého textu, který je pro systém dosud nepoznán. [15]

2.4 Vybrané úlohy

2.4.1 Sumarizace

Automatická sumarizace vytváří z rozsáhlých textů a dokumentů souhrn nejdůležitějších informací a faktů. Stojí na porozumění obsahu původního textu a

obecně k ní lze přistupovat dvěma způsoby. Souhrn extraktní sumarizace je poskládán z jednotlivých vět původního textu či textů. Abstraktní sumarizace s využitím generování přirozeného jazyka sestaví shrnutí z nově vytvořených vět, které obsahují hlavní informace z originálu. [5][8]

Souhrn lze vytvořit z jednoho zdrojového textu (dokument, článek) nebo z několika. Multidokumentová sumarizace se používá třeba k shrnutí novinových (webových) článků na dané téma. Sumarizaci lze využít ke generování klíčových slov k danému textu, ke shrnutí obsahu jednou větou nebo k vytvoření souhrnu na základě uživatelova dotazu. [5][8]

2.4.2 NER

Rozpoznávání názvů entit (Name Entity Recognition - NER) automaticky identifikuje slova nebo fráze podle zvláštního významu v textech a zařadí je do patřičné skupiny (např. osoby, organizace, produkty, data, města, státy).

NER může být použito v široké škále aplikací. Např. v jiných NLP úlohách (např. Odpovídání na otázky nebo Strojový překlad) pro zlepšení svých výsledků, k indexování pro přesnější vyhledávání dokumentů týkajících se nějaké osoby či organizace nebo v analýze sentimentu k propojení výsledků s konkrétními produkty. [6]

Pro češtinu existují 4 systémy na rozpoznávání názvů entit. První dva systémy mají podobnou architekturu, ale používají různé metody (rozhodovací stromy a „support vector machines“) a lehce odlišnou sadu funkcí. V porovnání s konvenčními NER systémy používají oba nestandardní architekturu. Používají různé klasifikátory pro jednoslovně, dvouslovně a víceslovně pojmenované entity. Výstupy těchto klasifikátorů jsou v konečnému výsledku spojeny do jednoho, který obsahuje strukturované značky. Třetí systém může být považován za tradiční přístup k NER za použití „maximum entropy classifier“. Čtvrtý systém je založen na podmíněných náhodných polích. Výstup třetího a čtvrtého systému není strukturovaný. [4]

2.4.3 Identifikace jazyka

Rozpoznávání nebo-li identifikace jazyka určuje v jakém jazyce je obsah dokumentu napsán (např. čeština, angličtina, němčina atd.). Tento problém se řeší

pomocí různých statistických metod a bere se jako zvláštní případ kategorizace textu.

Může být využita ve spoustě jiných NLP úloh. Ve většině úloh je totiž nutné vědět, v jakém jazyce zpracovávaný text či hlasový projev je. Příkladem může být strojový překlad nebo odpovídání na otázky.

Existuje několik statistických postupů k identifikaci jazyka za pomoci různých technik pro klasifikaci dat. Jednou z metod je porovnat „stlačitelnost textu se stlačitelností textů“ v sadě známých jazyků. Tento postup je známý jako vzájemná výměna informací založená na změřené vzdálenosti. Stejná technika může být také použita k empirické konstrukci rodokmenů jazyků, které úzce odpovídají stromům zkonstruovaným za použití historických metod. Vzájemná výměna informací založená na měření vzdáleností je v podstatě ekvivalentní s více konvenčními „model-based“ metodami a není obecně považována za novější či lepší než jiné jednodušší techniky. [9]

Další možností je řešit problém pomocí n-gramového modelu. N-gram je sekvence písmen o `n` znacích extrahovaná z dokumentu. Nejčastěji se používají trigramy (n-gramy délky 3). Např. pro slovo „absolutní“ by vypadaly takto: abs-bso-sol-olu-lut-utn-tní. Pro každý rozpoznávaný jazyk je vytvořen model z trénovacích textů. Z části textu, pro kterou chceme jazyk určit, se vytvoří model stejným způsobem a následně je porovnán s modely vytvořenými z trénovacích textů. Nejpravděpodobnější jazyk je ten, jehož model se nejvíce podobá modelu, který chceme identifikovat. Problém nastává v případě, že text je v jazyce, pro který nemáme vytvořený model. V takovém případě může metoda jako výsledek vrátit jiný jazyk, který je identifikovanému nejpodobnější. Další problém může nastat v případě, že dokument obsahuje texty v různých jazycích (např. na webu). [1][9]

Jazyk lze identifikovat také pomocí starší statistické metody založené na převažujícím výskytu „funkčních“ slov (např. „the“ v angličtině) nebo vyhledáváním „zvláštních“ znaků, které daný jazyk používá (např. č, ř, š, ž, atd. v češtině). Problémovým případem je situace, kdy text žádné tyto znaky neobsahuje. Např. je příliš krátký nebo jej autor psal bez diakritiky. Některé jazyky mohou mít několik shodných zvláštních znaků. Např. „é“ používá čeština, italština, francouzština nebo švédština. To může být další problém. [1]

3 Návštěvnost a zpětná vazba

V rámci webové prezentace řešení NLP úloh je důležité mít zpětnou vazbu od návštěvníka. Autora algoritmu zajímá např. jaká data uživatel vložil, zda-li zkoušel upravit parametry, jaký výsledek dostal a jestli s ním byl spokojen. Proto jsem se v rámci práce seznámil se základními funkcemi jednoho z nejznámějších analytických nástrojů pro sledování webu, s Google Analytics. Zaregistroval jsem si účet a jako webovou stránku použil veřejný adresář mého studentského konta na školním serveru (home.zcu.cz). Zkusil jsem si nastavit sledování hlavní stránky a jedné v podadresáři, přidal sledování události pro stažení souboru a nakonec prošel několik přehledů.

3.1 Google Analytics

Google Analytics nabízejí spoustu funkcí a možností jak sledovat např. kdo a odkud se na vaše stránky dostal, na jaké odkazy klikal, jak interaguje s prvky Flash, kolik času na stránce strávil nebo kolikrát bylo spuštěno stahování nějakého souboru. Google Analytics nabízí pomoc především marketingovým oddělením firem či internetovým obchodům, jak zaujmout a přitáhnout více návštěvníků (zákazníků). Statistické údaje si lze prohlížet v připravených přehledech nebo si můžete vytvořit vlastní a zobrazovat si jen informace, které jsou pro vás relevantní.

Abyste je mohli začít používat, musíte mít vytvořený Google účet, jaký se používá i pro jiné jejich služby (např. Gmail, Youtube, G+, atd.). K tomuto účtu si zaregistrujete web (url adresu), který chcete sledovat. Pak do svých stránek (HTML souborů) přidáte kód v „JavaScriptu“, kterým načtete api a nastavíte sledování stránky. Úpravou kódu můžete nastavit např. název stránky, se kterým se bude v přehledu zobrazovat nebo sledování událostí. Pro správné přiřazení dat k vašemu účtu se použije uživatelské číslo, které vám bude vygenerováno po registraci stránky.

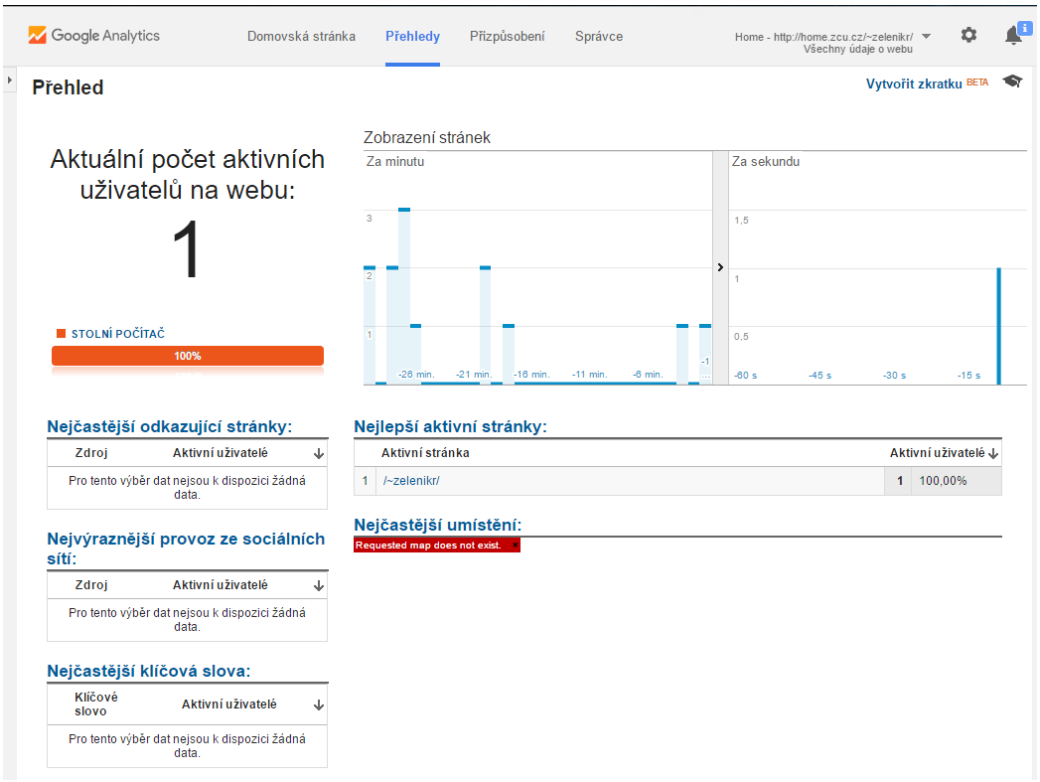
3.1.1 Přehledy

Google Analytics nabízejí spoustu různých tzv. Přehledů (např. „V reálném čase“, „Cílové publikum“ nebo „Chování“). Přehledy jsou složeny z podskupin

tématicky zaměřených grafů a tabulek s možností filtrování dat. Uživatelé si mohou navíc vytvořit vlastní přehledy pod záložkou „Přizpůsobení“.

V reálném čase

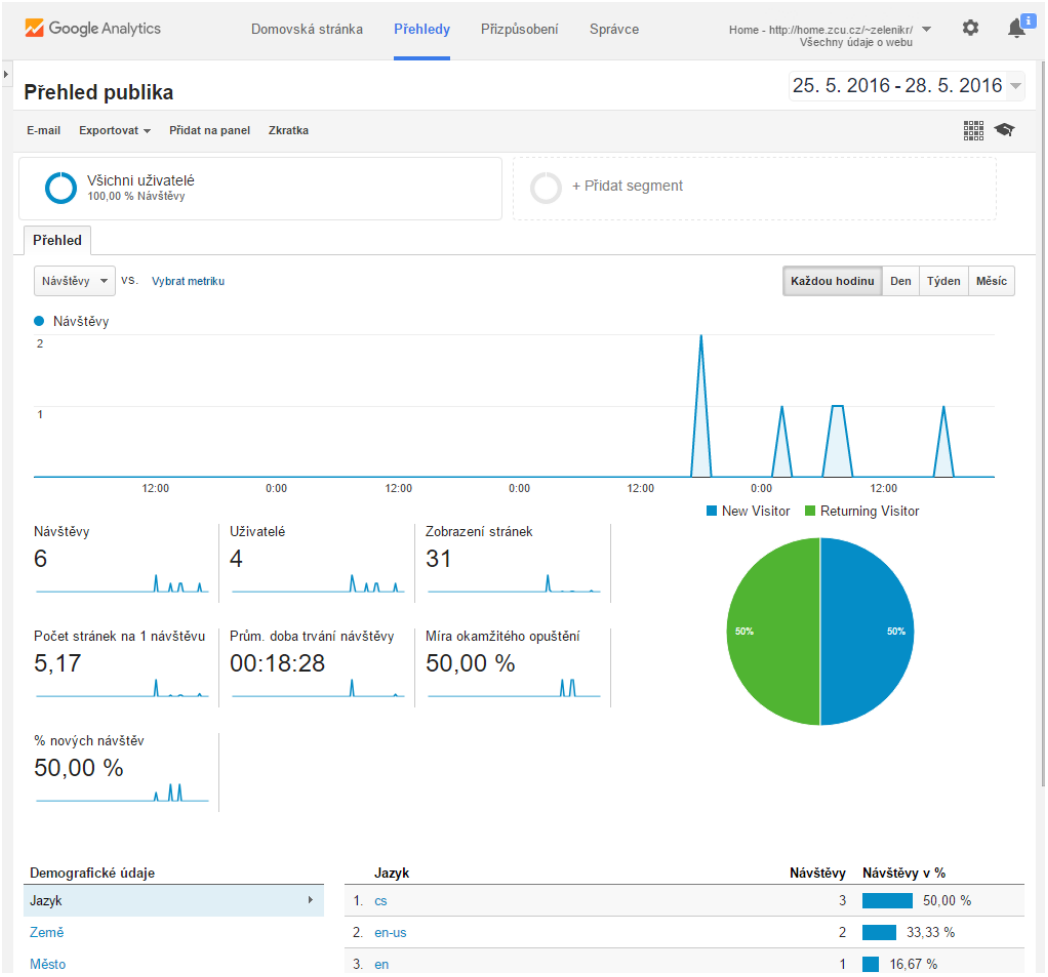
Díky prvnímu zmíněnému přehledu můžete sledovat aktivní uživatele. To znamená kolik uživatelů je právě aktivních na jednotlivých (těch sledovaných) stránkách, z jaké země (města) k webu přistupují nebo vyvolané události (ty sledované). Veškerá tato zobrazená data jsou za posledních 30 minut.



Obrázek 3.1: Ukázka přehledu „V reálném čase“

Cílové publikum

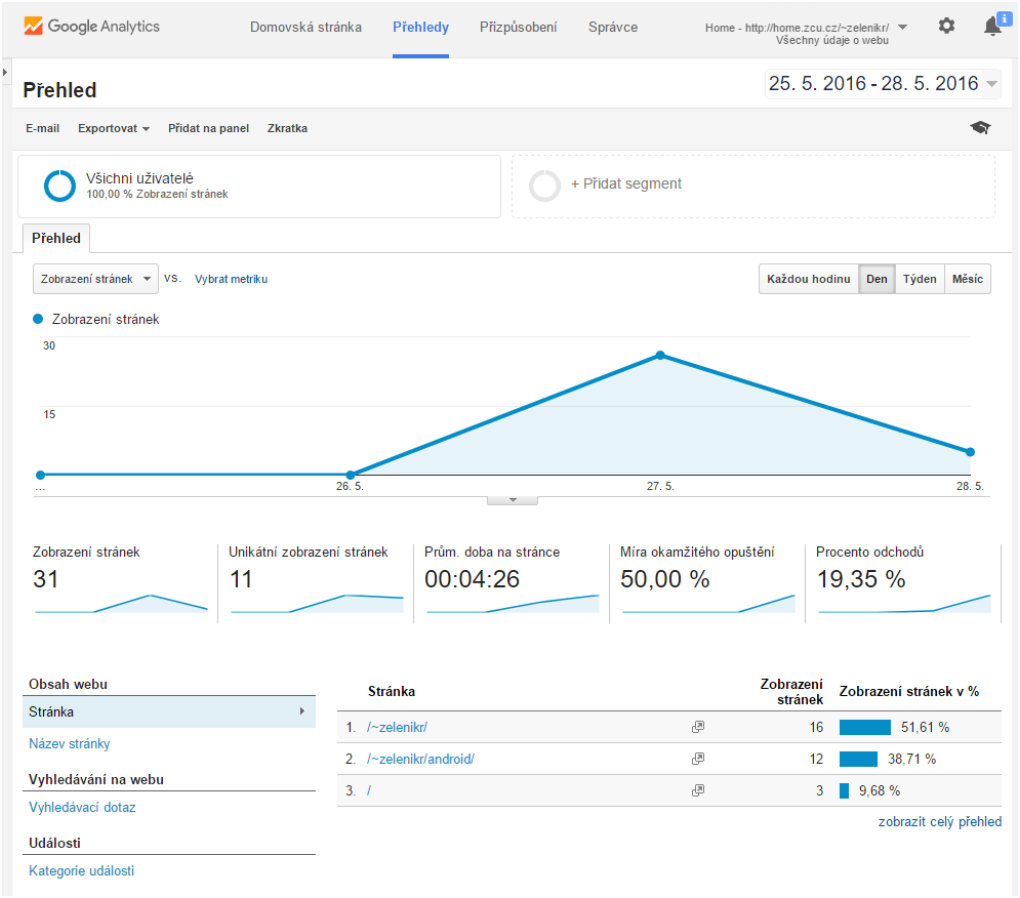
Tento přehled zobrazuje data z určitého období. Stejně jako v předchozím případě poskytuje demografické údaje o návštěvnících nebo z jakého zařízení (stolní počítač, tablet, mobilní telefon) a s jakým operačním systémem si uživatel stránky otevřel a jiné informace.



Obrázek 3.2: Ukázka přehledu „Cílové publikum“

Chování

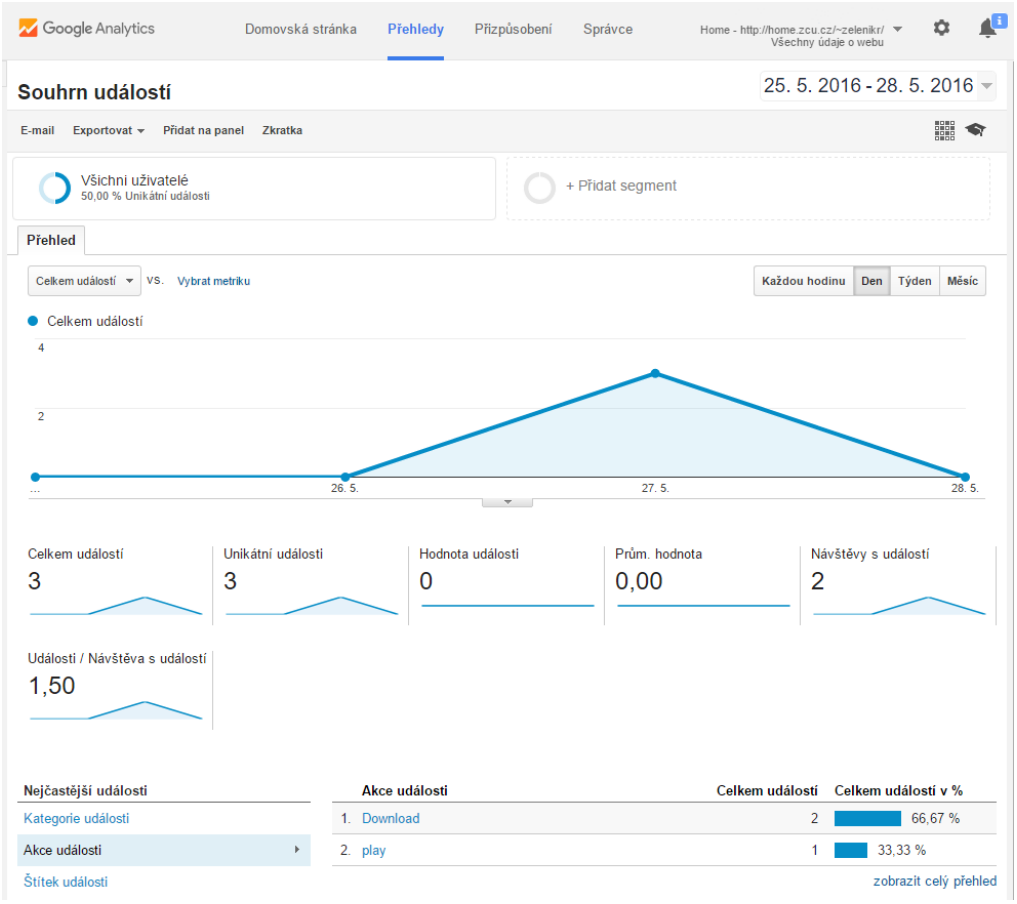
Třetí přehled, který bych chtěl zmínit, umožňuje prohlédnout si návštěvnost jednotlivých stránek nebo počty událostí (seskupené podle kategorie, typu nebo štítku).



Obrázek 3.3: Ukázka přehledu „Chování“

3.1.2 Události

V rámci této práce mě kromě přístupů na stránku zajímalo hlavně sledování událostí na stránce. Událost v Google Analytics má 4 parametry: kategorie, akce, štítek a hodnota. Události je tedy možno rozdělovat podle kategorií či typů akce. Štítek a hodnota jsou nepovinné. Hodnotou navíc může být pouze číslo.



Obrázek 3.4: Ukázka přehledu „Chování“ zaměřená na události

Pro implementaci ohodnocení výstupu algoritmu je to v pořádku, možná by šlo použít i k zaznamenání, jaké parametry uživatel nastavil, avšak uložit si vstupní či výstupní text nelze. To by musel řešit každý programátor v rámci běhu svého algoritmu. Jelikož to ale byl jeden z požadavků, který tato práce měla za cíl splnit, nepřichází tato možnost v úvahu.

3.2 Vlastní „Analytics“

Navrhl a implementoval jsem tedy vlastní zjednodušenou verzi monitorovacího systému webové stránky, který výše zmíněné požadavky splňuje. Snažil jsem se systém navrhnout tak, aby byl snadno rozšiřitelný a šly jednoduše přidávat další funkce pro monitorování činnosti návštěvníka či různých událostí.

Klientská část `Analytics` je v podstatě modul, který `NLP Framework` může používat a automaticky jej nastavit pro každé demo. Autor ho však může úpravou jedné řádky v konfiguračním souboru pro své demo vypnout, pokud potřebuje např. jen testovat výsledky algoritmu. Veškerá data k analyzování budou navíc uložena na předem určeném serveru (např. jednom z katedry KIV) a ne na serveru třetí strany.

4 Návrh

4.1 NLP framework

NLP algoritmus může být implementován např. v *PHP* nebo jako *Java Servlet*. Výsledkem by tedy měla být „front-endová“ aplikace běžící na straně klienta nezávisle na použitém serveru a programovací jazyce algoritmu. Aplikace bude tedy mezi klientem a serverem, na kterém algoritmus poběží, předávat mu vstupní data a výstupní prezentovat zpět klientovi. Komunikovat by tedy měl pomocí *HTTP* požadavků, které dokáže zpracovat každý webový server.

4.1.1 Struktura

Logika

Framework bude mít na starosti hlavně logiku. Vzít data ze správného elementu, zkontrolovat, připravit, odeslat na server ke zpracování a výsledek vložit do patřičného elementu stránky. Tyto funkce by se mohly rozdělit mezi různé komponenty, ze kterých se výsledná aplikace poskládá. Kde bude element na stránce umístěn nebo jak bude stylizován by funkčnost nemělo ovlivnit.

Měl by být tedy nezávislý na vzhledu stránky. Buď mu budou elementy, se kterými má pracovat, předány např. jako parametry při inicializaci nebo budou doplněny takovými atributy, aby mohly být na stránce jednoznačně identifikovány.

Vzhled

Vzhled různých komponent bude tedy definován mimo samotné komponenty. Po vzoru *MVC* (Model View Controler) by se o vzhled logických komponent - kontrolerů mohl starat jiný typ komponenty - view.

View se budou starat o rozložení a stylizování elementů pro kontrolery. Pomocí „JavaScriptu“ sice lze vytvořit *DOM* elementy s atributy a vnořovat je do sebe, nicméně podoba komponent by tak byla pevně daná a jakákoliv úprava vzhledu by znamenala zásah do kódu. Toto lze snadno vyřešit vytvořením *HTML* šablon. Pak jen stačí určit, jakou šablonu má view použít.

4.1.2 Rozšiřitelnost

Přidání nového dema by mělo být snadné, rychlé a s minimální nutností psaní kódu. Dema se od sebe budou hlavně lišit adresou serveru, kam odeslat data a případně jaké parametry (např. jazyk vstupního textu) algoritmus potřebuje nastavit. Další věci jako např. název/nadpis dema na stránce lze ovlivnit použitou šablonou, případně parametry šablony. Tato nastavení by bylo vhodné uložit do externích konfiguračních souborů a podle požadovaného dema je předat view a kontrolerům.

4.1.3 Nastavení

Vzhledem k tomu, že aplikace bude napsaná v jazyce *JavaScript* bude ideální použít pro definici konfiguračních souborů formát *JSON*.

4.2 Analytics

4.2.1 Interakce uživatele

Zachytávání událostí

Pomocí „JavaScriptových Handlerů“ lze reagovat na spoustu událostí. Knihovna jQuery umožňuje pro jeden typ události přiřadit elementu několik „Handlerů“. Je tedy možné ukládat si informace o interakci uživatele s různými elementy stránky a neovlivnit jejich funkčnost.

Ukládání událostí

Jsou dvě možnosti. Buď odeslat informace o události ihned, jakmile nastane nebo je dočasně ukládat v paměti prohlížeče a odeslat hromadně. Pokud bychom sledovali každý uživatelův klik a pohyb myši, byla by lepší druhá možnost. Jelikož nás budou zajímat jen některé události, můžeme je odesílat ihned.

Data se mohou na serveru ukládat do textových souborů nebo do databáze. Pro snazší analýzu a pozdější dotazování bude vhodnější ukládání do databáze.

Databáze

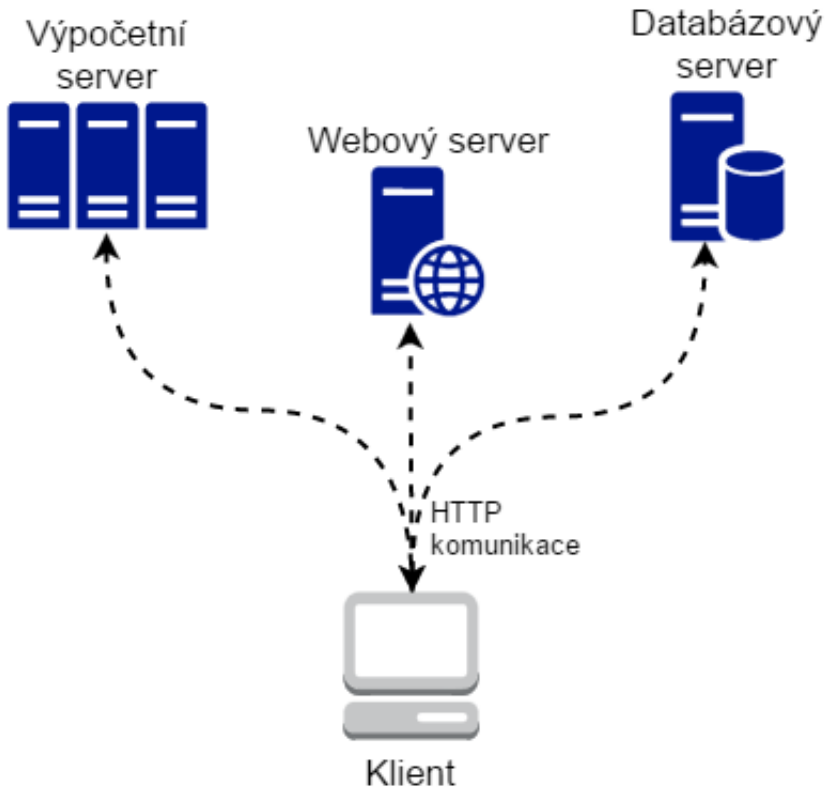
Existuje několik databázových modelů např. Relační databáze, Objektová databáze nebo dokonce Objektově relační databáze. Zvolil jsem relační, protože je nejznámější a pracoval jsem s ní již dříve. Kvůli programovému vybavení použitému při vývoji a testování (EasyPHP DevServer s modulem PhpMyAdmin) jsem jako SŘBD (Systém řízení báze dat) použil MySQL.

4.2.2 Vizualizace dat

Pro zobrazení a lepší srovnávání statistických údajů je vhodné použít grafy. Můžeme je vykreslit sami pomocí *HTML*, *CSS* a *SVG* (Scalable Vector Graphics – škálovatelná vektorová grafika). Druhou možností je použít hotové *JS* řešení a předat dané knihovně jen data k vykreslení. [3]

Existuje několik „JavaScriptových“ knihoven pro vytváření a práci s grafy. Např. Chart.js, Google Charts nebo C3.js. Vybral jsem si Google Charts, protože kromě velkého množství typů grafů umožňují zobrazit data i v tabulce. Všechny typy grafů jsou samozřejmě interaktivní a dají se různě přizpůsobit. Hlavním důvodem pro mě však byl takzvaný Dashboard, což je propojení grafu a filtru pro data. V jednom Dashboardu může být více grafů i filtrů a všechny mají společný zdroj dat.

5 Implementace

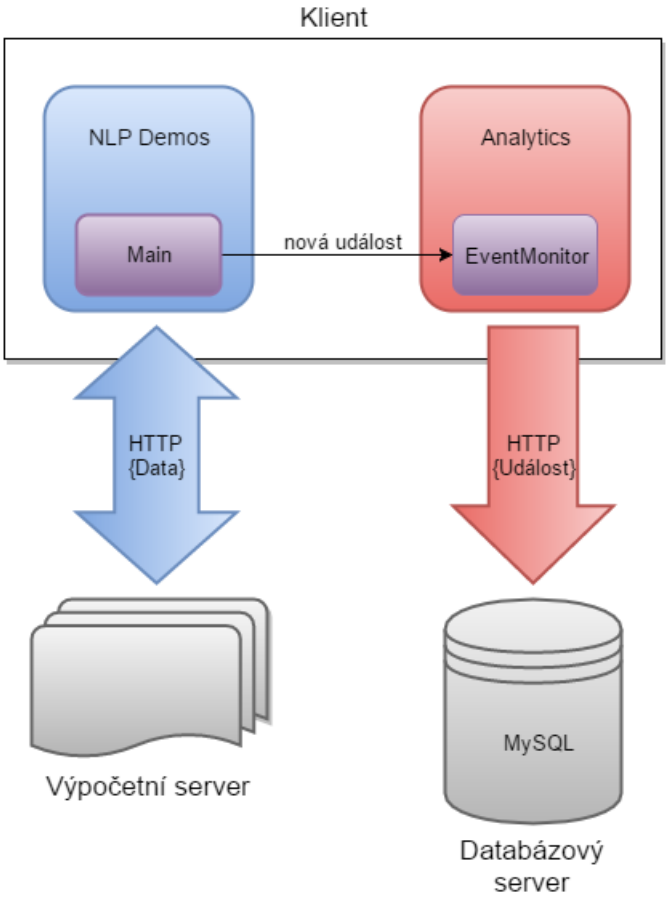


Obrázek 5.1: Rozdělení komunikace webu se servery

Na obrázku 5.1 jsou znázorněny 3 typy serverů, se kterými mnou vytvořený web a aplikace komunikují. Základem je webový server, kde jsou uloženy *HTML* stránky, styly, *JS* skrypty a knihovny. Výpočetní server představuje *HTTP* server, se kterým komunikuje aplikace v „NLP Frameworku“ prostřednictvím *HTTP* požadavků. Je znázorněn skupinou serverů, protože je možné nastavit pro každé demo (každý algoritmus) vlastní url adresu, kam data odesílat. Prostřednictvím posledního (Databázového) serveru `Analytics` ukládají nebo získávají data z databáze. Jedná se opět o *HTTP* server, který na základě přijatého požadavku uloží předaná data v „JSONu“ do databáze nebo naopak požadovaná data pro prezentaci z databáze získá.

Toto rozdělení serverů je možné, ale ne nutné. Funkce všech 3 výše zmíněných serverů může zprostředkovávat i jen jeden.

Ve výchozím nastavení je `EventMonitor` pro každé demo vypnutý a adresa databázového serveru je `null`. Pokud je v konfiguračním souboru zapnut a adresa nastavena, `Main` aplikace u něj zaregistruje posluchače, případně sám požádá o uložení konkrétní události. Obrázek 5.2 blíže znázorňuje předávání dat mezi aplikací pro dema a `Analytics` a jejich komunikaci s příslušnými servery.



Obrázek 5.2: Předávání dat mezi aplikacemi a servery

5.1 Použité technologie

5.1.1 Klient

HTML

Základní kostra a struktura všech stránek je po přeložení *Jade* šablon v *HTML*. O náplň jejich obsahu se stará *JavaScript*.

CSS

Vzhled stránek je definován pomocí *CSS* stylů. Použil jsem Bootstrap framework s motivem SpaceLab, který je možné zdarma stáhnout na `bootstrapzero.com`.

JavaScript

V „JavaScriptu“ je napsán „front-end“ celé aplikace pro prezentaci NLP algoritmů, zachytávání akcí uživatele, prezentace a interakce s daty v *Analytics* či *Bootstrap* animace.

AJAX

Je to asynchronní způsob komunikace a posílání dat mezi klientem a serverem prostřednictvím *HTTP* požadavků bez nutnosti opětovného načítání celé stránky. Posílají se jím na server uživatelem zvolené parametry a text ke zpracování NLP algoritmem nebo akce uživatele na stránce.

CoffeeScript

Zdrojové soubory „NLP Frameworku“ a `Analytics` jsou napsány v „CoffeeScriptu“, který umožňuje jednoduší a rychlejší psaní „JavaScriptového“ kódu. Ty jsou pak zkompilovány do „JavaScriptu“ a pomocí „webpack node-modulu“ spojeny do jednoho výsledného `.js` souboru, který je umístěn na stránce.

Jade

Jade je další *node-module*, který jsem při vývoji použil. Je to šablonovací jazyk pro psaní *HTML* šablon. Pomocí něj je definovaný vzhled komponent „frameworku“ i veškeré *HTML* stránky.

Node.js, npm a node-modules

Node.js je softwarový systém navržený pro psaní vysoce škálovatelných internetových aplikací, především webových serverů. Programy pro *Node.js* jsou psané v jazyce *JavaScript*. *NPM* (*node package manager*) slouží ke správě modulů, kterými je možné rozšiřovat *Node.js*.

Já jsem při vývoji použil např. již zmiňovaný *webpack*, kompilátor pro *CoffeeScript* a *Jade* šablony, *jade-loader*, *uglify* pro minimalizaci `.js` souborů nebo *grunt*, což je spouštěč úloh (např. kompilace `.coffee` souborů), který při každé změně zdrojových souborů nebo šablon automaticky spustí nový překlad, zkompilování a sestavení zdrojových souborů do výsledné `.js` aplikace.

5.1.2 Server

PHP

Implementace všech NLP algoritmů, zpracovávání „AJAXových“ požadavků či komunikace s databází jsou napsány jako *PHP* skripty.

MySQL

Pro vytvoření databáze, tabulek, ukládání nebo načítání dat jsem použil *MySQL* dotazy.

5.2 Adresářová struktura webu

5.2.1 Diagram

Všechny potřebné adresáře souboru pro chod webových stránek a aplikací jsou ve složce `public/`:

```
public/
├── analytics/
├── css/
│   ├── bootstrap/
│   └── fonts/
├── demo/
├── js/
│   └── lib/
└── php/
```

5.2.2 Obsah adresářů

Hlavní stránka `index.html`, na které je zobrazen také seznam dostupných dem, je v kořenovém adresáři `public/`. Stránka pro dema je v adresáři `demo/` a stránka se statistikami v adresáři `analytics/`. Obě se taktéž nazývají `index.html`.

Adresář `css/` obsahuje veškeré použité styly. V `bootstrap/` jsou styly a ve `fonts/` ikony v *svg* formátu, které využívá Bootstrap framework.

V `js/` jsou veškeré potřebné `.js` soubory. `NLP framework` uložený jako `nlpd.min.js` nebo `Analytics` jako `analytics.min.js`, který se stará o zobrazování tabulek, grafů a načítání statistických dat ze serveru. V podadresáři `lib/` jsou knihovny

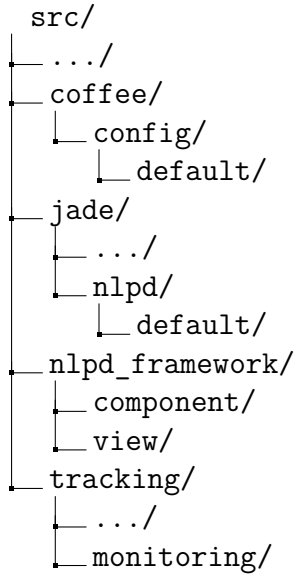
`bootstrap.min.js`, `jquery.min.js` a `lodash.compat.min.js`. Stránka se statistikami využívá ještě externích knihoven z *jsapi* od google.com. Především Google Charts pro práci s tabulkami a grafy.

Adresář `php/` je určen pro veškeré *PHP* skripty. Obsahuje „baseline“ implementace NLP algoritmů (`ajaxSumarization.php`, `ajaxNER.php` a `ajaxLangIdent.php`), `Database.php` pro komunikaci s databází a skripty pro zpracování *AJAX* požadavků.

5.3 Architektura NLP frameworku

5.3.1 Adresářová struktura

Diagram



Obsah adresářů

Adresář `nlpd_framework/` obsahuje zdrojové kódy v *CoffeeScriptu*. Jsou rozděleny do dvou podadresářů `component/` a `view/`. Je zde ještě soubor `main.coffee`, který určuje, s jakými parametry (demem) se aplikace spustí.

V `coffee/config/` jsou konfigurační soubory sloužící k nastavení aplikace. Jsou to v podstatě pouze *JSON* objekty. Výchozí nastavení určují konfigurační soubory v podadresáři `default/`. Do `config/` je možné přidávat vlastní konfigurační soubory obsahující jen ty parametry, které chceme změnit. Jsou zde i konfigurační

soubory pro *Jade* šablony.

V adresáři `jade/nlpd/` jsou šablony, které používá `NLP Framework` a určují vzhled komponent. V `default/` jsou předdefinované šablony použité ve výchozím nastavení. Do `nlpd/` je také možné přidávat vlastní šablony a tím změnit vzhled jednotlivých komponent.

Adresář `tracking/monitoring` obsahuje modul `eventMonitor`, který slouží k zachytávání a odesílání akcí uživatele na server k dalšímu zpracování.

5.3.2 (M)VC architektura

Component

Logiku a funkce mají na starosti komponenty. Nepracují s celou stránkou, ale jen s „DOMem“, který mají k dispozici. V něm vyhledají elementy, se kterými pracují. Elementy musí mít definovány patřičné `nlpd` atributy, aby např. vstupní parametry nebyly posbírány ze všech formulářových elementů a zároveň aby tyto elementy bylo možné nezávisle vnořit do jiných. Přípravu a prezentaci komponent uživateli zajišťují `view`. To se týká hlavních komponent.

compBase

Od této komponenty dědí všechny hlavní komponenty. Obsahuje metody pro načtení výchozích parametrů, vnořování dalších komponent a jiné.

Hlavní komponenty

Všechny hlavní komponenty jsou potomky `compBase` a jsou to:

compDemo - představuje samotné demo v závislosti na použitém nastavení a je složena z následujících komponent

compParameters - zpracuje parametry pro algoritmus nastavené uživatelem

compInputData - zpracuje vstupní data pomocí nastaveného „parseru“, přidá parametry a pošle na serveru ke zpracování daným algoritmem

compOutputData - přijme data ze serveru, zpracuje pomocí „parseru“ do prezentovatelné podoby a zobrazí je uživateli

compGlyphiconRating - umožňuje ohodnotit výstup NLP algoritmu, po potvrzení ohodnocení jej odešle na server k uložení

Pomocné komponenty

Pomocné komponenty nemají své vlastní `view` a nejsou potomky `compBase` komponenty. Jsou to:

compBaseParser - rozhoduje, která „parsovací“ funkce se má zavolat podle názvu v konfiguračním souboru, obsahuje „parsery“ `none` (ten je použit jako výchozí) a `json`, který předaná data (jsou-li ve formátu *JSON*) vrátí jako textový řetězec komponenty, které od této dědí, mohou oba tyto „parsery“ použít

compInputParser - potomek třídy `compBaseParser`, předzpracuje vstupní data do požadované podoby pro algoritmus na serveru
- jednotlivé „parsovací“ funkce jsou metody této komponenty, v konfiguračním souboru je uveden pouze název funkce (bez předpony `parser_`), kterou chceme na data použít
- je možné přidat vlastní „parsovací“ funkci (název musí začínat předponou `parser_`) a nastavit její název v konfiguračním souboru

compOutputParser - další potomek třídy `compBaseParser`, který upraví přijatá data do podoby, ve které jsou prezentována uživateli
- „parsovací“ funkce se používají stejně jako v předchozím případě a komponentu je možné stejně tak doplnit o další

View

Načte šablonu, vytvoří z ní *DOM* a předá ho jako *jQuery Object* komponentě. Každému `view` může být přiřazena jen jedna hlavní komponenta, kterou má za úkol uživateli zobrazit. Může také obsahovat další `view` pro vnořené komponenty. Vzhled je definován *Jade* šablonou, kterou načte pomocí *jade-loader* modulu jako „javascriptovou“ funkci, vytvoří z ní *DOM*, předá komponentě kvůli přípravě obsluhy a nakonec jako *HTML* kód vloží podle `wrapperId` do elementu s odpovídajícím id atributem.

viewBase

Všechna `view` od tohoto dědí. Obsahuje metody pro načtení šablony, konfiguračních souborů, jejich sloučení, vnořování dalších `view`, vykreslení na stránku atd.

Model

Tato část *MVC* architektury tu v podstatě chybí, protože neexistují data, která by zastupovala.

5.3.3 Main

Podle url adresy („hash tagu“ #) načte z `demos.coffee` parametry pro dané demo, vytvoří všechna `view` a komponenty, nastaví „callback“ funkce a spustí aplikaci (vygeneruje a zobrazí demo). Na základě načtených parametrů také připraví či nikoliv `eventMonitor` modul.

5.3.4 Komunikace a formát dat

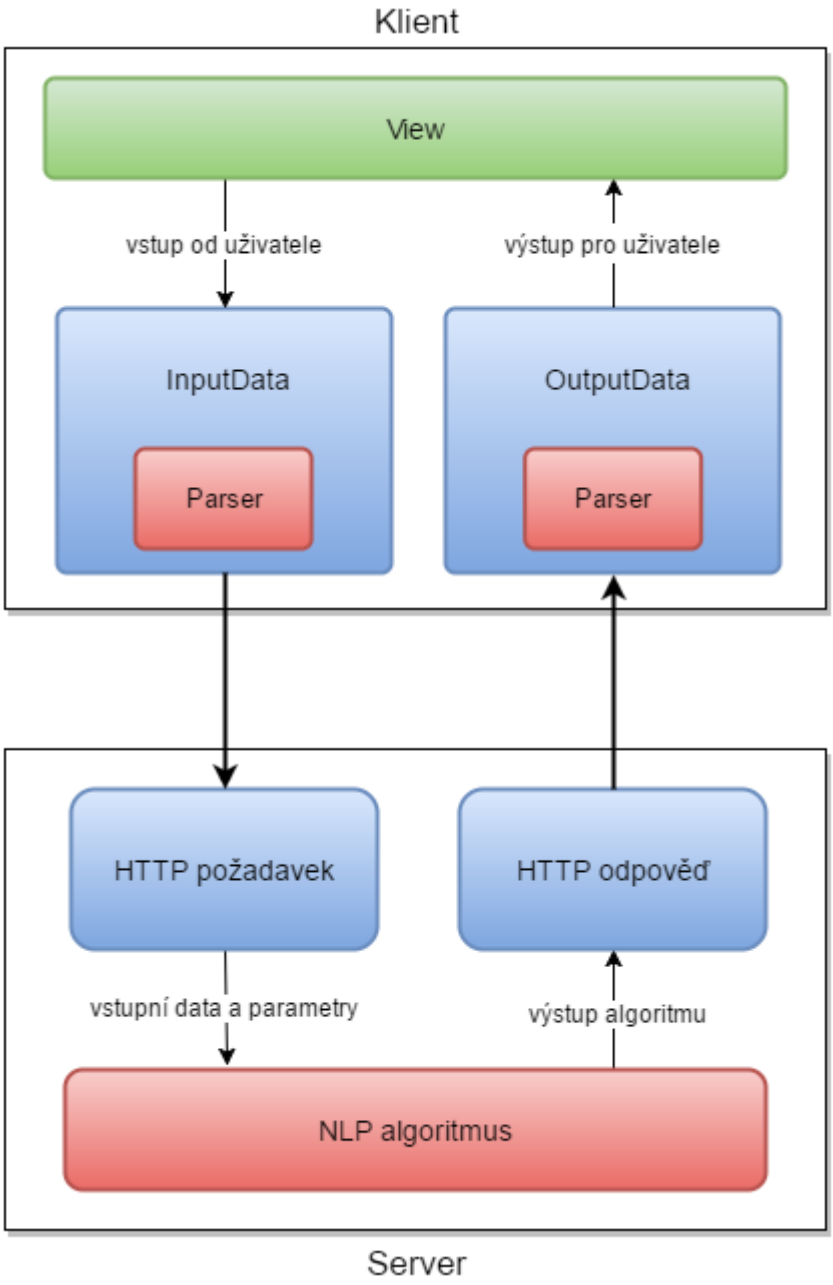
Komunikace s výpočetním serverem je uskutečněna pomocí *HTTP* požadavků. Jejich obsahem je *JSON*. Před odesláním jsou data předzpracována nastavenou „parsovací“ funkcí. Stejně tak jsou po přijetí od serveru zpracována do podoby, ve které se prezentují uživateli (viz. obr. 5.3).

Odchozí data

Nastavení parametrů a vstupní text od uživatele je po úpravě zvoleným „parserem“ odeslán na server jako `POST` s obsahem v následujícím formátu:

```
{
    "json": {
        "data": // vystup parseru
        "params": JSON
    }
}
```

Parametry jsou odeslány jako *JSON* objekt, kde klíčem je název parametru (hodnota atributu `name` daného *HTML* elementu) a hodnotou uživatelem vyplněná hodnota. Pokud parametr (element) postrádá `name` atribut, je uložen do pole. Toto pole je v objektu uloženo pod klíčem „_noname_“ a obsahuje všechny „bezejmenné“ parametry.



Obrázek 5.3: Diagram toku dat od uživatele na server a zpět

Tento požadavek posílá komponenta `compInputData` a jako odpověď očekává *JSON* objekt, který je předán komponentě `compOutputData`.

Příchozí data

Odpověď výpočetního serveru by měla být v tomto formátu:

```
{
    "data": // obsah
}
```

Obsahem může být text, pole nebo i jiný objekt. Záleží na programátorovi algoritmu. Hodnota klíče „data“ je předána výstupnímu „parseru“ a jeho výstup se vloží do stránky. Výstup může obsahovat libovolné *HTML* značky. Není-li nastaven žádný „parser“ (respektive je-li nastaven výchozí `none`), hodnota klíče „data“ je do stránky vložena rovnou.

5.3.5 Konfigurace, úpravy a rozšíření

Formát

Veškeré konfigurační soubory jsou v podstatě *JSON* objekty zapsané syntaxí jazyka *CoffeScript*. To znamená, že nejsou nutné `{}`. Zanoření je dáno odsazením jako např. v jazyce *Python*. Názvy klíčů nemusí být v `""` a za párem klíč-hodnota nemusí být `,` (pokud je každý pár na vlastní řádce). Kompilátor si však poradí i s klasickou syntaxí (jako v jazyce *JavaScript*).

Takto může vypadat nastavení pro demo Sumarizace:

demos.coffee

```
module.exports =
  sum:
    demo:
      confTemplate: "sumTempl"
    inputData:
      parserType: "sum"
      urlAjax: "../php/ajaxSumarization.php"
    parameters:
      template: "sumParams"
      confTemplate: "sumParamsTempl"
    monitor:
      monitor: true
      urlAjax: "../php/ajaxEventMonitoring.php"
      siteLabel: "Sumarizace"
```

Pro jeho načtení a zobrazení se použije url s `#sum`, aby `Main` vybral *JSON* objekt pod klíčem `sum`. Tyto atributy jsou předány příslušným komponentám a `view`. Pro každé `view` je možné nastavit atributy `template` (název použité šablony) a `confTemplate` (název souboru s atributy pro nastavenou šablonu).

Pod klíčem `monitor` jsou atributy pro `eventMonitor` module. Ve výchozím nastavení je vypnutý.

Pokud je třeba nastavení pro více dem, stačí každé přidat pod vlastní klíč:

demos.coffee

```
module.exports =
  sum:
    ...

  ner:
    ...

  langident:
    ...
```

Vzhled

Vzhled aplikace je možné změnit použitím vlastních šablon nebo konfiguračních souborů, které změní nastavení výchozích šablon.

Nové demo

Přidání nového dema se skládá z několika kroků:

1. Do konfiguračního souboru `demos.coffee` se přidají informace pro nové demo. Je třeba vždy nadefinovat url adresu, kam budou posílána data ke zpracování *NLP* algoritmem. Přidání dalších parametrů už není povinné. Avšak pro zachování konzistence webu je dobré nastavit ještě alespoň název konfiguračního souboru pro šablonu dema, ve které se nastaví např. název dema a v případě potřeby název šablony, která definuje použité formulářové elementy. Jako vzor může posloužit soubor `nlpd/sumParams.jade`, případně `config/sumParamsTempl.coffee`.

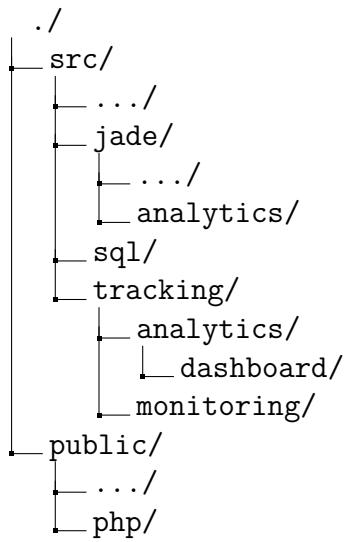
2. Je třeba připravit veškeré vlastní šablony a konfigurační soubory, které se nastavili v předchozím kroku.

3. Pokud potřebujeme vstupní/výstupní data před použitím zpracovat, přidáme do `demos.coffee` ještě název funkce našeho „parseru“. Jednotlivé „parsery“ jsou funkce v komponentách `compInputParser` a `compOutputParser`. Ve výchozím nastavení není použit žádný „parser“, respektive je nastaven `none`.

5.4 Architektura Analytics

5.4.1 Adresářová struktura

Diagram



Obsah adresářů

Adresář `jade/analytics/` obsahuje šablony pro `public/analytics/index.html` a její části či rozvržení tabulek a grafů.

V adresáři `sql/` je soubor `db_monitoring.sql` pro vytvoření databáze a všech tabulek.

Adresář `tracking/analytics` obsahuje veškeré zdrojové kódy v „CoffeeScriptu“ pro vytvoření a obsluhu grafů, tabulek a filtrů. V `dashboard/` jsou jednotlivé „Dashboardy“.

V `tracking/monitoring` je soubor `eventMonitoring.coffee`. Slouží k zachytávání a ukládání událostí na stránce.

V `php/` jsou mimo jiné soubory `ajaxEventMonitoring.php`, `ajaxAnalytics.php` a `Database.php` pro ukládání a získávání dat z databáze.

5.4.2 Sbírání a ukládání dat

EventMonitor

Ukládá datum a čas, kdy byla stránka otevřena, zavřena nebo kdy uživatel odeslal data ke zpracování algoritmem. Dále jaký text a s jakými parametry byl odeslán, přijatý výstup algoritmu, a jak byl uživatelem ohodnocen. Tyto události odesílá ihned na server k uložení jako *HTTP* požadavky.

Požadavky zpracovává `ajaxEventMonitoring.php`. Podle typu požadavku uloží data prostřednictvím instance třídy `Database.php` do databáze. Obsahem je *JSON*, který má tento formát:

```
{
    "type" : string // typ pozadavku
    "data" : JSON
}
```

Typem může být např. „pageLoad“ nebo „pageClose“. Pod klíčem `data` je vždy *JSON* s údaji vztahujícími se k typu události. Může obsahovat např. „timeStamp“ nebo „siteUrl“.

5.4.3 Prezentace dat

Google Charts

Na veškeré tabulky, grafy či filtry jsem použil Google Charts. Jedná se o „JavaScriptové“ api od společnosti Google. Umožňuje prezentovat data různými typy grafů nebo tabulkou. Jsou interaktivní a umožňují reagovat na různé události.

Google Charts umožňují vytvářet takzvané „Dashboardy“. `Dashboard` se skládá z jednoho či více grafů a tabulek (objektů typu `ChartWrapper`), filtrů (objektů typu `ControlWrapper`) a společnými daty. Data jsou reprezentována pomocí `DataTable`.

5.4.4 Příprava a interakce s daty

Pro získání určitých dat se pošle *HTTP* požadavek, který je zpracován skriptem v `ajaxAnalytics.php`. Podle typu požadavku (např. celková návštěvnost) zažádá prostřednictvím instance třídy `Database` definované v `Database.php` databázi o data. Spojení a veškeré *SQL* dotazy vytváří a spouští objekt třídy `Database`.

Každý `Dashboard` má svou *Jade* šablonu, která definuje rozvržení objektů `ChartWrapper` a `ControlWrapper` na stránce. Také je pro vytvoření a přípravu všech jeho částí a reakcí na události vytvořen samostatný `.coffee` soubor.

5.4.5 Main

Načte api pro Google Charts. Po načtení dat z databáze pro konkrétní `Dashboard` jej vytvoří, předá data a vloží ho do stránky.

5.4.6 Databáze

Veškerá data se ukládají do *MySQL* `nlpd_monitoring` databáze (viz. obr. 5.4). Databáze je navržena tak, aby bylo možné snadněji rozšířit `Analytics` o ukládání dalších informací, především různých událostí nad *HTML* elementy podobně jako to umožňují Google Analytics. K tomu slouží hlavně tabulky `events`, `event`, `action` a `element`.

Stručný popis tabulek:

visit - představuje jednu návštěvu, respektive vyzkoušení dema
- obsahuje uživatelem zadaný text, výstup NLP algoritmu, jeho ohodnocení a časové údaje

visitor - obsahuje informace o návštěvníkovi, který si stránku otevřel

site - načtená stránka, nemůžou zde být dvě se stejnou url adresou

parameter - obsahuje název a hodnotu parametru nastavenou uživatelem

parameters - rozkladová tabulka pro `visit` a `parameter`

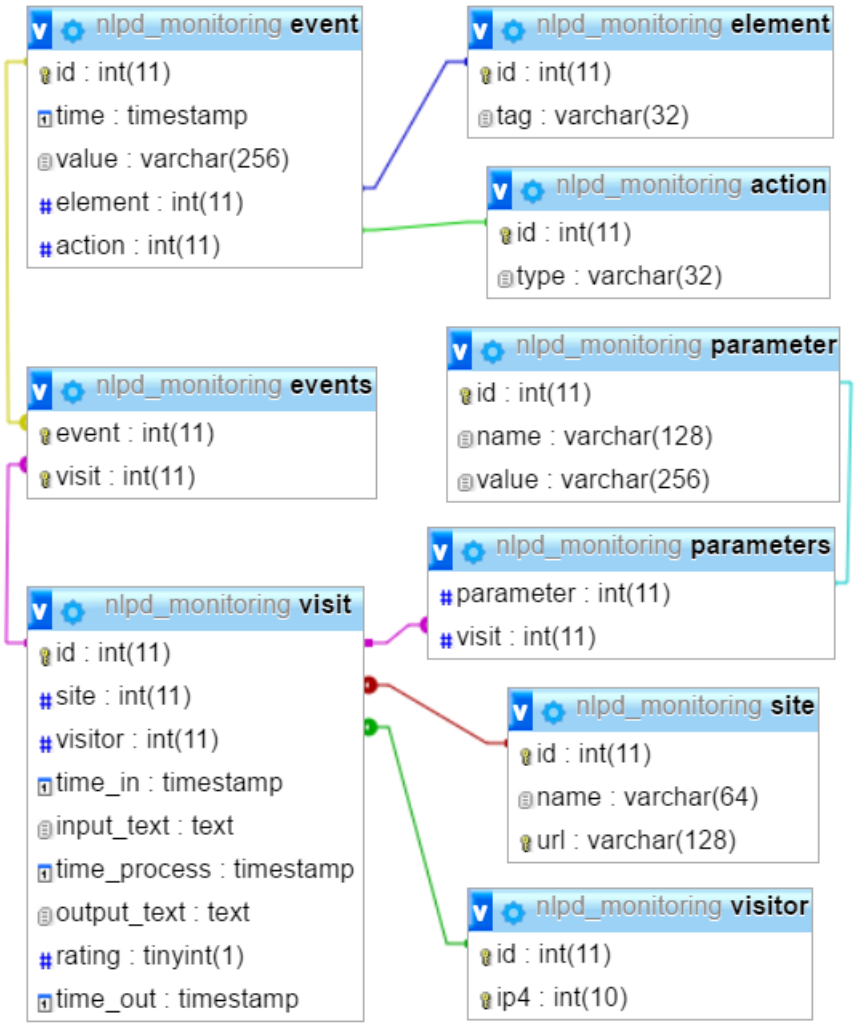
Následující tabulky jsem nakonec při ukládání dat nepoužil, avšak ponechal jsem je pro snazší rozšíření `Analytics`.

events - rozkladová tabulka pro `event` a `visit`

event - obsahuje časovou značku, hodnotu, id elementu a typu akce

action - slouží k ukládání typů akcí

element - slouží k ukládání elementů



Obrázek 5.4: ERA model

6 Dosažené výsledky

V rámci testování funkčnosti a pro demonstraci použití frameworku a Analytics jsem připravil mimo jiné tři vzorová dema a tři webové stránky. Na první stránce se nachází seznam všech dem, na druhé běží aplikace pro vybrané demo a třetí stránka prezentuje data nasbíraná pomocí Analytics.

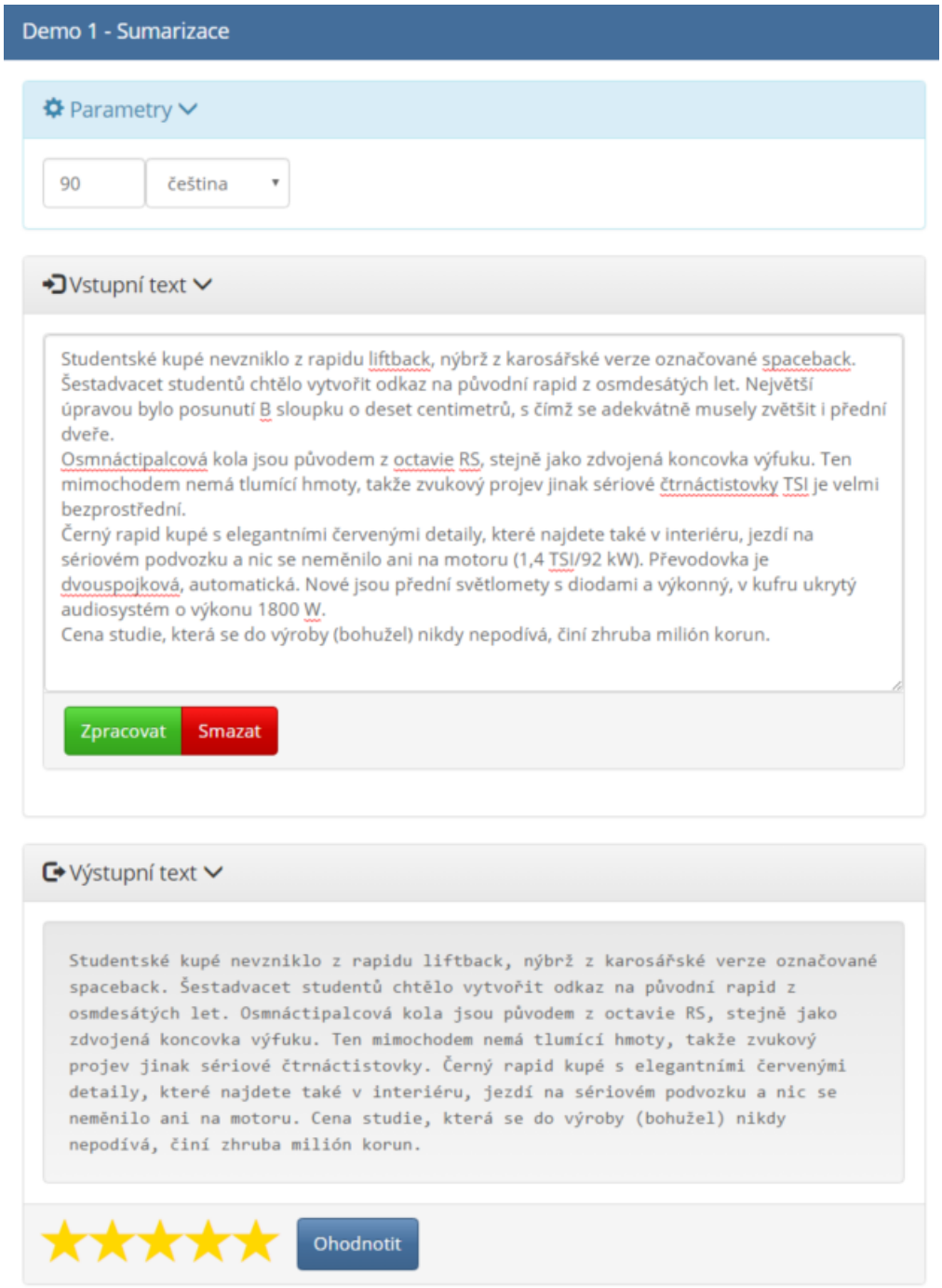
Konfigurační soubory a šablony, které jsem pro dema připravil mohou sloužit jako vzory pro pozdější přidání dalších dem. Obsahují několik variant nastavení a přizpůsobení. Např. přípravu šablony pro parametry algoritmu, nastavení nadpisu pro demo, nastavení „parseru“ (předpřipraveného, vlastního nebo žádného), nastavení bezparametrového dema nebo úpravu „glyphicon“ pro hodnocení.

6.1 Zkušební dema

Vybral jsem si tři NLP úlohy, pro které jsem naimplementoval velice zjednodušená řešení, abych ověřil správný chod komunikace, (před)zpracování vstupních/-výstupních dat a monitorování pomocí Analytics. Řešení každé úlohy je implementováno v samostatném PHP skriptu.

Sumarizace

Z každého odstavce použije pokud možno stejný počet slov tak, aby se vytvořený souhrn svou délkou (počtem slov) rovnal uživatelem nastavenému počtu slov. Na klienta odesílám hotový souhrn, který aplikace vloží rovnou do stránky.

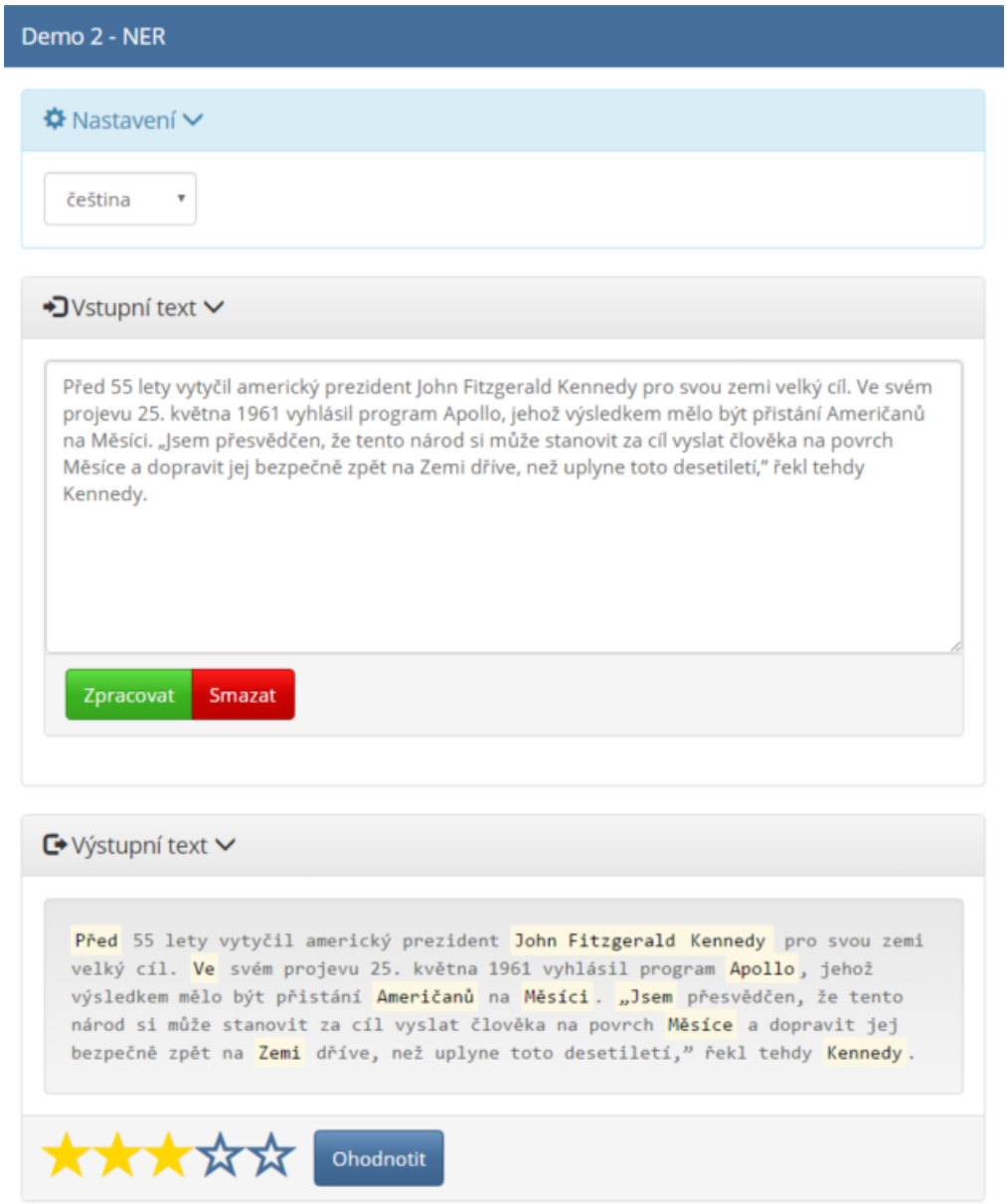


Obrázek 6.1: Ulázka Sumarizace

NER

Entity vyhledávám podle velkého počátečního písmena pomocí regulárních výrazů. Pokud následují odpovídající slova bezprostředně za sebou a nejsou odděleny interpunkčním znaménkem, označím je jako jednu entitu. Jako výsledek odesílám

zpět do aplikace původní text a pole s nalezenými entitami (bez duplicit). Pomocí „output parseru“ je v textu zvýrazním přidáním HTML tagu.

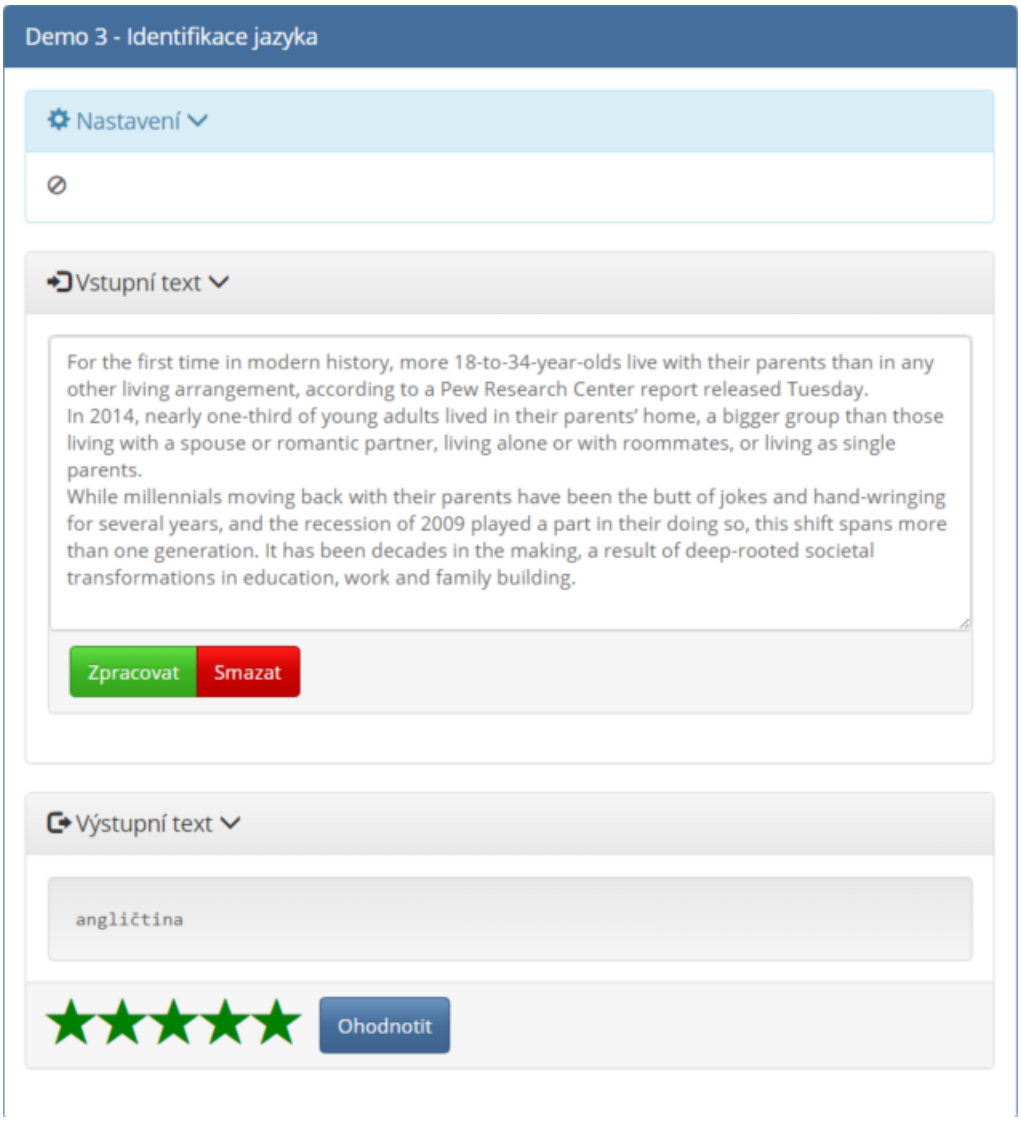


Obrázek 6.2: Ukázka NER

Během testování jsem zjistil, že můj zobecněný regulární výraz neakceptuje slova začínající „Š“. Nepomohlo ani jeho přímé přidání do samotného regulárního výrazu.

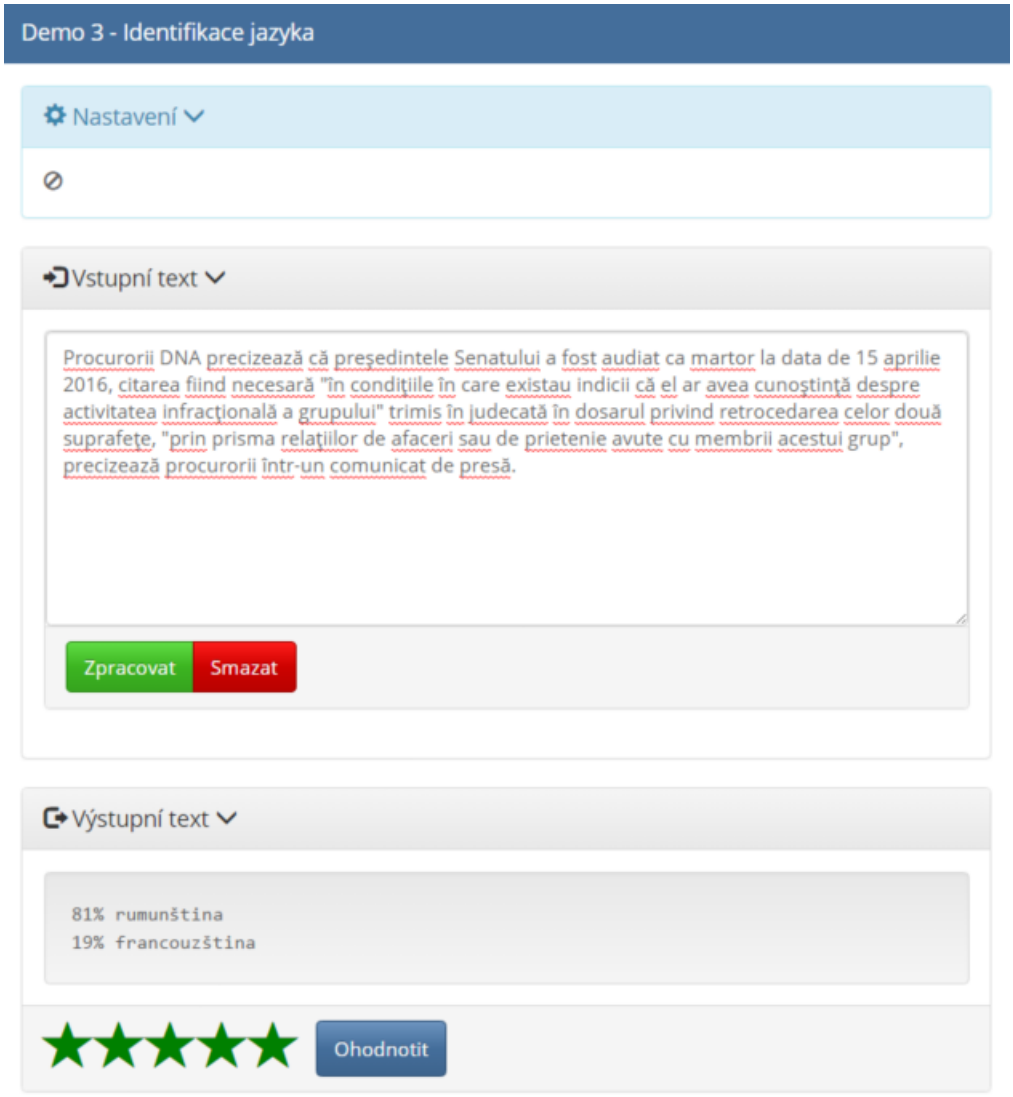
Identifikace jazyka

Pro identifikaci jazyka jsem použil hledání zvláštních znaků v textu. Připravil jsem si kolekci několika jazyků s jejich zvláštními znaky. Na základě celkového počtu (včetně duplicit) nalezených znaků pro každý jazyk se vytvoří seznam jazyků, ve kterých mohl být text napsán. Seznam je seřazen podle nejlepší shody (nejvíce nalezených znaků) pro daný text. Tento seznam (jako text) odešlu zpět do aplikace a ta ho rovnou vloží do stránky. Pokud text neobsahuje žádné zvláštní znaky, zkusím v něm vyhledat klíčová slova („a“, „an“, „the“). Pokud text některé z nich obsahuje, považuji ho za anglický. Nenajdu-li žádné, výsledkem je „Neznámý“.

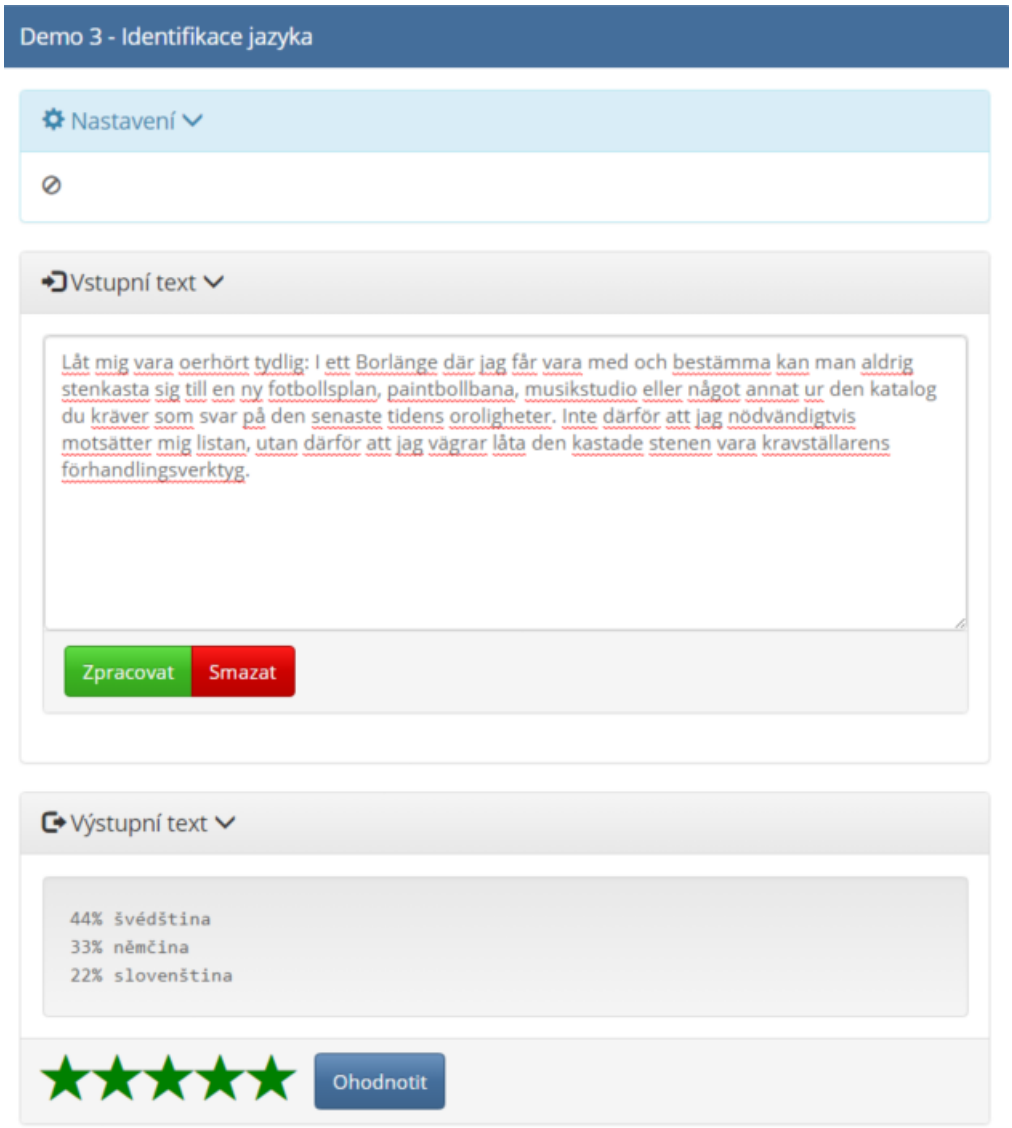


Obrázek 6.3: Ukázka rozpoznání anglického textu

Každý výčet zvláštních znaků je samostatný regulární výraz. Jako zkušební texty jsem použil odstavce z webových článků v jednotlivých jazycích. Algoritmus umí rozpoznat češtinu, slovenštinu, němčinu, francouzštinu, polštinu, italštinu, rumunštinu, švédštinu a španělštinu.



Obrázek 6.4: Ukázka rozpoznání rumunského textu



Obrázek 6.5: Ukázka rozpoznání švédského textu

6.2 Prezentace Analytics

Za pomoci `Google Charts` jsem vytvořil interaktivní „Dashboard“ s několika tabulkami a grafy.

Celkový přehled

Po načtení stránky se zobrazí souhrnný přehled monitorovaných dem. Skládá se z tabulky a grafu s proměnným zdrojem dat. Tabulka obsahuje název dema, jeho

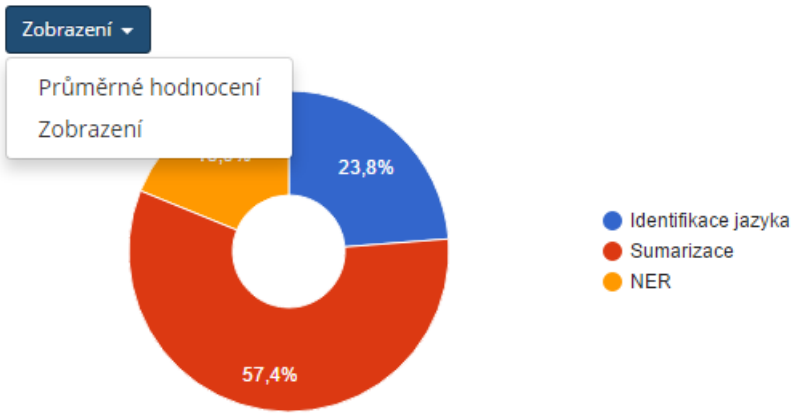
relativní url adresu na serveru, kolikrát bylo demo zobrazeno (a spuštěno[1]), kolika různými návštěvníky (přesněji IP adresami) bylo navštíveno a průměr hodnocení výstupů (viz obr. 6.6). Řádky je možné řadit podle kteréhokoliv sloupce.

Demo

Demo	Adresa	Zobrazení	Návštěvníků	Průměrné hodnocení
Sumarizace	/edsa-NLP%20Demos/public/demo/#sum	70	3	3,8
Identifikace jazyka	/edsa-NLP%20Demos/public/demo/#langident	29	3	3,8
NER	/edsa-NLP%20Demos/public/demo/#ner	23	2	2,9

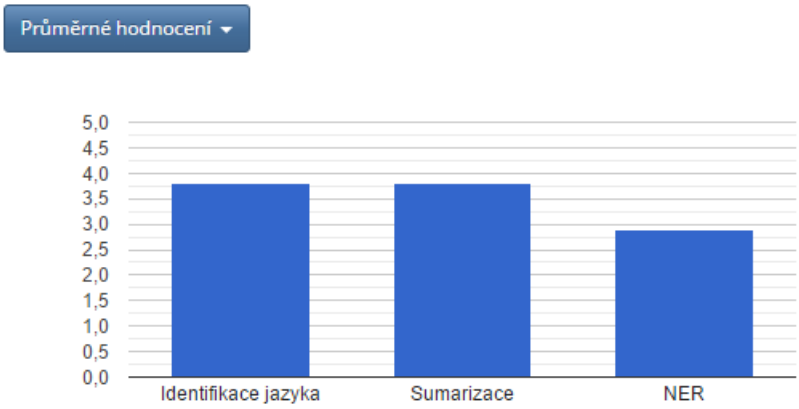
Obrázek 6.6: Seznam sledovaných dem

U grafu lze měnit zdrojový sloupec. Jako výchozí se zobrazí koláčový graf pro procentuální porovnání počtu zobrazení (a spuštění) (viz. obr. 6.7). Druhou možností je sloupcový graf pro porovnání průměrného hodnocení (viz. obr. 6.8).



Obrázek 6.7: Graf návštěvnosti dem

[1]Jako zobrazení se počítá každé první načtení dema a po prvním odeslání textu ke zpracování každé další kliknutí na zpracování.



Obrázek 6.8: Graf průměrného hodnocení dem

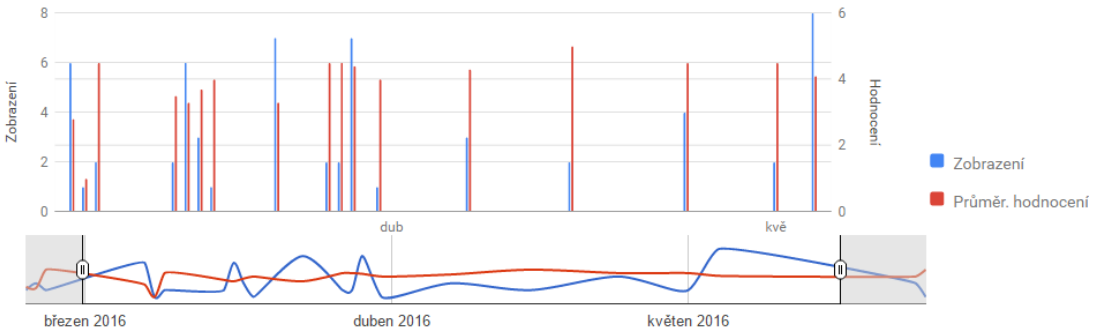
Přehled o používání dema

Po kliknutí na konkrétní demo v tabulce či grafu se načtou bližší informace o používání tohoto dema a zobrazí se pod celkovým přehledem.

	Datum	Text	Hodnocení	Uživatel	Čas přípravy	Čas zhodnocení	Celkový čas
1	25. 5. 2016	Studentské kupé nevzniklo z rapidu liftback, nýbrž z karosářské verze...	5	127.0.0.1	1m 25s	13m 33s	14m 58s
2	24. 5. 2016	Studentské kupé nevzniklo z rapidu liftback, nýbrž z karosářské verze...	5	127.0.0.1	5s	1m 18s	1m 23s
3	24. 5. 2016	Studentské kupé nevzniklo z rapidu liftback, nýbrž z karosářské verze...	4	127.0.0.1	33s	1m 20s	1m 53s
4	24. 5. 2016	„Po Slovensku a České republice se lidé z výšky nově...	3	127.0.0.1	1m 51s	1m 36s	3m 27s
5	4. 5. 2016	Developers wanting to track outbound links and forms can use...	5	127.0.0.1	8s	5s	13s
6	4. 5. 2016	Developers wanting to track outbound links and forms can use...	4	127.0.0.1	17s	9s	26s
7	4. 5. 2016	The configuration options of the chart drawn inside the control....	5	127.0.0.1	31s	4s	35s
8	4. 5. 2016	Replaces the automatically generated X-axis ticks with the specified array....	4	127.0.0.1	9s	11s	20s
9	4. 5. 2016	If true, draw the horizontal axis text at an angle,...	3	127.0.0.1	32s	15s	47s
10	4. 5. 2016	The difference between attributes and properties can be important in...	5	127.0.0.1	12m 58s	7s	13m 5s

Obrázek 6.9: Přehled pro Sumarizaci

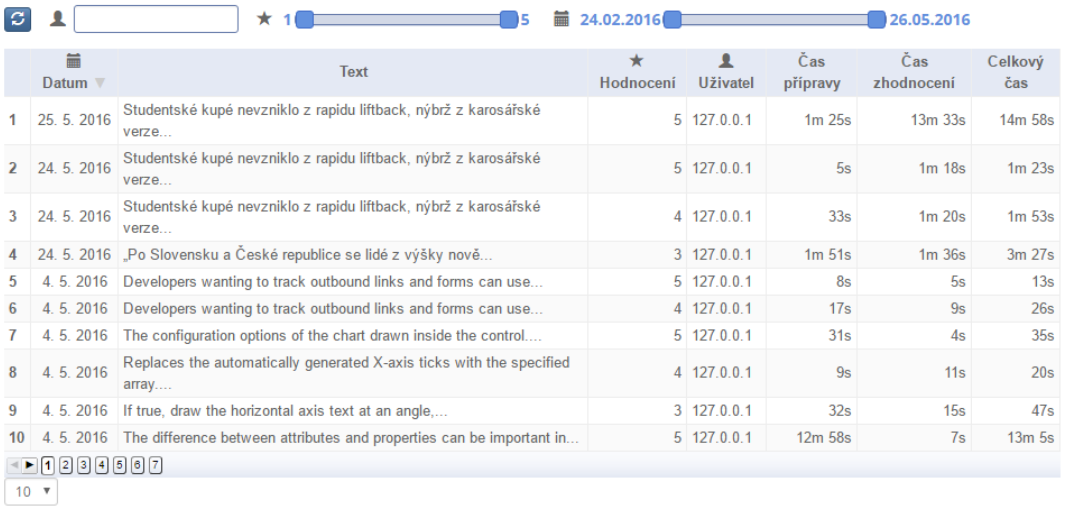
Z předchozího obrázku (6.9) je vidět, že se přehled pro konkrétní demo opět skládá z několika částí. První částí jsou dva grafy. Horní graf se dvěma Y-osami, a tedy se dvěma sloupci pro každý den, ve kterém bylo demo použito. Modrý představuje počet zobrazení, červený průměrné ohodnocení uživateli pro tento den. Pod sloupcovým grafem je ještě jeden menší, jehož modrá a červená křivka jsou aproximací dat sloupcového. Jezdce na krajích lze posouvat směrem ke středu, a tím vymezit specifičtější časový interval pro sloupcový graf (viz obr. 6.10).



Obrázek 6.10: Graf s časovou osou pro demo Sumarizace

Pod grafem je souhrnná tabulka (viz obr. 6.11) se třemi filtry pro výběr záznamů podle IP adresy návštěvníka, hodnocení výstupu a vymezení intervalu data návštěvy. Filtry je možné spolu kombinovat. Tabulka zobrazuje datum návštěvy, náhled zadaného textu, hodnocení výstupu, IP adresu návštěvníka a 3 časové údaje. Prvním je `Čas přípravy` pro zadání vstupního textu a případnou úpravu parametrů. Počítá se jako doba mezi načtením stránky a kliknutím na zpracování textu. Druhým je `Čas zhodnocení` výstupu. Tím se myslí prohlédnutí výstupu uživatelem a případné ohodnocení. Počítá se jako doba mezi odesláním textu a kliknutím na „Ohodnotit“, případně ukončením dema[2]. Třetím časovým údajem je `Celkový čas`, který představuje jak dlouho toto jedno použití dema trvalo. Jedná se o součet předchozích dvou časů. Záznamy jsou po načtení seřazeny od nejnovějších po nejstarší. Opět je možné použít řazení podle kteréhokoliv sloupce.

[2]Ukončením je míněno zavření stránky, přepnutí na jiné demo nebo znovu načtení stránky. V případě zpracování dalšího textu je použit časový údaj této události a následující záznam má tak 0 `Čas přípravy`.

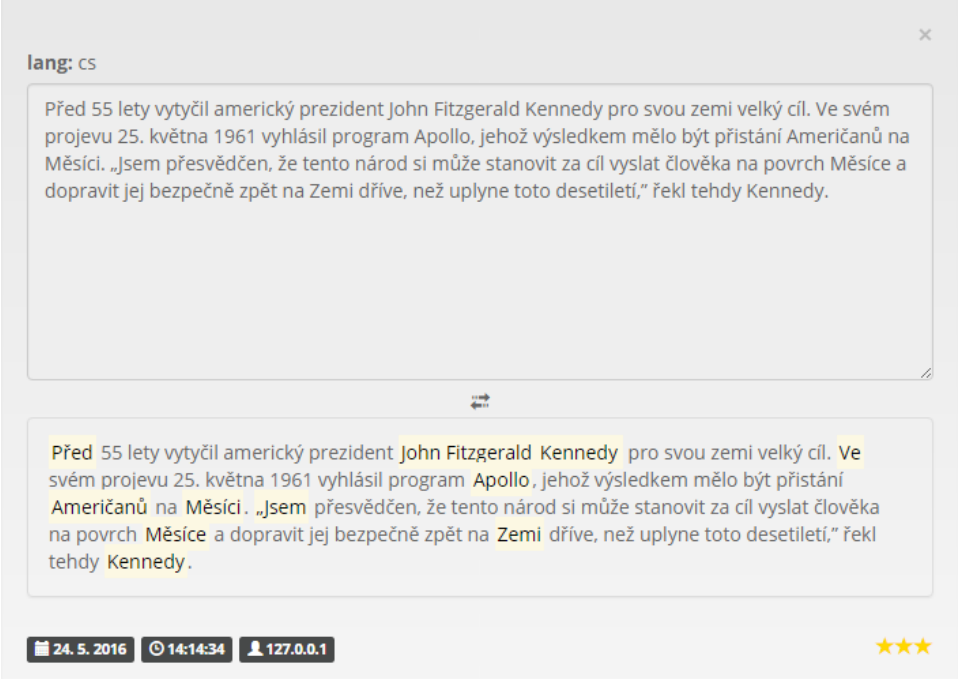


	Datum	Text	Hodnocení	Uživatel	Čas přípravy	Čas zhodnocení	Celkový čas
1	25. 5. 2016	Studentské kupé nevzniklo z rapidu liftback, nýbrž z karosářské verze...	5	127.0.0.1	1m 25s	13m 33s	14m 58s
2	24. 5. 2016	Studentské kupé nevzniklo z rapidu liftback, nýbrž z karosářské verze...	5	127.0.0.1	5s	1m 18s	1m 23s
3	24. 5. 2016	Studentské kupé nevzniklo z rapidu liftback, nýbrž z karosářské verze...	4	127.0.0.1	33s	1m 20s	1m 53s
4	24. 5. 2016	„Po Slovensku a České republice se lidé z výšky nově...	3	127.0.0.1	1m 51s	1m 36s	3m 27s
5	4. 5. 2016	Developers wanting to track outbound links and forms can use...	5	127.0.0.1	8s	5s	13s
6	4. 5. 2016	Developers wanting to track outbound links and forms can use...	4	127.0.0.1	17s	9s	26s
7	4. 5. 2016	The configuration options of the chart drawn inside the control....	5	127.0.0.1	31s	4s	35s
8	4. 5. 2016	Replaces the automatically generated X-axis ticks with the specified array....	4	127.0.0.1	9s	11s	20s
9	4. 5. 2016	If true, draw the horizontal axis text at an angle,...	3	127.0.0.1	32s	15s	47s
10	4. 5. 2016	The difference between attributes and properties can be important in...	5	127.0.0.1	12m 58s	7s	13m 5s

Obrázek 6.11: Záznamy o používání dema Sumarizace

Detailní zobrazení

Po výběru záznamu (kliknutí na řádek) se načtou detailnější informace. Obsahují kompletní vstupní text, zvolené hodnoty parametrů, výstup (tak jak byl prezentován návštěvníkovi), datum a čas otevření dema, IP návštěvníka a jak výstup ohodnotil (viz obr. 6.12).



Obrázek 6.12: Detail záznamu NER dema

7 Závěr

Výsledkem této práce je jednoduše stylizovaný web složený ze tří stránek (seznam všech dem, vyzkoušení konkrétního dema a zobrazení statistik návštěvnosti a dalších informací) a tří vzorových interaktivních dem prezentujících NLP algoritmy. Dema představuje „single-page“ aplikace, která podle url mění svůj obsah (prezentovaný NLP algoritmus). Zdrojový kód aplikace je napsán v „CoffeeScriptu“, ve kterém je i kód celého NLP frameworku, jež aplikace využívá. Součástí práce jsou i dva speciální soubory pro přeložení aplikace do „JavaScriptu“ a šablon do *HTML* souborů pomocí programů Node.js a Grunt. Také je připraven *SQL* skript pro vytvoření *MySQL* databáze a *PHP* skripty, které slouží jako jednoduché „mini api“ pro ukládání a získávání dat z dané databáze. Tato data jsou sbírána a prezentovaná pomocí modulů souhrnně označovaných jako `Analytics`. Jsou taktéž napsány v „CoffeeScriptu“.

Úprava nastavení (url serverů, změna parametrů, názvu dema) či přidání nového dema do „JavaScriptové“ aplikace představuje jen úpravu konkrétního řádku v příslušném konfiguračním souboru (souborech). Je ale nutné ji znovu sestavit a výsledný skript nahrát na webový server namísto stávajícího. Při přidání nového dema nebo změně názvu je samozřejmě nutné upravit i webové stránky (seznam všech dem či rozbalovací seznam pro výběr dema). Změna vzhledu webových stránek se provádí v příslušných šablonách, které je také nutné znovu přeložit, a tak vygenerovat nové *HTML* soubory. Avšak je možné editovat stránky i přímo na serveru (je ale nutné ohlídat, aby nebyly přepsány při příštím překladu šablon).

Kromě webového serveru jsou potřeba ještě další dva *HTTP* servery. Jeden předává data algoritmům, druhý přistupuje k databázi. Avšak všechny 3 služby mohou být poskytovány i jedním *HTTP* serverem.

V databázi jsou připraveny tabulky, které mohou sloužit k ukládání vlastních akcí nad elementy stránky, včetně případných hodnot (např. `<input>` elementů). To by mělo usnadnit rozšíření `Analytics` o zaznamenávání dalších událostí a akcí nad demi. Dále je možné rozšířit prezentaci `Analytics` o další agregované údaje či přidat nové grafy nebo tabulky.

Literatura

[1] Barber, I. *Language Detection With N-Grams* [online]. 2009. [cit. 2016/05/11]. Dostupné z: http://phpir.com/language-detection-wth-n-grams/.

[2] Cvrček, V. – Richterová, O. *pojmy:korpus — Příručka ČNK* [online]. Příručka ČNK, 2014. [cit. 2016/05/13]. Dostupné z: http://wiki.korpus.cz/doku.php?id=pojmy:korpus&rev=1416829573.

[3] *JS grafy na webu* [online]. 2014. [cit. 2016/04/18]. Dostupné z: http://jecas.cz/grafy.

[4] Konkol, M. – Konopík, M. *CRF-based Czech Named Entity Recognizer and Consolidation of Czech NER Research* [online]. 2013. [cit. 2016/05/17]. Dostupné z: http://nlp.kiv.zcu.cz/file/10.

[5] Nenkova, A. – McKeown, K. *Automatic Summarization* [online]. 2011. [cit. 2016/05/18]. Dostupné z: https://www.cis.upenn.edu/~nenkova/1500000015-Nenkova.pdf.

[6] *NLP Group - Named entity recognition* [online]. 2014. [cit. 2016/04/12]. Dostupné z: http://nlp.kiv.zcu.cz/research/ner.

[7] Pala, K. *Počítačové zpracování přirozeného jazyka (pracovní verze)* [online]. 2000. [cit. 2016/05/12]. Dostupné z: https://nlp.fi.muni.cz/poc_lingv/pala_zprac.pdf.

[8] *Automatic summarization* [online]. Wikipedia, 2016. [cit. 2016/05/18]. Dostupné z: https://en.wikipedia.org/wiki/Automatic_summarization.

[9] *Language identification* [online]. Wikipedia, 2016. [cit. 2016/05/11]. Dostupné z: https://en.wikipedia.org/wiki/Language_identification.

[10] *Natural language processing* [online]. Wikipedia, 2016. [cit. 2016/05/13]. Dostupné z: https://en.wikipedia.org/wiki/Natural_language_processing.

[11] *Speech synthesis* [online]. Wikipedia, 2016. [cit. 2016/05/28]. Dostupné z: https://en.wikipedia.org/wiki/Speech_synthesis.

[12] *Stemming* [online]. Wikipedia, 2016. [cit. 2016/05/21]. Dostupné z: https://en.wikipedia.org/wiki/Stemming.

[13] *Text simplification* [online]. Wikipedia, 2016. [cit. 2016/05/21]. Dostupné z: https://en.wikipedia.org/wiki/Text_simplification.

[14] *Truecasing* [online]. Wikipedia, 2016. [cit. 2016/05/21]. Dostupné z: https://en.wikipedia.org/wiki/Truecasing.

[15] *Zpracování přirozeného jazyka* [online]. Wikipedia, 2015. [cit. 2016/04/11]. Dostupné z: https://cs.wikipedia.org/wiki/Zpracov%C3%A1n%C3%AD_p%C5%99irozen%C3%A9ho_jazyka.

Příloha

A Instalace a sestavení

Abychom mohli nainstalovat a spustit potřebné npm moduly, bude potřeba si nejdříve nainstalovat Node.js. Stáhnout si ho můžete z oficiálních stránek projektu `nodejs.org`.

1. Nainstalujte Node.js
2. Spusťte terminál/příkazový řádek v kořenovém adresáři. Měl by obsahovat všechny tyto položky:

   src/

   package.json

   Gruntfile.coffee
3. Nainstalujte CLI (Command Line Interface) pro Grunt příkazem:
   `npm install -g grunt-cli`
4. Spusťte příkaz:
   `npm install`
   Tím se nainstalují všechny moduly ze souboru `package.json`. V adresáři by měl přibýt podadresář `node_modules/`.

Nyní můžeme přeložit šablony a zdrojové kódy a sestavit aplikace příkazem: `grunt` Tím se spustí úlohy nastavené v souboru `Gruntfile.cofee`, které přeloží `.coffee` soubory do podadresáře `compiled/`. Z šablon vytvoří *HTML* stránky a z `.js` souborů v `compiled/` sestaví `nlpd.js` a `analytics.js`. Ty jsou nakonec minimalizovány do `nlpd.min.js` a `analytics.min.js` a uloženy s *HTML* stránkami (včetně podadresářů) v podadresáři `public/`. Nyní stačí obsah `public/` přidat do adresáře na serveru, kde webové stránky poběží.

B Uživatelská příručka

B.1 NLP Demos (web)

B.1.1 Navigace

Obrázek B.1: Hlavní stránka

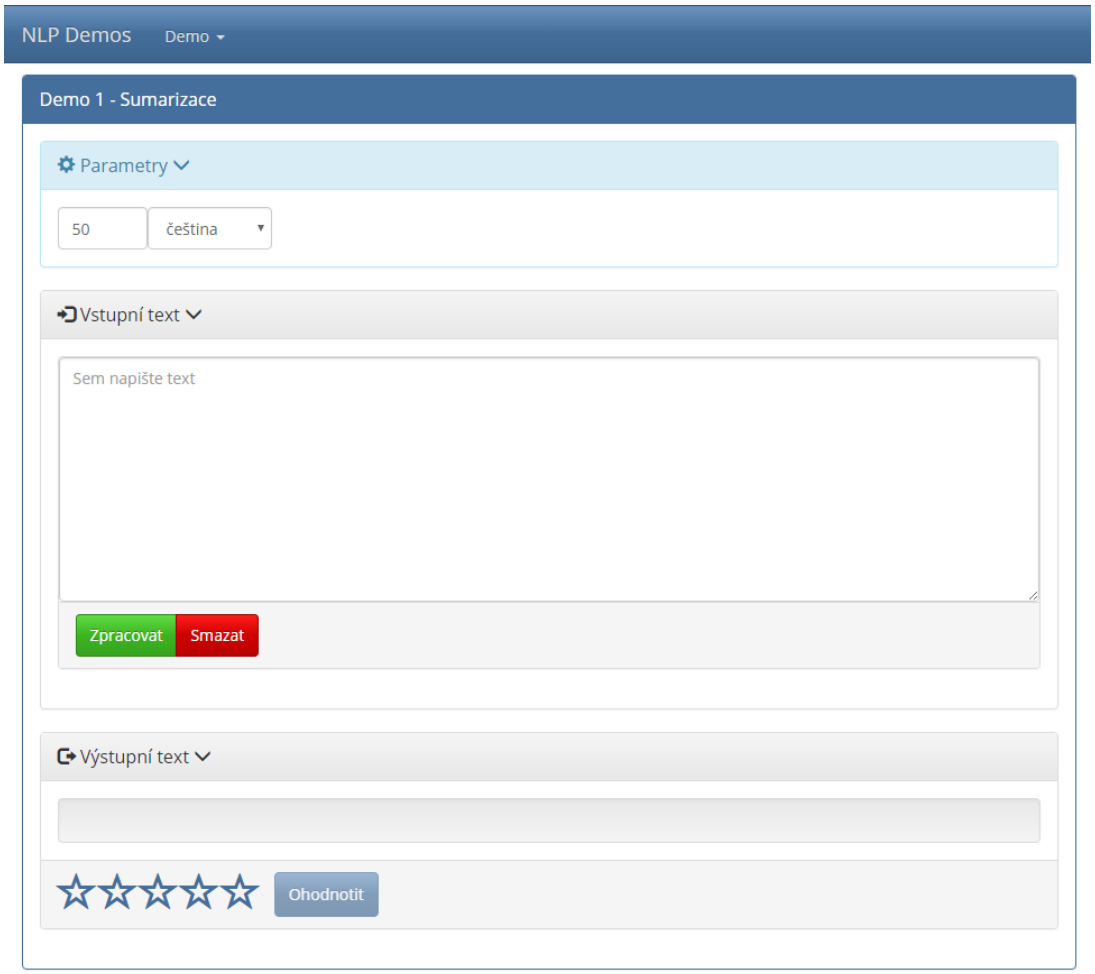
Hlavní stránka obsahuje seznam dem se stručným popisem dané NLP úlohy. Konkrétní demo se otevře po kliknutí na „Vyzkoušet“ nebo výběrem z rozbalovacího seznamu (viz. obr. B.2).

Obrázek B.2: Seznam dem

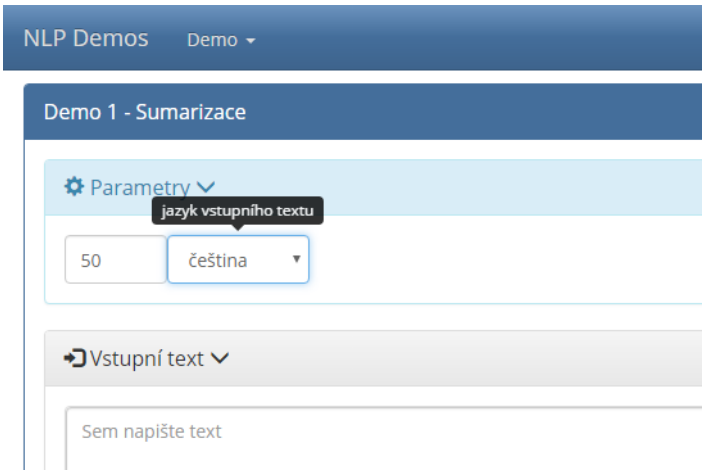
Tento rozbalovací seznam lze využít i při přechodu z jednoho dema na jiné. Kliknutím na „NLP Demos“ se vždy vrátíme na hlavní stránku.

B.1.2 Ovládání dema

Demo je složené ze 3 panelů (viz obr. B.3). Každý panel je možné srolovat kliknutím na jeho název. V panelu „Parametry“ může uživatel měnit přednastavené hodnoty parametrů pro algoritmus. Najetím ukazatele myši na konkrétní parametr se zobrazí krátký popis (viz obr. B.4). Panel „Vstupní text“ obsahuje pole pro zadání textu, který chceme algoritmem zpracovat. Tlačítkem „Zpracovat“ text odešleme na server ke zpracování i se zadanými parametry. „Smazat“ slouží k rychlému odstranění textu z pole. Nastavení parametrů zůstane nezměněné. V panelu „Výstupní text“ se zobrazí řešení úlohy (zpracovaný vstupní text). Kvalitu řešení lze ohodnotit pomocí hvězd (5 hvězd = výborné). Pro potvrzení (odeslání) hodnocení klikněte na „Ohodnotit“. Po té už nelze hodnocení změnit.



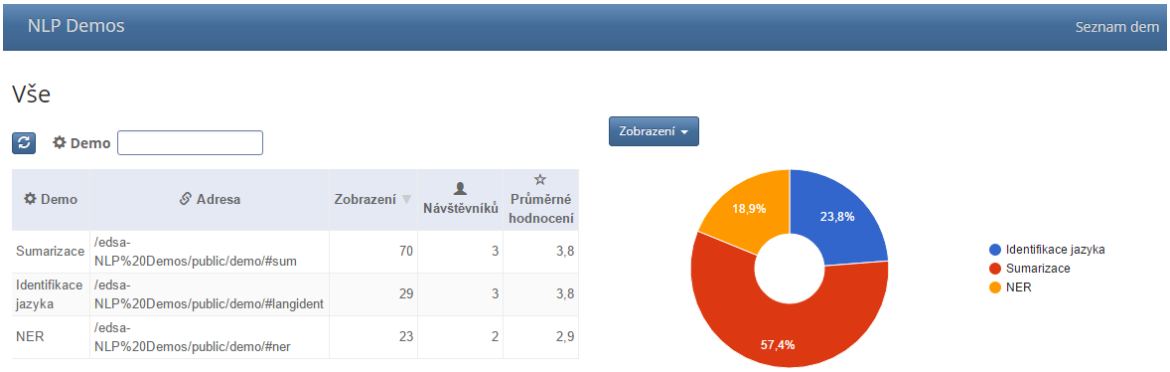
Obrázek B.3: Demo po načtení stránky



Obrázek B.4: Popisek parametru

B.2 Analytics (přehled)

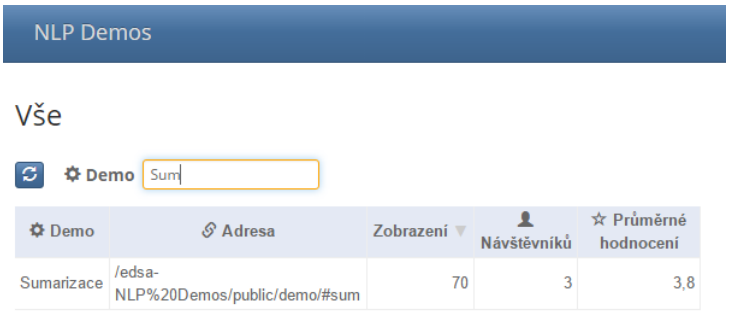
Návštěvnost dem otevřeme kliknutím na „Statistiky“ vpravo nahoře na hlavní stránce (viz. předchozí obr. B.1). Po načtení se zobrazí souhrnný přehled o všech sledovaných demech.



Obrázek B.5: Celkový přehled

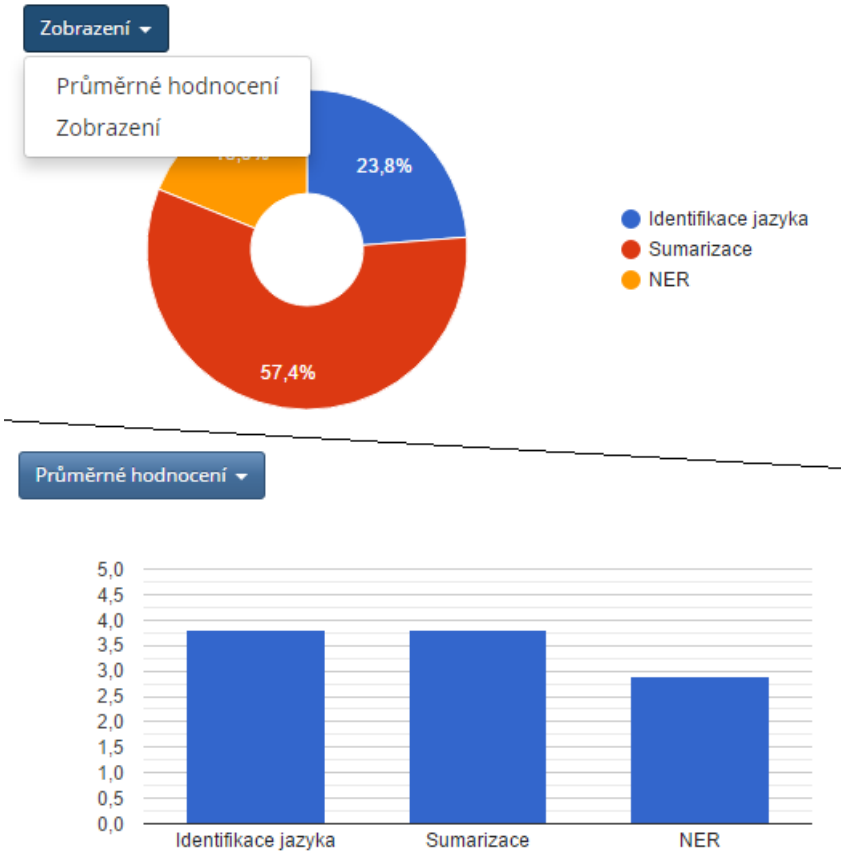
Ovládání

Záznamy v tabulce lze řadit podle kteréhokoli sloupce kliknutím na jeho název (výchozí je řazení sestupné podle počtu zobrazení). Textové pole nad tabulkou slouží k vyhledání záznamu podle názvu dema (resp. odfiltruje ty, které se neshodují) (viz. obr. B.6). Tlačítko nalevo od tohoto pole slouží k aktualizaci záznamů v tabulce.



Obrázek B.6: Vyhledávání dema

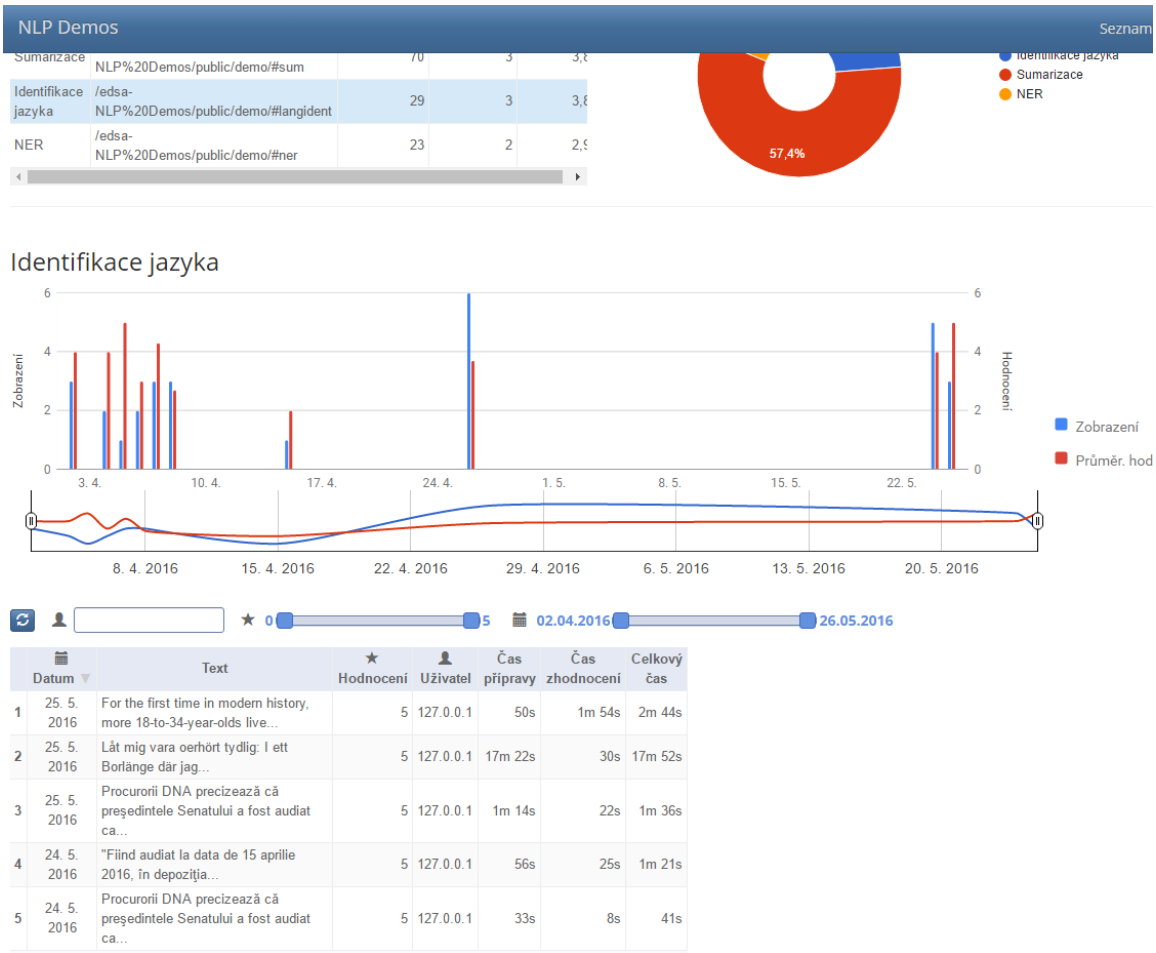
Graf vedle zobrazuje podíl návštěvnosti dem v procentech. Lze jej přepnout na porovnání celkové průměru hodnocení od uživatelů (viz. obr. B.7).



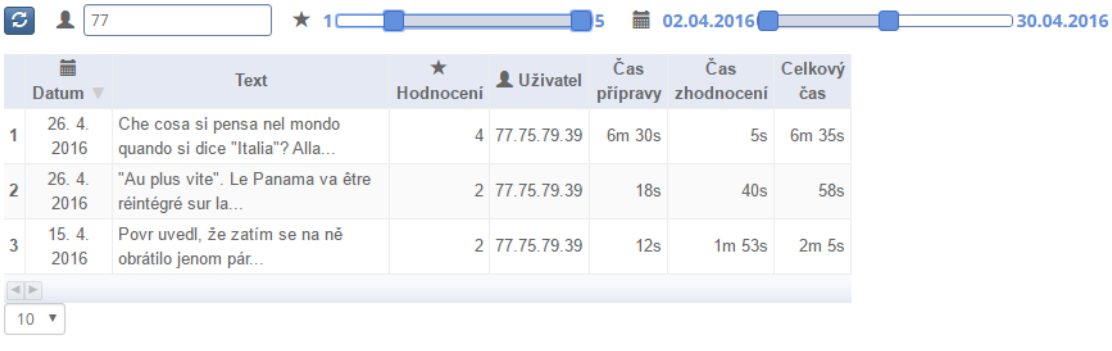
Obrázek B.7: Přepínání grafů

Zobrazení záznamů

Kliknutím na řádek v tabulce nebo na sloupec/díl grafu se načtou záznamy návštěv dema a zobrazí se pod celkovým přehledem (viz. obr. B.8). Tabulku záznamů lze řadit stejně jako v předchozím případě. Stejně tak je možné aktualizovat záznamy tabulky bez nutnosti opětovného načtení celé stránky. Nad tabulkou se nachází opět několik filtrů, které je možné kombinovat. Záznamy lze filtrovat podle IP adresy návštěvníka, hodnocení výstupu a vymezení intervalu data návštěvy (viz. obr. B.9).

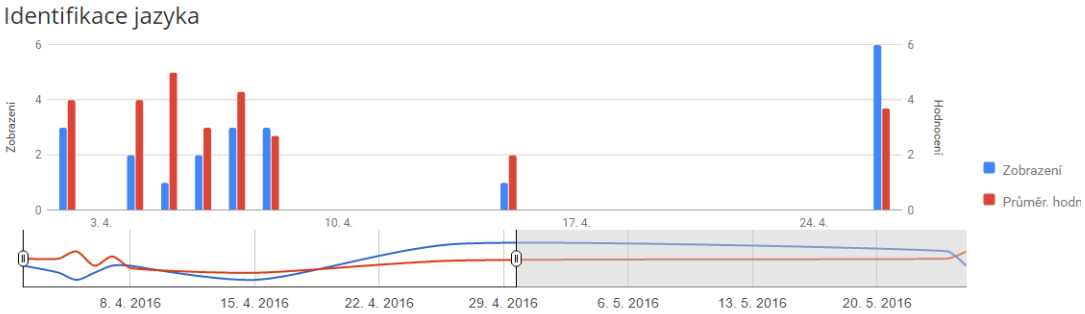


Obrázek B.8: Záznamy návštěv



Obrázek B.9: Použití filtrů

Tyto filtry nemají vliv na graf nad tabulkou. Ten na časové ose zobrazuje stejný typ dat jako grafy v celkovém přehledu. Menší spojnicový graf pod ním může sloužit jako jeho filtr. Posuvem hranic na boku lze vymezit menší časový interval (viz. obr. B.10).



Obrázek B.10: Použití filtru časové osy

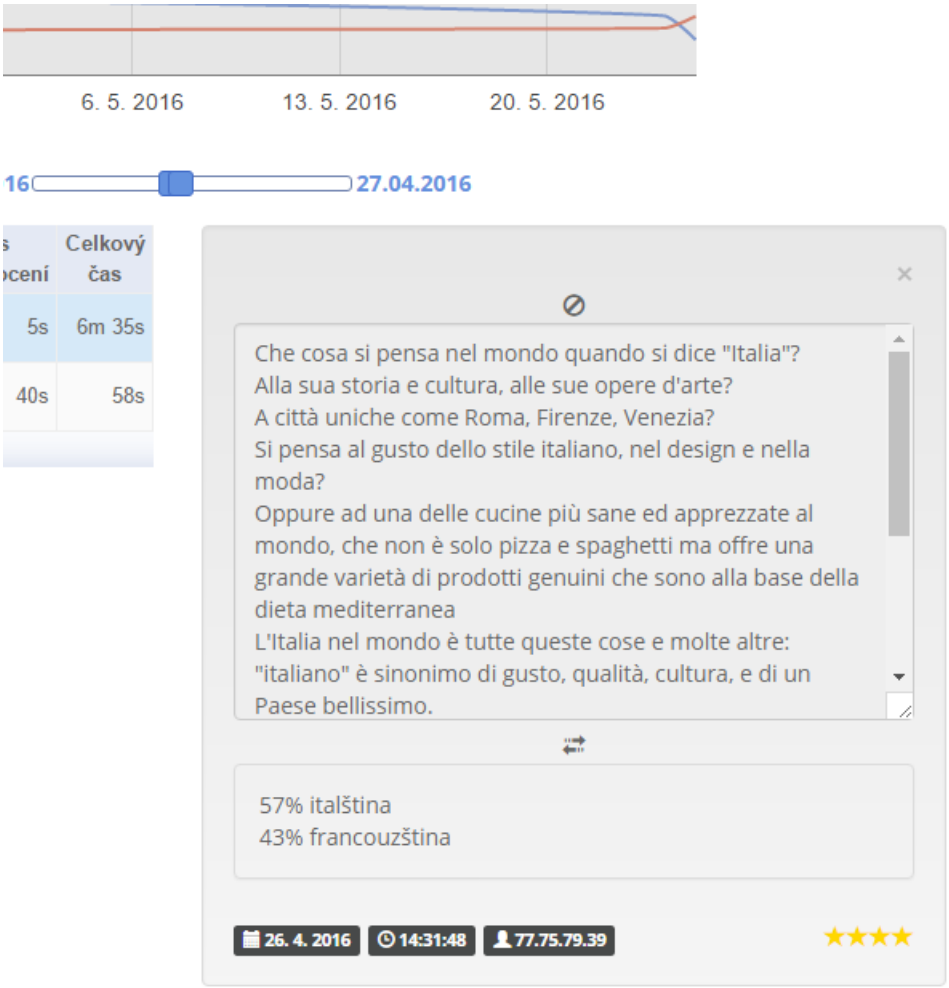
Kliknutím na sloupec v tomto grafu se nastaví filtr tabulky tak, že se zobrazí pouze záznamy z tohoto dne (viz. obr. B.11). Opětovným kliknutím na sloupec nebo někam mimo do grafu se filtr vrátí do výchozího nastavení (zobrazí se všechny záznamy).

Obrázek B.11: Výběr data podle sloupce

Detail záznamu

Kliknutím na konkrétní záznam (řádek) se načtou a zobrazí jeho detailnější informace. Detail záznamu se zobrazí vedle tabulky záznamů a obsahuje kromě data a času otevření dema, IP adresy uživatele a uděleného hodnocení navíc ještě celý

vstupní text, výstup jak ho viděl uživatel a nastavené parametry (jako dvojice `název:hodnota`). Bezparametrové demo symbolizuje ikona škrtlého kruhu (viz. obr. B.12).



Obrázek B.12: Detailní zobrazení záznamu

Poděkování

Chtěl bych poděkovat panu Doc. Ing. Josefu Steinbergerovi, Ph.D. za odborné vedení, pomoc a cenné rady při vypracování této práce.